Fluid Simulation using Laplacian Eigenfunctions

TYLER DE WITT, CHRISTIAN LESSIG and EUGENE FIUME
University of Toronto

We present an algorithm for the simulation of incompressible fluid phenomena that is computationally efficient and leads to visually convincing simulations with far fewer degrees of freedom than existing approaches. Rather than using an Eulerian grid or Lagrangian elements, we represent vorticity and velocity using a basis of global functions defined over the entire simulation domain. We show that choosing Laplacian eigenfunctions for this basis provides benefits, including correspondence with spatial scales of vorticity and precise energy control at each scale. We perform Galerkin projection of the Navier-Stokes equations to derive a time evolution equation in the space of basis coefficients. Our method admits closed form solutions on simple domains but can also be implemented efficiently on arbitrary meshes.

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

General Terms: Animation, Fluid Simulation

Additional Key Words and Phrases: Laplacian eigenfunctions, physically based animation, fluid, smoke, geometric mechanics

ACM Reference Format:

de Witt, T., Lessig, C., and Fiume, E. 2011. Fluid Simulation using Laplacian Eigenfunctions. ACM Trans. Graph.

1. INTRODUCTION

Fluid motion is naturally captivating. Over the years, it has piqued the imagination and curiosity of artists, mathematicians and scientists. The fascination with fluid motion is exemplified by its long history in the computer graphics literature. Early work focused on obtaining motion that is visually interesting and convincing. More recent physically based techniques rely primarily on numerical approximation of the Navier-Stokes equations. Computer simulation of a model necessitates a finite representation of its spatial quantities. In the past, many approaches for choosing a finite representation have been proposed including the use of Eulerian grids, Lagrangian fluid mass particles, vorticity primitives and model reduction.

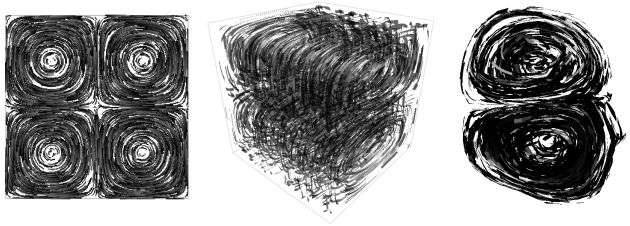

Fig. 1. Examples of Laplacian eigenfunction velocity basis fields on various domains. They are stationary solutions to the Navier-Stokes equations. For simple domains Laplacian eigenfunctions have closed form expressions; for general meshes they are defined through eigendecomposition of a discrete operator.

Grid-based techniques are the most common approach. However, they suffer from high computational complexity, due to the general requirement at each simulation step to solve a system of equations whose size is proportional to the number of grid elements in the domain. Lagrangian techniques, such as mass particles, removes the dependence on the simulation domain. That said, the computation of pressure and other fluid quantities are expensive and approximations lead to noticeable violations of incompressibility. Vorticity primitives, including particles and filaments, are very effective at simulating smoke in inviscid media but have difficulties modelling diffusion and handling boundary conditions. Model reduction is a data-driven approach that exploits a precomputed set of example simulations to obtain a low dimensional representation for fluid motion. While this technique is very efficient at run-time, it suffers from significant costs for precomputation and storage, and is dependent on the performance of an existing simulator or other mechanism to obtain ground-truth data.

We propose an algorithm for the interactive simulation of fluid motion that avoids many of the shortcomings of existing techniques. We employ a representation of fluid velocity and vorticity in a finite dimensional basis of Laplacian eigenfunctions. The resulting velocity basis fields are divergence free and respect boundary conditions, so that these constraints are enforced automatically without the need for additional computation. Our algorithm can be formulated as Galerkin projection of the vorticity form of the Navier-Stokes equations onto Laplacian eigenfunctions defined over the simulation domain. The resulting finite dimensional form of the equations describes the time evolution of the basis coefficients. We precompute the non-linear advection terms between pairs of basis functions and store the result as *structure coefficients* in a set of matrices. Viscosity and external forces are incorporated using linear terms, and the basis function coefficients are hence updated using a simple matrix-vector equation.

