Chapter 9:
Discrete, Vorticity-Preserving, and Stable Simplicial Fluids

Sharif Elcott Yiying Tong Eva Kanso Peter Schröder Mathieu Desbrun
Caltech

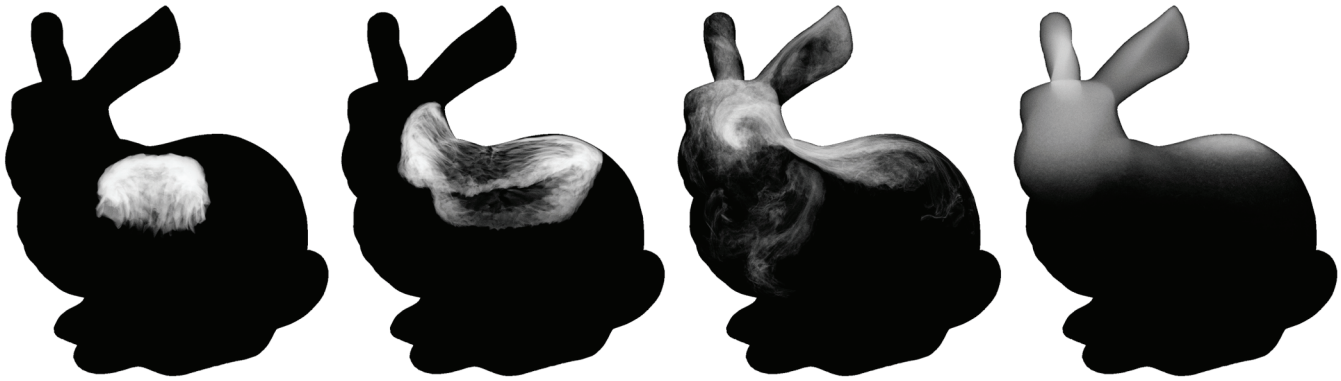


Figure 1: *Discrete Fluids: we present a novel integration scheme for fluid simulation applicable to tetrahedral meshes of arbitrary domains. Aside from resolving the exact boundaries, our approach also provides an accurate treatment of the vorticity through a discrete preservation of Kelvin's circulation theorem. Here, a hot smoke cloud rises inside a bunny shaped domain of* $32K$ *tets, significantly reducing the computational complexity of the simulation for such an intricate boundary compared to regular grid-based techniques (less than* $2s$ *per frame on Pentium IV 3GHz).*

Abstract

Visual accuracy, low computational cost, and numerical stability are foremost goals in computer animation. An important ingredient in achieving these goals is the conservation of fundamental motion invariants. For example, rigid or deformable body simulation have benefited greatly from conservation of linear and angular momenta. In the case of fluids, however, none of the current techniques focuses on conserving invariants, and consequently, they often introduce a visually disturbing numerical diffusion of *vorticity*. Visually just as important is the resolution of complex simulation domains. Doing so with regular (even if adaptive) grid techniques can be computationally delicate.

In this paper, we propose a novel technique for the simulation of fluid flows. It is designed to respect the defining differential properties, i.e., the *conservation of circulation* along arbitrary loops as they are transported by the flow. Consequently, our method offers several new and desirable properties: (1) arbitrary simplicial meshes (triangles in 2D, tetrahedra in 3D) can be used to define the fluid domain; (2) the computations are efficient due to discrete operators with small support; (3) the method is stable for arbitrarily large time steps; and (4) it preserves a *discrete circulation* avoiding numerical diffusion of vorticity. The underlying ideas are easy to incorporate in current approaches to fluid simulation and should thus prove valuable in many applications.
Keywords: Fluid Dynamics, Discrete Exterior Calculus, Computational Algorithms, Circulation Preservation

1 Introduction

It is now taken for granted that properties such as conservation of linear and angular momentum in solid mechanics simulations are a key ingredient in both numerical stability and the realism of the resulting animations. Much of the progress in this direction has been enabled by a deeper understanding of the underlying geometric structures and how they can be preserved as we go from continuous models to discrete computational realizations. So far, advances of this type have not yet deeply impacted fluid flow simulations. Current methods in fluid simulation are rarely designed to conserve *defining physical properties*. Consider, for example, the need in many methods to continually project the numerically updated velocity field onto the set of divergence free velocity fields.

1.1 Previous Work

Fluid Mechanics has been studied extensively in the scientific community both mathematically and computationally. The physical behavior of incompressible fluids is usually modeled by Navier Stokes (NS) equations for viscous fluids and by Euler equations for inviscid (non-viscous) fluids. Numerical approaches in computational fluid dynamics typically discretize the governing equations through Finite Volumes (FV), Finite Elements (FE) or Finite Differences (FD) methods. We will not attempt to review the many methods proposed (an excellent survey can be found in [Langtangen et al. 2002]) and instead focus on approaches used for fluids in computer graphics. Some of the first fluid simulation techniques used in the movie industry were based on Vortex Blobs [Yaeger et al. 1986] and Finite Differences [Foster and Metaxas 1997]. To circumvent the ill-conditioning of these iterative approaches for large time steps and achieve unconditional stability, Jos Stam [1999; 2001] pioneered in graphics the use of the *method of characteristics* for fluid advection, and of the Helmholtz-Hodge decomposition to preserve the divergence-free nature of the fluid motion [Chorin and Marsden 1979]. This extremely successful semi-Lagragian approach based on an Eulerian discretization through a regular space partitioning has led to a series of refinement over the past years. We mention the use of Galilean invariance [Shah et al. 2004], staggered grids, and monotonic cubic interpolation [Fedkiw et al. 2001], which have all significantly contributed to visual impact of fluid animations. In the wake of this success, improvements were made on the handling of the interfaces with air [Foster and Fedkiw 2001], extensions to curved surfaces [Stam 2003; Tong et al. 2003; Shi and Yu 2004] and visco-elastic objects [Goktekin et al. 2004], and goal oriented control of the fluid motion [Treuille et al. 2003; McNamara et al. 2004; Pighin et al. 2004].

However, the Stable Fluids technique is not without drawbacks. First, complex domain boundaries are difficult to handle with regular grids due to scaling issues. This can be addressed through local adaptivity of the domain [Losasso et al. 2004], but the associated

octree structures require significant overhead and lead to possible loss of accuracy. Second, regular as well as octree partitionings of space suffer from preferred direction sampling, leading to artifacts similar to aliasing in rendering. Lastly, due to numerical dissipation, the current methods do not preserve fundamental invariants aside from the divergence-free nature of the flow. While exaggerated loss of total energy is often difficult to notice, excessive diffusion of vorticity affects the motion significantly. The presence of vortices in liquids and volutes in smoke is one of the most important visual clues to our perception of fluidity. Vorticity confinement [Steinhoff and Underhill 1994; Fedkiw et al. 2001] counteracts this diffusion by locally reinjecting vorticity. Unfortunately, it is hard to control how much can safely be added back into the flow without affecting stability.

Our main argument in this paper is that a careful setup of *discrete differential quantities* together with their structural relationship (*e.g.*, well known vector calculus identities) is a necessary first step in building discrete simulation methods for fluids which can respect some of the underlying physics and in this way overcome limitations of earlier approaches. As it turns out this setup also provides guidance in designing time integration methods with attractive features such as stability and efficiency. The key ingredient to this approach is a return to the *geometric* foundations of physics.

1.2 Towards a Geometric Approach to Simulation

In recent years, there has been a renewed emphasis on the geometric structure of physical systems as a key feature for developing reliable and efficient numerical methods that better respect the underlying physics. Computational Electro-Magnetism (E&M) and Discrete Variational Mechanics, for instance, have independently demonstrated that geometric understanding of the continuous model and proper geometric discretization are crucial for obtaining stable numerical results that conserve charge, momentum, and even energy (see, for example, [Bossavit 1998; Marsden and West 2001; Kane et al. 2000; Lew et al. 2003; Fetecau et al. 2003]).

The geometric structure of Fluid Mechanics, specifically Euler's equations for inviscid fluids, has been investigated from a theoretical point of view (see [Marsden and Wenstein 1983] and references therein). In this geometric framework, vorticity plays a central role since Euler's equations can be written directly as a simple vorticity advection (see Section 2 for details). Inspired by this geometric viewpoint and the recent advances in Discrete Exterior Calculus (DEC—see [Bossavit 1998; Hirani 2003] , and Chapter 7), we propose to mimic these geometric properties on the discrete level through a *discrete differential approach to fluid mechanics*.

1.3 Contributions and Outline

In this paper, we present a radically different approach to fluid simulation based on a tailored discretization of the geometric structure of the fluid equations. We depart from the concepts of most previous computational approaches by locating physical quantities on vertices, edges, faces, or cells, depending on their geometric nature. Through a proper discrete calculus on simplicial complexes, our novel integration scheme directly manipulates intrinsically divergence-free variables, alleviating the need for a numerically-detrimental Hodge projection. Our technique offers control over the structural invariants in fluid flows thanks to our structure-preserving space and physical discretization. Finally, our novel technique fits the specific requirements of the CG community that are simplicity and unconditional stability, with high visual quality even for very large time steps.

The organization of this paper is as follows. In Section 2, we motivate our approach through a brief overview of the theory and computational algorithms for Fluid Mechanics. We propose a novel discrete fluid theory in Section 3 and we discuss the associated circulation-preserving integration algorithm in Section 4. Several numerical examples are shown and discussed in Section 5.

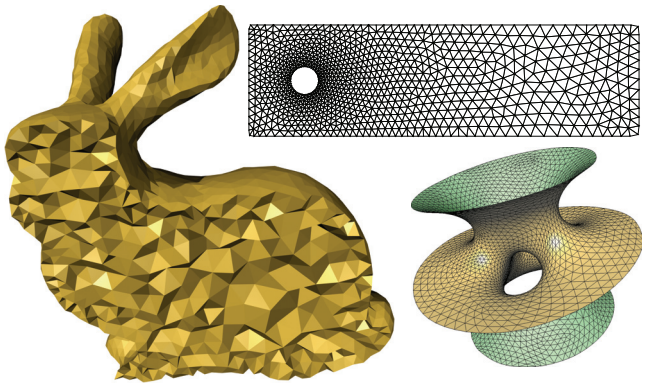


Figure 2: *Domain Mesh: our fluid simulator uses a simplicial mesh to discretize the equations of motion; (left) the domain mesh (shown as a cutaway view) used in Fig. 1; (up) a coarser version of the flat 2D mesh used in Fig. 8; (right) the curved triangle mesh used in Fig. 10.*

2 Background on Fluid Mechanics

2.1 Theory and Geometry of Euler Equations

Consider an ideal (inviscid, incompressible and homogeneous) fluid flow on a domain $\mathscr{D}$ in two- or three-dimensional physical space. The Euler equations, governing the motion of this fluid (with no external forces for now), can be written as:

$$\begin{aligned} \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u} &= -\nabla p\,, \\ div(\mathbf{u}) &= 0\,, \quad \mathbf{u} \parallel \partial\mathscr{D}\,. \end{aligned} \tag{1}$$

Here, we have set the density of the fluid $\rho = 1$ and used $\mathbf{u}$ to denote the fluid velocity, p the pressure, and $\partial\mathscr{D}$ the boundary of the fluid region $\mathscr{D}$. The pressure term in Eq. (1) can be easily dropped by rewriting the Euler equations in terms of *vorticity*. Recall first that, in traditional vector calculus notation, the vorticity $\boldsymbol{\omega}$ is defined as the curl of the velocity field; then, by taking the curl ($\nabla\times$) of Eq.(1), we obtain:

$$\begin{aligned} \frac{\partial \boldsymbol{\omega}}{\partial t} + \mathscr{L}_{\mathbf{u}}\boldsymbol{\omega} &= \mathbf{0}\,, \\ \boldsymbol{\omega} = \nabla\times\mathbf{u}\,, \quad div(\mathbf{u}) &= 0\,, \quad \mathbf{u} \parallel \partial\mathscr{D}\,. \end{aligned} \tag{2}$$

where $\mathscr{L}$ represents the Lie derivative. To put it simply, this last expression states that *vorticity is advected along the fluid flow*. Roughly speaking, vorticity measures the local rotation of a fluid parcel. We say the fluid parcel has vorticity when it spins as it moves along its path. Therefore, vorticity advection means that the local spin moves dynamically as if pushed forward by the flow.

Now, since the integral of the vorticity on a given bounded domain is equal, by Stokes' theorem, to the *circulation* around the loop enclosing the domain, one can explain the geometric nature of an ideal fluid flow in particularly simple terms: **the circulation around any closed loop $\mathscr{C}$ is conserved throughout the motion of this loop in the fluid**. This key result is known as Kelvin's circulation theorem, and is usually written as:

$$\Gamma(t) = \oint_{\mathscr{C}(t)} \mathbf{u}\cdot\mathbf{dl} = \text{constant}\,, \tag{3}$$

where $\Gamma(t)$ is the circulation of the velocity on the loop $\mathscr{C}$ at time t as it gets advected in the fluid.

Additionally, one can readily verify that Euler equations (1), equivalently (2), also preserve the total energy of the fluid which can be

written as:

$$E = \frac{1}{2}\int_{\mathscr{D}} \| \mathbf{u} \|^2 , \quad \text{or, equivalently,} \quad \mathrm{E} = \frac{1}{2}\int_{\mathscr{D}} \boldsymbol{\omega}\cdot\Delta^{-1}\boldsymbol{\omega} . \quad (4)$$

2.2 Navier-Stokes Equations

In contrast to ideal fluids, incompressible *viscous* fluids generate very different fluid behaviors. However, they can be modelled by the Navier-Stokes equations which look very similar to Euler equations:

$$\begin{aligned} \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u} &= -\nabla p + \nu\Delta\mathbf{u} , \\ div(\mathbf{u}) &= 0 , \quad \mathbf{u}|_{\partial\mathscr{D}} = \mathbf{0} . \end{aligned} \quad (5)$$

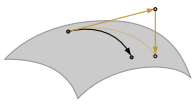
where Δ represents the Laplacian operator, and ν is a parameter called the *kinematic viscosity*. Note that various types of boundary conditions are sometimes added, depending on the chosen model. Despite the apparent similarity between these two models for fluid flows, it is important to notice that the added diffusion term dampens the motion, resulting in a slow decay of both circulation *and* total energy. This diffusion also implies that the velocity of a viscous fluid at the boundary of a domain must be null, whereas an inviscid fluid could have a non-zero tangential component on the boundary. Here again, one can avoid the pressure term by taking the curl of the equations, finally yielding :

$$\begin{aligned} \frac{\partial \boldsymbol{\omega}}{\partial t} + \mathscr{L}_{\mathbf{u}}\boldsymbol{\omega} &= \nu\Delta\boldsymbol{\omega} , \\ \boldsymbol{\omega} = \nabla\times\mathbf{u} , \quad div(\mathbf{u}) &= 0 , \quad \mathbf{u}|_{\partial\mathscr{D}} = \mathbf{0} . \end{aligned} \quad (6)$$

2.3 Stable Fluids Discretization

The different variants of the original Stable Fluids algorithm [Stam 1999] are all based on a class of discretization approaches known in Computational Fluid Dynamics as fractional step methods. In order to numerically solve the Euler equations over a time step h, they proceed in two stages. They first update the velocity field assuming the fluid is inviscid and disregard the divergence-free constraint of Eq. (1). Then, the resulting velocity is projected onto the closest divergence-free flow (in the $\mathscr{L}^2$ sense) through a Helmholtz-Hodge decomposition.

Although each step of this approach is unconditionally stable, one of the consequences of this fractional integration is the exaggerated energy loss it creates: advecting velocity before reprojecting onto a divergence-free field creates major energy loss and, more importantly in a CG context, diffusion of vorticity as reported in [Fedkiw et al. 2001] for instance. One can understand this numerical flaw through the following geometric argument: physically speaking, the solution of Euler equations are *geodesic* (i.e., shortest) paths *on* the manifold of all possible divergence-free flows; advecting the fluid *out* of the manifold is not a proper substitute to this intrinsic constrained minimization, even the post re-projection is, in itself, exact.



2.4 Our Geometric Approach

Given the difficulties discussed above a natural question is whether these problems can be overcome by designing more careful discretizations which are better suited to maintain the underlying geometric structures—for example, flows that are always divergence free without the need to continually project onto the space of divergence free fields and incurring the associated losses. Or, perhaps even more importantly for visual simulation, one may wonder if it is possible to find discretizations that conserve circulation.