Laplacian eigenfunctions form an orthogonal basis, allowing one to easily compute the energy of the fluid. Additionally, Laplacian eigenfunctions of increasing eigenvalue magnitude have a natural visual correspondence with decreasing scales of vorticity. Coupled

Tyler de Witt acknowledges a NSERC-Canada grant.
tyler@dgp.toronto.edu, lessig@dgp.toronto.edu, elf@dgp.toronto.edu

DOI 10.1145/XXXXXXX.YYYYYYY
http://doi.acm.org/10.1145/XXXXXXX.YYYYYYY

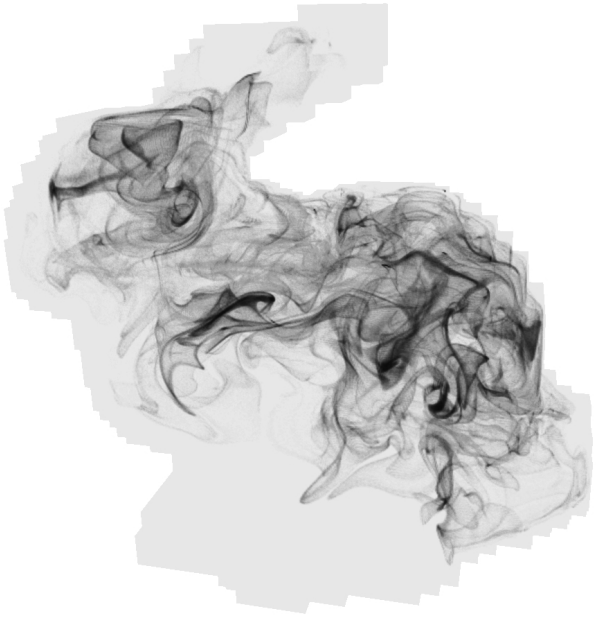

Fig. 2. A buoyant smoke simulation using a basis of 128 Laplacian eigenfunctions.

with orthogonality, the correspondence allows precise control of a fluid's turbulent spectrum through adjustment of basis coefficients. With Laplacian eigenfunctions the viscosity can be simulated accurately through a simple exponential decay of basis coefficients, but also allows arbitrary user-controlled or automatic adjustment of the spectrum to achieve a desired effect.

For some practically important simulation domains such as a 2-D plane and 3-D rectangular cavity, Laplacian eigenfunctions have closed form expressions, allowing fully analytic simulation. In these cases, no mesh is required to store the fluid's velocity. Instead, a velocity can be precisely evaluated at any spatial coordinate without the need for interpolation. Furthermore, closed form expressions allow symbolic evaluation of the precomputed advection operator, making this process fast and exact. However, our method is not limited to these domains, and we present a formulation on structured and irregular meshes using discrete exterior calculus, in which velocity and vorticity basis fields are eigenvectors of a discrete Laplacian operator. Additionally, our method supports the interaction of immersed moving obstacles and buoyancy through projection of forces to the velocity basis fields.

Our method allows considerable flexibility in choosing the basis dimensionality. Even simulations with few degrees of freedom provide visually convincing results, avoiding the artifacts common to very low-dimensional representations in Eulerian or Lagrangian simulations. In this respect, our method provides a principled means of dimensionality reduction of the Navier-Stokes fluid equations. However, our method is *not* data-driven as seen in current model reduction techniques and hence avoids the need for an existing fluid simulator or pre-existing data. We believe our algorithm and choice of basis provides an exciting avenue and will be an important complement to the methods in the literature.

2. RELATED WORK

Incompressible fluid dynamics is a vast subject. We survey some relevant work from geometric mechanics, computational fluid dynamics (CFD) and the computer graphics literature.

2.1 Geometric Mechanics

Euler's equations describing the dynamics of a rotating rigid body date from the 18th century. In 1901, Poincaré [1901] showed that by considering various group manifolds as the configuration space, Euler's equation could apply generally to a class of physical systems. For example, in the case of a rotating rigid body the group is the rotation group SO(3). Arnold [1966] showed that an ideal incompressible fluid is described similarly as geodesic motion on SDiff, the Lie group of volume preserving diffeomorphisms. The notion of *structure coefficients* to describe the interaction of Lie algebra basis elements of these groups is directly related to the precomputed coefficient matrices used in our method. Many of these concepts are summarized by Marsden and Ratiu [1999].