It is known that it is not possible to exactly preserve momenta *and* total energy simultaneously in the discrete setting [Zhong and Marsden 1988]. However, stable numerical techniques have been reported to exactly preserve momenta while keeping the total energy remarkably close to constant [Marsden and West 2001]. Such properties are obtained through the use of *variational integrators*, *i.e.*, by guaranteeing a discrete version of the least-action principle. Their design proceeds by keeping underlying geometric structures intact as one goes from the continuous to the discrete formulation. More precisely, appropriate geometric discretization of the physics allows one to construct discrete analogs of momenta and energy. Equipped with these discrete structure-preserving quantities, integration schemes can then be designed to enforce their invariant nature. We will loosely follow this path by using vorticity as our primary simulation variables (see Eq. (2)) and designing a time integration scheme which will conserve circulation (Eq. (3)) through vorticity advection. As a by-product our velocity fields will be divergence free *without* any need to continually reproject to keep this property. For comparison, and to the best of our knowledge, none of the integration schemes proposed in CFD have been designed to satisfy the conservation properties of the underlying equations (aside from the limited case of linearized NS equations [Morton and Roe 2001]).

We will limit ourselves to the investigation of such a scheme without focusing on the separate issue of order of accuracy. Coming up with an integration scheme that is of higher-order accuracy will be the object of further research.

3 Spatial and Physical Discretization

In this section, we define proper discrete analogs for the velocity and vorticity fields $\mathbf{u}$ and $\boldsymbol{\omega}$ on simplicial grids. We emphasize that the construction of these discrete fields is quite general as it does *not* depend on the assumption of an ideal fluid.

3.1 Space Discretization

We discretize the spatial domain (in which the flow takes place) using a locally oriented simplicial complex, *i.e.*, either a tet mesh for 3D domains or a triangle mesh for 2D domains, and refer to this discrete domain as $\mathscr{M}$ (see Figure 2). The domain may have non-trivial topology, *e.g.*, it can contain tunnels and voids (3D) or holes (2D), but is assumed to be compact. To ensure good numerical properties in the subsequent simulation we require the simplices of $\mathscr{M}$ to be well shaped, *i.e.*, the aspect ratios of tets (resp., triangles) are not near zero. This assumption is quite common since many numerical error estimates depend heavily on the element quality. Collectively we refer to the sets of vertices, edges, triangles, and tets as V, E, F, and T.

We will also need the concept of a *dual mesh*. It associates with each original simplex (vertex, edge, triangle, tet, respectively) its dual (dual cell, dual edge, dual face, and dual vertex, respectively) (see Fig. 3). The geometric realization of this dual mesh is defined as follows: we take circumcenters of tets as the dual vertices and the Voronoi cells as the dual cells; dual edges are then line segments connecting dual vertices of neighboring tets and dual faces are the faces of the Voronoi cells.

3.2 Physical Quantity Discretization

In order to faithfully capture the geometric structure of fluid mechanics on the discrete mesh, we need to define the usual physical quantities such as velocity and vorticity, for example, through *integral values over the simplices of the mesh* $\mathscr{M}$. This is the sharpest departure from traditional numerical techniques in CFD: we not only use values at nodes and tets (as in FEM and FVM), but also allow association (and storage) of field values at *any* appropriate

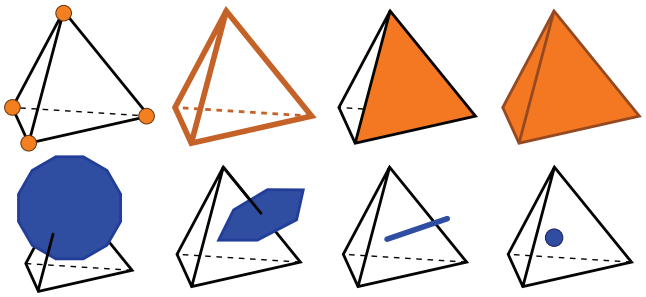


Figure 3: *Primal and Dual Cells: the simplices of our mesh are vertices, edges, triangles and tets (up); their circumcentric duals are dual cells, dual faces, dual edges and dual vertices (bottom).*

simplex. In particular, some quantities will live on edges (primal or dual), others on faces. Before our formal exposition of how these quantities are defined, we motivate our discretization choice with some physical intuition first.

Velocity as a Discrete Flux We wish to define a discrete quantity that encodes the fluid velocity field while being intrinsic to the mesh, i.e., with a coordinate-free representation. To do this, we consider the *flux* of the fluid, *i.e.*, the mass of fluid transported across a given surface per unit time. Note that the flux across a surface incorporates the area of the surface, indicating that it is an integrated quantity rather than a pointwise quantity. On the discrete mesh, a natural place to store the flux is on the triangles of a tet mesh (or edges in a 2D triangle mesh). This discrete flux is coordinate free, *i.e.*, it does not depend on whatever local or global coordinate frame we choose (vectors on the other hand have different representations depending on the coordinate system).

One can equivalently think of the flux as living on dual edges; the proper term should, however, be *circulation* in that case: recall that a dual edge connects the dual vertices associated with the two incident tets and is thus (in a sense) "transverse" to a shared primal face. This point of view is reminiscent of the staggered grid method used in [Fedkiw et al. 2001] and other non-collocated grid techniques (see [Goktekin et al. 2004]). In the staggered grid approach one does not store the x,y,z components of a vector at nodes but rather associates them with the corresponding grid faces. We may therefore think of the idea of storing fluxes on the triangles of our tet mesh as a way of *extending* the idea of staggered grids to the more general simplical mesh setting. This was previously exploited in [Bossavit and Kettunen 1999] in the context of E&M computations. It also makes the usual no-transfer boundary conditions easy to encode: boundary faces experience no flux across them. Encoding this boundary condition when storing velocity vectors at vertices is far more cumbersome.

Divergence as Net Flux on Tets Given the incompressibility of the fluid, the velocity field must be divergence-free ($\nabla \cdot \mathbf{u} = 0$), hence the integral of $\nabla \cdot \mathbf{u}$ is constant. We would like to write this condition in the discrete setting. Given the flux across all the faces of a tet, the integral of the divergence over the tet, or, said differently, the net flux of the tet, becomes particularly simple. According to the generalized Stokes' theorem this integral equals the sum of the integral of the flux on all four faces. Divergence is thus naturally thought of as a value at each tet (see Fig. 4). Physically speaking, the notion of a divergence-free velocity field is equivalent to saying that, at each tet, everything that gets in must get out.

Vorticity as Flux Spin Finally we need to define vorticity on the mesh. To see the physical intuition behind our definition, consider an edge in the mesh. It has a number of faces incident on it, akin to a paddle wheel (see Figure 4). The flux on each face contributes a net torque to the edge. The sum of all these, when going around an edge, is the net torque that would "spin" the edge. We

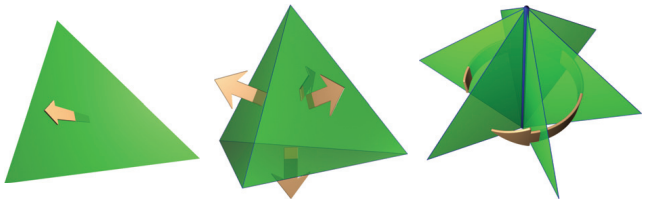


Figure 4: *Discrete Physical Quantities: in our discrete geometric discretization of fluid mechanics, fluid flux lives on faces (left), divergence lives on tets (middle), and vorticity lives on edges (right).*

can thus give a physical definition of vorticity as the sum of fluxes on all faces incident to a given edge: this quantity is now associated with primal edges—or, equivalently, dual faces.

3.3 Discrete Differential Structure

The definition of these intuitive physical quantities living at different simplices on the mesh can be made precise through the definition of a *discrete differential structure*. In this framework, a mesh is seen as the only given structure to work with, with no reference to the continuous space that it approximates, and Discrete Exterior Calculus (DEC) defines a coherent calculus on the mesh using only discrete combinatorial and geometric operations [Munkres 1984; Hirani 2003; Tong 2004][1]. Although we can not discuss at length such a vast mathematical machinery, we briefly cover the fundamental aspects and the discrete differential operators we need to link flux and vorticity (just as in the differential case). For a comprehensive exposition, we refer the interested readers to Chapter 7 on discrete differential forms.

Discrete Forms As Integrals Before we discuss the discrete exterior structure inherent to the mesh, we briefly review exterior forms in the continuous setting (for a more comprehensive discussion, see, for example, [Abraham et al. 1988]). To this end, recall that, given a three-dimensional space, a 0-form is simply a function on that space; a 1-form ω is a proxy to a vector field; a 2-form, or area-form, is to be *integrated over a surface*, that is, it can be viewed as a proxy to the vector perpendicular to that surface; and, a 3-form, or volume-form, is *to be integrated over a volume* and is viewed as a function.

A *discrete* differential k-form, $k = 0,1,2$, or 3, is then the evaluation (i.e., the integral) of the differential k-form on all k-cells, or k-simplices. In practice, discrete k-forms can simply be considered as *vectors of numbers* according to the simplices they live on: 0-forms, live on vertices, and are expressed as a vector of length $|V|$; and correspondingly, 1-forms live on edges (length $|E|$), 2-forms live on faces (length $|F|$), and 3-forms live on tets (length $|T|$). Dual forms are treated similarly. In the examples above, flux is thus a primal 2-form (integrated over faces), vorticity a dual 2-form (integrated over dual faces), and divergence a primal 3-form (integrated over tets).

Discrete Differential Calculus on Simplicial Meshes These *discrete* differential forms can now be used to build the tools of calculus. The mesh has a natural structure, called the DeRham complex, which offers discrete operators on discrete forms, mimicking the continuous setting. At the core of its construction is the definition of the discrete d_k operators (analog to the continuous exterior derivative).

Discrete Exterior Derivatives A key ingredient to define this discrete derivative is Stokes' theorem on a k-form:

$$\int_{\sigma_{k+1}} d_k \omega_k = \int_{\partial_{k+1}\sigma_{k+1}} \omega_k,$$

[1] Although we use many notions from DEC in this paper, the theory presented here is self-contained and does *not* assume previous knowledge of this machinery.

where σ_k denotes a k-cell while ω_k is a k-form. Stokes' theorem states that the integral of $d_k\omega_k$ (a $(k+1)$-form) over a $(k+1)$-cell equals the integral of the k-form ω_k over the *boundary* of the $(k+1)$-cell. The boundary of a $(k+1)$-cell of course consists of k-cells, making everything well defined. Stokes' theorem can thus be used as a way to *define* the d operator in terms of the boundary operator ∂. Or, said differently, once we have the boundary operator, the operator d follows immediately if we wish Stokes' theorem to hold on the simplicial complex.

To use a very simple example, consider a 0-form ω_0, *i.e.*, a function giving values at vertices. With that $d_0 f_0$ is a 1-form which can be integrated along an edge (say with end points denoted a and b) and Stokes' theorem states the well known fact

$$\int_{[a,b]} d_0 f_0 = f_0(b) - f_0(a).$$

The right hand side is simply the evaluation of the 0-form f_0 on the boundary of the edge, *i.e.*, its endpoints (with appropriate signs indicating the orientation of the edge). Actually, one can define a hierarchy of these operators that mimic the operators given in the continuous setting by the gradient (∇), curl ($\nabla\times$), and divergence ($\nabla\cdot$), namely,

- d_0: maps 0-forms to 1-forms and corresponds to the **Gradient**;
- d_1: maps 1-forms (values on edges) to 2-forms (values on faces). The value on a given face is simply the sum (by linearity of the integral) of the 1-form values on the boundary (edges) of the face with the signs chosen according to the local orientation. d_1 corresponds to the **Curl**;
- d_2: maps 2-forms to 3-forms and corresponds to the **Divergence**.

From this basic setup, we see that all that is required now is to define the boundary operator. This is done using *incidence matrices*, which then act on the vectors of our discrete k-forms. For example d_0 follows as the incidence matrix of vertices and edges. The incidence matrix has $|E|$ rows and $|V|$ columns. Each row contains a $+1$ and -1 for the two end points of the given edge (and zero otherwise). The sign is determined from the orientation of the edge. Similarly for the incidence relations of edges and faces: this is a sparse matrix with $|F|$ rows and $|E|$ columns, with appropriate $+1$ and -1 entries according to the relation of the orientation of edges as one moves around a face (according to its orientation). More generally d_k is the incidence matrix of k-cells on $k+1$-cells.

Implementation Given the oriented mesh $\mathscr{M}$ all that is required to implement the necessary operators is to assemble the incidence matrices. Note that these are sparse and contain only entries of type 0, $+1$, and -1. As we pointed out earlier, care is required in assembling these incidence matrices: the orientation must be taken into account in a *consistent* manner. A simple debugging sanity check (necessary but not sufficient) is to compute consecutive products: d_0 followed by d_1 must be a matrix of zeros, as must be d_1 multiplied by d_2. This reflects the fact that the boundary of any boundary is the empty set. It also corresponds to the calculus fact that curl of grad is zero as is divergence of curl.

Hodge Stars We need one last operation to complete our machinery. We noted earlier that fluxes can be seen as 2-forms on primal faces, or as dual 1-forms on dual edges. This desirable projection of a primal k-form to a conceptually-equivalent dual $(3-k)$-form is called the k^{th} *Hodge star*. We will denote $\star_0$ (resp., $\star_1, \star_2, \star_3$) the Hodge star taking a 0-form (resp., 1-form, 2-form, and 3-form) to a dual 3-form (resp., dual 2-form, dual 1-form, dual 0-form). These linear operators, describing the local metric, can also be stored in sparse matrices. In this paper, we will use what is known as the *diagonal Hodge stars* [Bossavit 1998] as they are particularly simple to compute: only the diagonal terms are non-zero, and they are equal to the ratio of sizes of corresponding dual and primal cells: let vol(.) denote the volume of a cell (*i.e.*, 1 for vertices, length for edges, area for 2D cells, and volume for 3D cells), then the diagonal matrix entries are

$$(\star_k)_{qq} = \text{vol}(\tilde{\sigma}_q)/\,\text{vol}(\sigma_q)$$

where σ is a primal k-simplex, and $\tilde{\sigma}$ its dual. The subscript q indicates the index in the list of all cells. The Hodge star matrices are therefore symmetric positive definite. More accurate metric representations can be used, leading to less sparse Hodge stars; however, for our purposes, this one is sufficient. It also has the nice property that its inverse is trivial to compute.

The Hodge stars allow us to go from primal forms to dual forms of complementary dimension. In order to complete the deRham complex, we now need to define the *dual* version of the operators d_k; i.e., their equivalents on the dual side. These operators turn out to be quite simple: one can prove that the *transpose* of the d_k operators (and therefore, the transpose of their matrices) serve this purpose [Bossavit 1998]. Figure 5 summarizes the various operations between forms that we just defined. Equipped with these

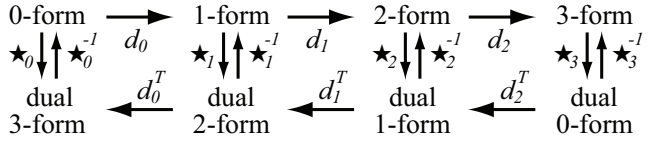


Figure 5: *Discrete Differential Calculus: the operators d_k and $\star_k$ allow proper manipulation of arbitrary discrete forms on the domain mesh.*

matrices, we can now formalize the definition of the various quantities we need for fluid mechanics—and see how they parallel their continuous analogs.

3.4 Revisiting Fluid Discretization

The fluid velocity $\mathbf{u}$ is treated as a flux, *i.e.*, as a 2-form. It is therefore represented by a vector U of values on faces (size $|F|$). Since we store fluxes on faces, the *circulation* can be derived as values on dual edges through $\star_2 U$ (Hodge star of the 2-form U). *Vorticity*, typically a 2-form in fluid mechanics [Marsden and Wenstein 1983], is easily computed by summing this circulation along the dual edges that form the boundary of a dual face. So we store the vorticity (seen as a dual 2-form) on dual faces. In other words, our choice of flux representation imposes that ${d_1}^T \star_2 U$ be used to represent $\boldsymbol{\omega} = \nabla \times \mathbf{u}$ (see Fig. 5). This is a vector Ω of size $|E|$, representing the vorticity on each dual face. These formal discretizations match our physically-motivated definitions previously given in Section 3.2.

With the appropriate Hodge star, we can go from a 2-form to a dual 1-form and vice-versa, so the reader may notice that it is in a sense equivalent to use a 2-form (resp., a dual 2-form) to represent a flux or to use the associated dual 1-form (resp., primal 1-form) to represent circulation. The reason we chose a 2-form instead of dual 2-form for vorticity is that it is easier to represent the boundary of our 3D domain by faces rather than dual faces.

3.5 From Vorticity Back To Flux

We have just seen how the vorticity can be directly derived from the set of all face fluxes. However, during the simulation, we will also need to recover flux *from* vorticity. For this we employ the Helmholtz-Hodge decomposition theorem, stating that any vector field $\mathbf{u}$ can be decomposed into three components (given appropriate boundary conditions)

$$\mathbf{u} = \nabla\phi + \nabla \times \boldsymbol{\psi} + \mathbf{h}.$$

A generalization to nD-domains and for k-forms reads as follows:

$$f_k = d_{k-1}\phi_{k-1} + \star_k{}^{-1} {d_k}^T \star_{k+1} \psi_{k+1} + h_k \qquad (7)$$

Because of our use of 2D fluxes, we only need the latter for $k = 2$. For the case of incompressible fluids (i.e., with zero divergence), two of the three components are sufficient to describe the velocity field: the curl of a vector potential and a harmonic field. This implies that when decomposing the 2-form U, we may set ψ_3 to 0. If the topology of the domain is trivial, we can furthermore ignore the harmonic part h_2 (we will discuss a full treatment of arbitrary topology in Section 4.8), leaving us with $U = d_1\phi_1$.

Thus, we can recover the velocity field solely from the vorticity by solving a Poisson equation to get the potential ϕ_1 and then applying the curl operator to the potential. The Poisson equation to solve for the 1-form ϕ (values on primal edges) is as follows:

$$(\star_1 d_0 \star_0{}^{-1} d_0^T \star_1 + d_1^T \star_2 d_1)\phi_1 = d_1^T \star_2 U = \omega \qquad (8)$$

To arrive at this equation, we applied $d_1^T \star_2$ to both sides of Eq. (7), and set the gauge of this Poisson problem as ${d_0}^T \star_1 \phi_1 = 0$. As the Laplacian Δ in differential calculus is $d \star d \star + \star d \star d$, one can readily verify that the previous equation is, indeed, a discrete version of the Poisson equation: it literally corresponds to $\Delta\phi_1 = (\nabla\nabla\cdot - \nabla\times\nabla\times)\phi_1 = \nabla\times\mathbf{u}$. Notice that the left-side matrix is *symmetric and sparse*, thus ideally suitable for fast numerical solvers.

Our linear operators (and, in particular, the discrete Laplacian) differ sharply from another discrete Poisson setup on simplicial complexes proposed in [Tong et al. 2003]: the ones we use have smaller support, which results in sparser and better conditioned linear systems [Bossavit 1998]—an attractive feature in the context of numerical simulation.

3.6 Interpolating Velocity and Circulation

So far we have only defined physical quantities as values on simplices. Although most computations can be carried out in this format, evaluation of such quantities *anywhere in space* is also necessary in practice.

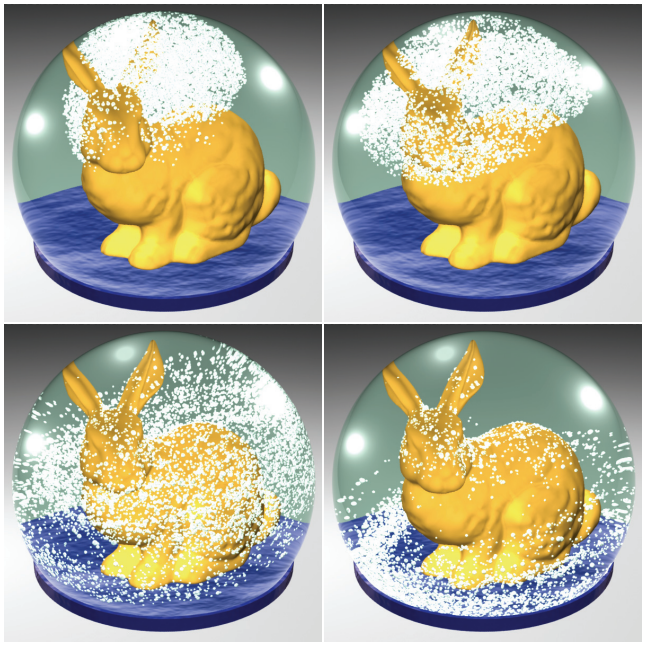


Figure 6: *Bunny Snow Globe: the snow in the globe is advected by the inner fluid, initially stirred by a vortex to simulate a spin of the globe.*

Piecewise-Constant Velocity Field When considering the fluxes on the primal mesh, we can interpolate within each tet using the usual 1-form linear basis functions for discrete forms, known as Whitney forms, and described in detail in [Bossavit 1998] and in these course notes in Chapter 7. These interpolating basis functions are piecewise linear within each tet, and easy to compute. However, in our context of incompressible fluids, it turns out that we do not even need to use them. Since our velocity field is divergence free (the sum of the four fluxes on a tet equals 0), it can be shown that this piecewise-linear interpolation of the velocity field is *piecewise constant within each tet*. That is, there is a unique vector $\mathbf{u}_i$ per tet T_i that simultaneously *agrees with all four fluxes*. It is thus a trivial matter to deduce these vectors inside the tets given the vector U of all fluxes, rendering the Whitney forms unnecessary. Notice finally that the normal component to a face F_i of such a tet-based vector agrees (by definition) with the normal component of its neighboring tet through F_i, as they must both be equal to the flux of the velocity field on that face; therefore, advection along this velocity field is properly defined (i.e., you can always step over a face and will never get stuck somewhere) and easy to compute.

Piecewise Rational Circulation We will also need to interpolate the circulation from dual edges to the whole space. Unfortunately, Whitney forms are defined *only* for primal forms. In order to bypass this limitation, we propose a novel dual 1-form interpolant based on generalized barycentric coordinates. Taking a dual cell in isolation, note that each vertex of this dual cell has 3 dual edges incident on it. Given values of circulation on these adjacent dual edges, there exists a unique vector (or covector, to be mathematically correct) at the vertex that will fit these circulations. That is, we find the vector whose projection onto each dual edge is equal to the magnitude of the circulation along that edge. This amounts to reconstructing a vector based on its projection onto 3 independent vectors. Coincidentally, in our application (and once again, because of the divergence-free property), this vector turns out to be the vector value $\mathbf{u}_i$ of the velocity field defined in the tet associated with this dual vertex—we already have stored this value in the tet, and have no need to recompute it on the fly. With these vectors at each dual vertex, we can now *interpolate* them using generalized barycentric coordinates on 3D polytopes as recently proposed in [Warren et al. 2004]. This technique offers a fast evaluation of weights at each corner of an arbitrary polytope that provide a smooth interpolation of the corner values. It is also proved that this interpolation is *linear acurate*, i.e., it reconstructs exactly constant and linear fields. Finally, because these 3D barycentric weights have the property to degenerate into their 2D equivalents on the polytope's faces, the reader can verify that such an interpolation of circulation does fit the initial values of the circulation on dual edges, providing a good and fast computational method to handle circulation. Notice finally that this interpolation, just like their (simpler) primal equivalent (Whitney forms), are only tangentially-continuous across dual edges, reflecting that a dual 1-form has only meaning as a circulation as already explained in Chapter 7.

4 A Circulation-Preserving Integration

Once the proper discretization of space has been defined, we can now turn our attention to the actual integration of the Euler and Navier-Stokes equations. We propose a numerical integration scheme that solves Eq. (2) and preserves the circulation as stated in Eq. (3). We give details on how to efficiently implement this novel integration scheme.

4.1 Rationale: Vorticity Advection

Equipped with the spatial discretization defined above, we want to integrate the fluid equations. As noticed earlier, we wish to avoid having to resort to a projection to divergence-free (i.e., incompressible) flows. Therefore, what is truly needed is an *integration of the vorticity of the fluid*: this particular physical variable is by nature

divergence-free, and we have shown in Section 3.5 that the flow itself can be entirely derived from its vorticity (modulo a proper treatment of the genus of the domain and boundary condition, which we will detail in Section 4.8). In other words, and as we cover next, our approach can be summarized as effectively performing a *vorticity advection*.

Discrete Loops Guided by Kelvin's theorem, we wish to drive the integration by preserving the circulation along loops as they are advected in the flow. Since we are now working in a discrete space, satisfying this property for *any* loop $\mathscr{C}$ does not make sense. Remember that we are limited by the resolution of the spatial discretization that the mesh $\mathscr{M}$ provides and that we decided to describe the vorticity as a dual 2-form. Thus, we propose to *satisfy Kelvin's property on each boundary of dual 2-cells*. These Voronoi loops can indeed generate *any* discrete, dual loop C : the sum of adjacent loops is a larger, outer loop since all the interior edges cancel out due to opposite orientation as sketched in Fig. 7(right). Consequently, preserving Kelvin's theorem on each of these Voronoi loops will provide a discrete analog of the continuous case.

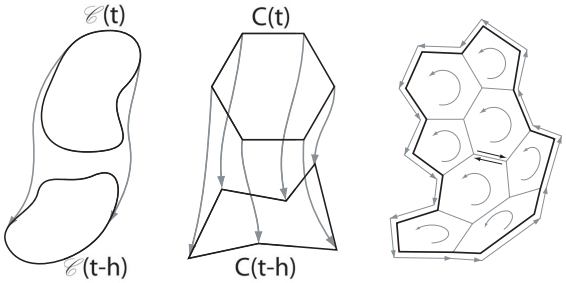


Figure 7: *Kelvin's Theorem: (left) in the continuous setting, the circulation on any loop being advected by the flow is constant. (middle) our discrete integration scheme enforces this property on each Voronoi loop, (right) thus on any discrete loop.*

Backtracking Discrete Loops For a given discrete Voronoi loop $C_i(t)$ considered at time t, we can conceptually *backtrack* it in time to find the loop $C_i(t-h)$ it originated from if the fluid velocity is assumed constant during the time step. Notice that this amounts, in the discrete case, to backtracking each circumcenter, as the piecewise-linear loops are exclusively defined by their positions. Ensuring circulation preservation is now a trivial matter: we simply have to compute the *current* circulation of this backtracked loop $C_i(t-h)$ and assign this value to the loop $C_i(t)$. By construction, we have *advected* the circulation along the flow. Using Stokes' theorem, this circulation is also the integral of the vorticity over the Voronoi face: thus we have formally found the new, advected vorticity. Notice that this backtracking is similar in spirit to the original Stable Fluids approach, but with a fundamental difference: the *vorticity* is advected instead of velocity. Again, its divergence-free nature makes it the natural variable to advect to avoid spurious numerical diffusion.

4.2 Setup and Pseudocode

An implementation of this vorticity advection algorithm requires rather usual data structures. We input a tet mesh $\mathscr{M}$ of the domain first, and start the preprocessing stage by storing the (signed) incidences between all simplices. The incidence matrices d_k described in Section 3.3 are subsequently stored. We also precompute the position of the circumcenter of each tet as we will repeatedly need them throughout the algorithm. Finally, we assemble the dual (Voronoi) cell of each vertex as they are used in the generalized barycentric coordinates interpolation described in Section 3.6. The integration from time t to time $t+h$ is then performed by successive updates of the vorticity according to the following pseudocode, starting from an initial set of fluxes U_t and corresponding vorticities Ω_t:

- **Advect Vorticity** (see Section 4.3 and Fig. 7)
  1. Backtrack each circumcenter through the flow, along the current piecewise-linear primal velocity field defined by U_t.
  2. Integrate circulation along each backward-advected Voronoi loop using numerical quadrature.
  3. Deduce the advected vorticity on each edge accordingly, and store the result in Ω^A_{t+h}
- **Add Body Forces**: From a set of external body forces (gravity, buoyancy) on each face, we further update the previous vorticity and store it in Ω^B_{t+h} as explained in Section 4.4.
- **Apply Diffusion** If we wish to simulate NS equations, we add an (optional) diffusion step on the resulting vorticity field to get Ω_{t+h} (see Section 4.5). Otherwise, we set $\Omega_{t+h} = \Omega^B_{t+h}$.
- **Convert Vorticity to Fluxes** We finally need to update the velocity field U_{t+h} for the next step, by converting the final value of Ω_{t+h} as detailed in Section 4.6.

For each item of this pseudocode, we now give details on what is involved and how it relates to the machinery previously developed.

4.3 Vorticity Advection

As sketched earlier, we can compute the new, advected vorticity on each edge through the following procedure: given the current velocity field U_t, we calculate the new vorticity on each dual face at time $t+h$ by backtracking its boundary (Voronoi) loop $C_i(t)$ through U_t and integrating the current circulation around $C_i(t-h)$ using numerical quadrature. The backtracking of all the loops is done by simply backtracking each tet's circumcenter through the flow using, for instance, a simple Euler integration. Note that this is particularly easy as the primal, divergence-free velocity field is constant per tet (see Section 3.6). We then use a simple quadrature to compute the circulation of the piecewise-linear loop defined by the backtracked circumcenters. The loop is considered as the union of each segment going from one backtracked circumcenter to the next. Using the generalized barycentric interpolation described in Section 3.6, we evaluate the interpolated velocity field at each segment end point. The circulation over each segment is then taken as the average of the velocity field at its two endpoints dotted with the segment. The new circulation around the loop is just the sum of the circulations on all these sample segments, giving us the advected vorticity Ω^A_{t+h}.

The quadrature accuracy could be easily improved by increasing the number of quadrature points on each segment, or by formally computing the circulation on the resulting piecewise-linear loops (a numerically expensive procedure as the circulation is locally expressed as a rational polynomial due to the necessary use of generalized barycentric coordinates). However, note that our spatial discretization is entirely based on linear basis functions; a linear-accurate quadrature suffices, as our numerical tests confirmed.

Note that one would be tempted to shortcut this quadrature by simply using the circulation computed from the primal velocity field. Unfortunately, such a procedure introduces significant inaccuracy since even in the case of stationary flows, the circulation would not be exactly preserved: the use of generalized barycentric coordinates and linear-accurate quadrature is a computationally-efficient *must*.

4.4 External Body Forces

The use of external body forces, like buoyancy, gravity, or stirring, is common practice to create interesting motions. Incorporating external forces into Eq. (5) is, fortunately, straightforward, resulting in:

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u} \;=\; -\nabla p + \nu\Delta\mathbf{u} + \mathbf{f}\,.$$

Again, taking the curl of this equation allows us to recast this equation in terms of vorticity:

$$\frac{\partial \boldsymbol{\omega}}{\partial t} + \mathscr{L}_{\mathbf{u}}\boldsymbol{\omega} \;=\; \nu\Delta\boldsymbol{\omega} + \nabla\times\mathbf{f}\,. \tag{9}$$

Thus, we note that an external force influences the vorticity only through the force's curl (the $\nabla\cdot\mathbf{f}$ term is compensated for by the pressure term keeping the fluid divergence-free). Thus, if we express our forces through the vector F of their resulting fluxes in each face, we can directly add the forces to the domain by incrementing Ω^A_{t+h} by the circulation of F, i.e.:

$$\Omega^B_{t+h} = \Omega^A_{t+h} + h\, d_1^T \star_2 F.$$

4.5 Adding Diffusion

If we desire to simulate a viscous fluid, we must add the diffusion term present in Eq. (6). Note that previous methods were sometimes omitting this term because their numerical dissipation was already creating (uncontrolled) diffusion. In our case, however, this diffusion needs to be properly handled if viscosity is desired. This is easily done through an unconditionally-stable implicit integration as done in Stable Fluids (i.e., we also use a fractional step approach). Using the discrete Laplacian in Eq. (8), we simply solve for the diffused vorticity Ω_{t+h} using the following linear system:

$$(I - \nu h\Delta)\Omega_{t+h} = \Omega^B_{t+h}.$$

4.6 Converting Vorticity Back To Velocity

Finally, given the updated value of the vorticity Ω_{t+h}, we need to update the corresponding velocity field. This step is straightforward by solving the linear system given in Eq. (8), and taking the circulation of the resulting potential field ϕ_1 around each face to derive the new set of fluxes U_{t+h}.

One may argue that solving this Poisson equation is strictly equivalent to the projection step in Stable Fluids. While it is true that this step has the same computational cost, their respective roles are *very* different: while the Poisson equation is used as a projection in Stable Fluids, it is used as a mere conversion in our case, and does *not* incur numerical dissipation.

4.7 Boundary Conditions

Special treatment of boundaries is needed to ensure proper behavior of the resulting simulations.

Enforcing Boundary Conditions No-transfer boundary conditions are easily imposed by setting the fluxes through the boundary triangles to zero. Non-zero flux boundary conditions (i.e, forced fluxes through the boundary as in the case of Fig. 8) are, however, more subtle to handle. First, remark that all these boundary fluxes *must* sum to zero; otherwise, we would have little chance of getting a divergence-free fluid in the domain! As the total divergence is zero, there *must exist* a harmonic velocity field satisfying exactly these conditions, as stated by the Helmholtz-Hodge decomposition theorem with normal boundary conditions [Chorin and Marsden 1979]. Thus, this harmonic part $\mathbf{h}^{\partial,\mathscr{M}}$ can be computed *once and for all* through a Poisson equation using the same setup as described in Section 3.5. This precomputed velocity field allows us to deal very elegantly with these boundary conditions: we simply perform the same algorithm as we described by setting all boundary conditions to zero (with the exception of backtracking which takes the precomputed velocity into account), and *reinject* the harmonic part at the end of each time step (i.e., add $\mathbf{h}^{\partial,\mathscr{M}}$ to the current velocity field).

Viscous Fluids near Boundaries The Voronoi cells at the boundaries are slightly different from the usual, interior ones, since boundary vertices do not have a full 1-ring of tets around them. In the case of NS equations, this has no significant consequence: we set the velocity on the boundary to zero, resulting also in a zero circulation on the dual edges on the boundary. The rest of the algorithm can be used as is.

Inviscid Fluids near Boundaries For Euler equations, however, the tangent velocity at the boundary is not explicitly stored anywhere. Consequently, the boundary Voronoi faces need an additional variable to remedy this lack of information. We store in these dual faces the current integral *vorticity*, bootstrapping it with an initial vorticity imposed by the initial velocity field. From this additional information, we can *deduce* at each time step the missing circulation on the boundary (since the circulation over the inside dual edges is known, and the total integral must sum to the vorticity through Stokes' theorem).

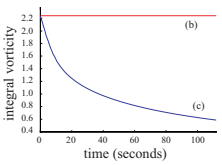
4.8 Handling Arbitrary Topology

Although the problem of arbitrary domain topology (e.g., when its first Betti number is not zero) is rarely discussed in CFD or in our field, it is important nonetheless. We first note that for Euler equations, Kelvin's theorem is also valid for *loops that are not shrinkable to a point* (i.e., loops around a tunnel, or obstacle). Therefore, in the absence of external forces, the circulation along each loop around a tunnel (note: for one period around the obstacle!) is constant in time. So once again, we precalculate a constant harmonic field based on the initial circulation around each tunnel, and simply *add it* to the current velocity field for advection purposes. This procedure serves two purposes: first, notice that we now automatically enforce the discrete equivalent of Kelvin's theorem on *any* (shrinkable or non-shrinkable) loop; second, arbitrary topologies are handled very efficiently.

5 Results and Discussion

We have tested our method on some of the usual "obstacle courses" in CFD. We start with the widely studied example of a flow past a disk (see Fig. 9). Starting with zero vorticity, it is well known that in the case of an inviscid fluid, the flow remains irrotational at all times. By construction, our method does respect this physical behavior since circulation is preserved for Euler equations. We then increase the viscosity of the fluid incrementally, and observe the formation of a vortex wake behind the obstacle, in agreement with physical experiments. As evidenced by the vorticity plots, vortices are shed from the boundary layer formed as a result of the adherence of the fluid to the obstacle, thanks to our proper treatment of the boundary conditions.

The behavior of vortex interactions observed in existing experimental results is now compared to numerical results based on our novel model and those obtained from the semi-Lagrangian advection method. It is known from theory that two like-signed vortices with a finite vorticity core will merge when their distance of separation is smaller than some critical value. This behavior is captured by the experimental data and shown in the first series of snapshots of Fig. 9. As the next row of snapshots indicates, the numerical results that our model generates present striking similarities to the experimental data. In the last row, we see that a traditional semi-Lagrangian advection followed by re-projection misses most of the fine structures of this phenomenon. This can be attributed to the loss of total integral vorticity as evidenced in this inset; in comparison our technique preserves this integral exactly.



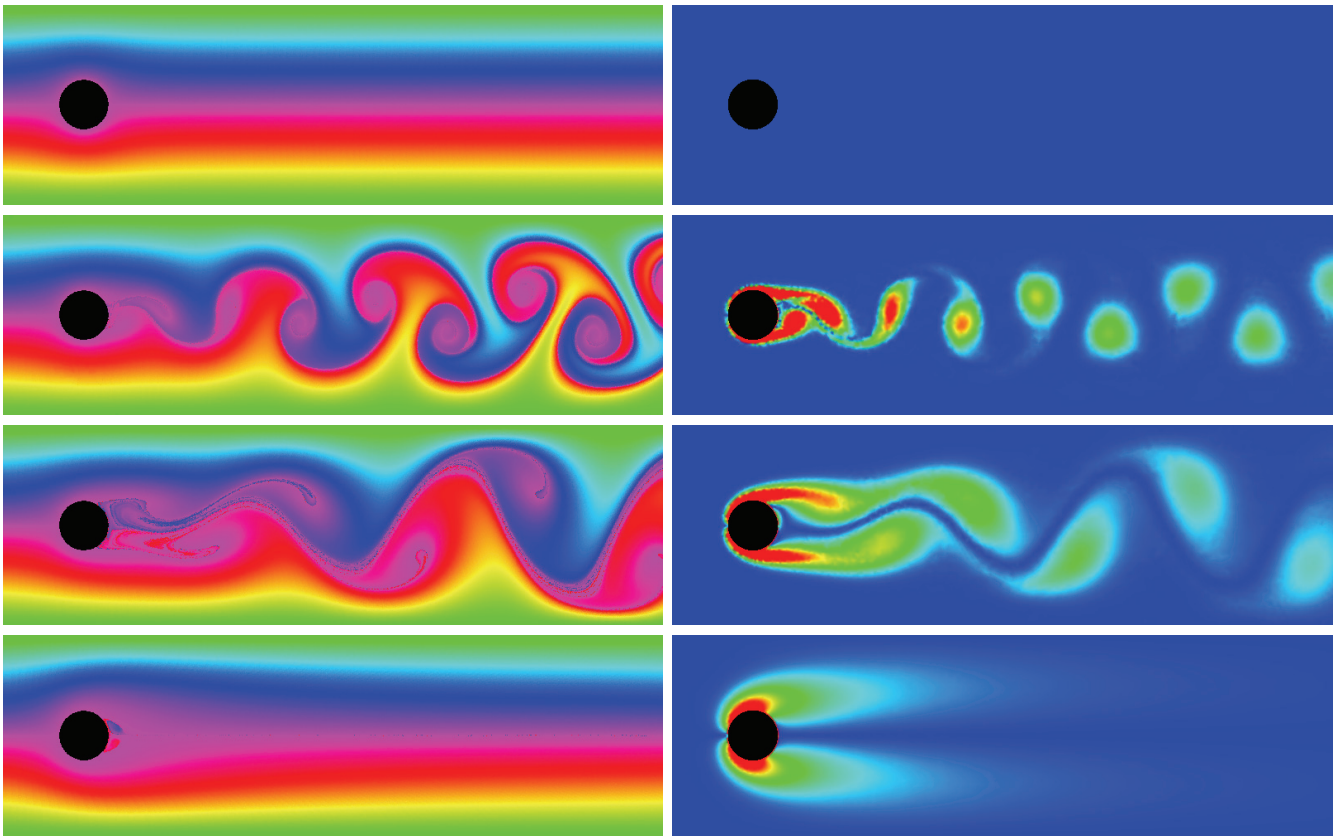


Figure 8: *Obstacle Course: in the usual experiment of a flow passing around a disk, the viscosity as well as the velocity can significantly affect the flow appearance; (left) our simulation results for increasing viscosity and same left boundary flux; (right) the vorticity magnitude (shown in false colors) of the same frame. Notice how the usual irrotational flow is obtained (top) for zero viscosity, while the von Karman vortex street appears as viscosity is introduced.*

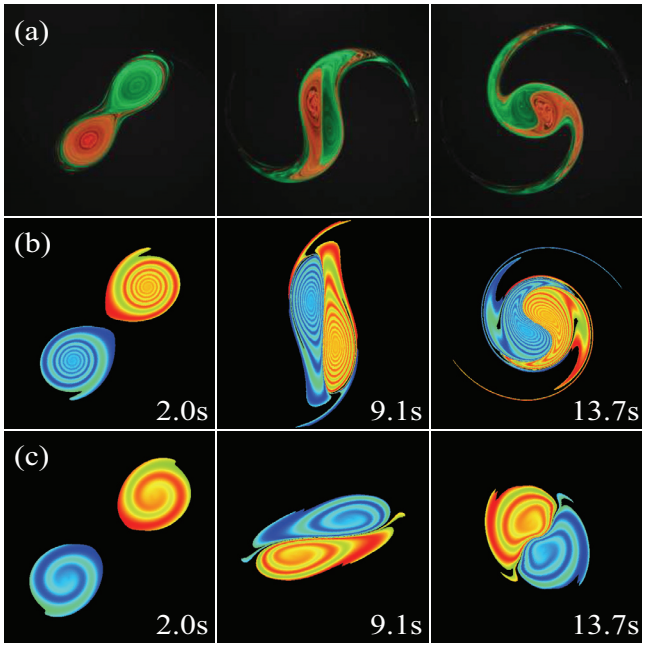


Figure 9: *Two Merging Vortices: discrete fluid simulations are compared with a real life experiment (courtesy of Dr. Trieling, Eindhoven University; see* `http://www.fluid.tue.nl/WDY/vort/index.html`*) where two vortices (colored in red and green) merge slowly due to their interaction (a); while our method faithfully simulates the merging phenomenon (b), a traditional semi-lagrangian scheme does not capture the correct motion because of vorticity damping (c).*

We have also considered the flow on curved surfaces in 3D with complex topology, as depicted in Fig. 10. We were able to easily extend our implementation of two-dimensional flows to this curved case thanks to the intrinsic nature of our approach.

The integration method we proposed can be directly applied to solve three-dimensional fluid flows. We consider a smoke cloud surrounded by air filling the body of a bunny as an example of flow in a domain with complex boundary. The buoyancy drives the air flow which, in turn, advects the smoke cloud in the three-dimensional domain bounded by the bunny mesh as shown in Fig. 1.

In the last simulation, we show a snow globe with a bunny inside in Fig. 6. We emulate the flow due to an initial spin of the globe using a swirl described as a vorticity field. The snow particles are transported by the flow as they fall down under the effect of gravity.

6 Conclusion

In this paper, we have introduced a novel theoretical approach to fluid dynamics, along with its practical implementation and various simulation results. We have carefully discretized the physics of flows to respect the most fundamental geometric structures that characterize their behavior. Amongst the several specific benefits that we demonstrated, the most important is the circulation preservation property of the integration scheme, as evidenced by our numerical examples. The discrete quantities we used are intrinsic, allowing us to go to curved manifolds with no additional complication. Finally, the machinery employed in our approach can be used on any simplicial complex. We wish to emphasize, however, that the same methodology also applies directly to more general spatial partitionings, and in particular, to regular grids or hybrid

meshes [Feldman et al. 2005]—rendering our approach widely applicable to existing fluid simulators.

For future work, a rigorous analysis (beyond the scope of this paper) of the advantages of the current method over some of the standard approaches should be properly investigated.

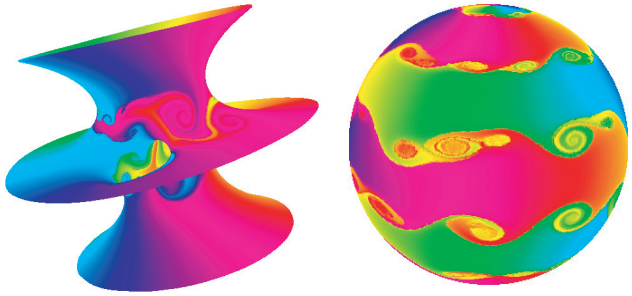


Figure 10: *Weather System on Planet Funky: the intrinsic nature of the variables used in our algorithm makes it amenable to the simulation of flows on arbitrary curved surfaces.*

References

ABRAHAM, R., MARSDEN, J., AND RATIU, T., Eds. 1988. *Manifolds, Tensor Analysis, and Applications*. Applied Mathematical Sciences Vol. 75, Springer.

BOSSAVIT, A., AND KETTUNEN, L. 1999. Yee-like schemes on a tetrahedral mesh. *Int. J. Num. Modelling: Electr. Networks, Dev. and Fields 12* (July), 129–142.

BOSSAVIT, A. 1998. *Computational Electromagnetism*. Academic Press, Boston.

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

FEDKIW, R., STAM, J., AND JENSEN, H. W. 2001. Visual Simulation of Smoke. In *Proceedings of ACM SIGGRAPH*, Computer Graphics Proceedings, Annual Conference Series, 15–22.

FELDMAN, B. E., O'BRIEN, J. F., AND KLINGNER, B. M. 2005. A method for animating viscoelastic fluids. *ACM Transactions on Graphics (SIGGRAPH)* (Aug.).

FETECAU, R. C., MARSDEN, J. E., ORTIZ, M., AND WEST, M. 2003. Nonsmooth Lagrangian Mechanics and Variational Collision Integrators. *SIAM J. Applied Dynamical Systems 2*, 381–416.

FOSTER, N., AND FEDKIW, R. 2001. Practical Animation of Liquids. In *Proceedings of ACM SIGGRAPH*, Computer Graphics Proceedings, Annual Conference Series, 23–30.

FOSTER, N., AND METAXAS, D. 1997. Modeling the Motion of a Hot, Turbulent Gas. In *Proceedings of SIGGRAPH 97*, Computer Graphics Proceedings, Annual Conference Series, 181–188.

GOKTEKIN, T. G., BARGTEIL, A. W., AND O'BRIEN, J. F. 2004. A method for animating viscoelastic fluids. *ACM Transactions on Graphics 23*, 3 (Aug.), 463–468.

HIRANI, A. 2003. *Discrete Exterior Calculus*. PhD thesis, California Institute of Technology.

KANE, C., MARSDEN, J. E., ORTIZ, M., AND WEST, M. 2000. Variational integrators and the Newmark algorithm for conservative and dissipative mechanical systems. *Internat. J. Numer. Methods Engrg. 49*, 1295–1325.

LANGTANGEN, H.-P., MARDAL, K.-A., AND WINTER, R. 2002. Numerical Methods for Incompressible Viscous Flow. *Advances in Water Resources 25*, 8-12 (Aug-Dec), 1125–1146.

LEW, A., MARSDEN, J. E., ORTIZ, M., AND WEST, M. 2003. Asynchronous Variational Integrators. *Arch. Rational Mech. Anal. 167*, 85–146.

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.

MARSDEN, J. E., AND WENSTEIN, A. 1983. Coadjoint orbits, vortices and Clebsch variables for incompressible fluids. *Physica D 7*, 305–323.

MARSDEN, J. E., AND WEST, M. 2001. Discrete Mechanics and Variational Integrators. *Acta Numerica*, 357–515.

MCNAMARA, A., TREUILLE, A., POPOVIC, Z., AND STAM, J. 2004. Fluid Control Using the Adjoint Method. *ACM Transactions on Graphics 23*, 3 (Aug.), 449–456.

MORTON, K. W., AND ROE, P. 2001. Vorticity-Preserving Lax-Wendroff-Type Schemes for the System Wave Equation. *SIAM Journal on Scientific Computing 23*, 1 (July), 170–192.

MUNKRES, J. R. 1984. *Elements of Algebraic Topology*. Addison-Wesley.

PIGHIN, F., COHEN, J. M., AND SHAH, M. 2004. Modeling and Editing Flows Using Advected Radial Basis Functions. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 223–232.

SHAH, M., COHEN, J. M., PATEL, S., LEE, P., AND PIGHIN, F. 2004. Extended Galilean Invariance for Adaptive Fluid Simulation. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 213–221.

SHI, L., AND YU, Y. 2004. Inviscid and Incompressible Fluid Simulation on Triangle Meshes. *Journal of Computer Animation and Virtual Worlds 15*, 3-4 (June), 173–181.

STAM, J. 1999. Stable Fluids. In *Proceedings of ACM SIGGRAPH*, Computer Graphics Proceedings, Annual Conference Series, 121–128.

STAM, J. 2001. A Simple Fluid Solver Based on the FFT. *Journal of Graphics Tools 6*, 2, 43–52.

STAM, J. 2003. Flows on Surfaces of Arbitrary Topology. *ACM Transactions on Graphics 22*, 3 (July), 724–731.

STEINHOFF, J., AND UNDERHILL, D. 1994. Modification of the Euler Equations for *Vorticity Confinement*: Applications to the Computation of Interacting Vortex Rings. *Physics of Fluids 6*, 8 (Aug.), 2738–2744.

TONG, Y., LOMBEYDA, S., HIRANI, A. N., AND DESBRUN, M. 2003. Discrete Multiscale Vector Field Decomposition. *ACM Trans. Graph. 22*, 3, 445–452.

TONG, Y. 2004. *Towards Applied Geometry in Graphics*. PhD thesis, University of Southern California.

TREUILLE, A., MCNAMARA, A., POPOVIĆ, Z., AND STAM, J. 2003. Keyframe Control of Smoke Simulations. *ACM Transactions on Graphics 22*, 3 (July), 716–723.

WARREN, J., SCHAEFER, S., HIRANI, A., AND DESBRUN, M., 2004. Barycentric Coordinates for Convex Sets. Preprint.

YAEGER, L., UPSON, C., AND MYERS, R. 1986. Combining Physical and Visual Simulation - Creation of the Planet Jupiter for the Film 2010. *Computer Graphics (Proceedings of SIGGRAPH 86) 20*, 4, 85–93.

ZHONG, G., AND MARSDEN, J. E. 1988. Lie-Poisson Hamilton-Jacobi Theory and Lie-Poisson Integrators. *Physics Letters A 133*, 3 (Nov.).