Representing vorticity using Laplacian eigenfunctions dates back at least to Yudovich [1963], who used this method to prove existence and uniqueness theorems for the two dimensional Navier-Stokes equations. More recently, Agrachev et al. [2005] used vorticity Laplacian eigenfunctions to prove theorems in the mathematical control literature. This paper was our inspiration for investigating a Laplacian eigenfunction representation of vorticity as a practical means of fluid simulation in computer graphics applications.

2.2 Computational Fluid Dynamics

In the 1950's, Silberman presented a fluid simulation algorithm for the earth's atmosphere in a basis of spherical harmonics, which are Laplacian eigenfunctions on the surface of a sphere [Silberman 1954]. This basis was applied to the vorticity stream function fluid equations in two dimensions, and the advection operator was evaluated symbolically. This method has come to be known in the CFD literature as the interaction coefficient method. Outside of atmospheric sciences, it is not widely used due to poor scaling for large basis dimensionality. Such performance considerations were the motivation for the development of spectral methods as pioneered by Orszag [Orszag 1969]. Spectral methods are characterized by the use of a fast transform allowing efficient calculation of advection in the spatial domain, thereby avoiding convolution sums in the spectral domain. They are often used to study homogeneous turbulence [Orszag and Patterson 1972; Rogallo et al. 1981]. Fourier series or Chebyshev polynomials are commonly employed, as spectral methods are limited to bases admitting a fast transform.

Our method is most analogous to the interaction coefficient method of [Silberman 1954], although we consider arbitrary domains. On arbitrary domains, Laplacian eigenfunctions do not in general admit a fast transform and hence do not share the inherent theoretical performance of a spectral method. However, Laplacian eigenfunctions have many other benefits as we describe in Section 3. Furthermore, theoretical performance scaling is less critical for the applications we consider and we show that visually detailed simulations are attainable at low cost.

Divergence free finite element methods (DFFEM) employ bases of discrete divergence free velocity fields to solve fluid equations in a space that satisfies mass continuity a priori [Gustafson and Hartman 1983]. Our method is similar in this respect. However, in contrast to DFFEM, for some simple domains Laplacian eigenfunctions do not require a discrete mesh. Also, to our knowledge no basis employed in DFFEM exhibits all of the advantageous proper-

ties of Laplacian eigenfunctions, including orthogonality, stationarity with respect to Navier-Stokes equations, global support, and correspondence with spatial scales of vorticity.

2.3 Computer Graphics

Fluid simulation methods in the computer graphics literature belong to roughly four categories: grid-based, mass particles, vortex elements and model reduction.

Grid-based techniques for simulating the 3-D Navier-Stokes equations were introduced by Foster and Metaxas [1996] but were unstable due to the use of explicit integrators. Stam developed an unconditionally stable integration scheme using semi-Lagrangian advection and an implicit integrator [Stam 1999]. However, the result produces artificial viscosity which dampens vortices prematurely, and requires an iterative linear solver to solve for a pressure field to enforce incompressibility. Works aimed at mitigating or minimizing artificial diffusion include vorticity confinement [Fedkiw et al. 2001] and high order advection schemes [Selle et al. 2008]. To improve the performance of the iterative pressure solver, use of adaptive grids [Losasso et al. 2004] and hierarchical coarse grids for projection [Lentine et al. 2010] have been proposed. Stam [2002] used the 2-D Fourier transform of a velocity field to perform fast pressure projection, but this method is limited to simple domains and boundary conditions, and still dissipates energy. Bridson presented a simple means to generate procedural divergence free flows through the curl of a vector potential stream function [Bridson et al. 2007] but this work did not address physical dynamics. Elcott presented a method that preserves circulation on simplicial meshes, but does not preserve energy [Elcott et al. 2007]. Mullen et al. developed a fluid integrator capable of perfect energy preservation or desired viscosity independent of grid-resolution [Mullen et al. 2009], but this method is complex and requires a solution to a non-linear system at each timestep. Hybrid particle-grid methods such as FLIP [Zhu and Bridson 2005] are effective in eliminating numerical diffusion, but still require a grid to enforce incompressibility. Common to all these stable grid-based techniques previously mentioned is the need to solve a system of equations at each time integration step, the size of which is proportional to the number of grid elements. In contrast, the performance of our method is independent of the domain or grid resolution. In fact, for typical domains such as a 2-D rectangle or 3-D rectangular cavity, the global basis functions we employ have closed form expressions, removing the need for a velocity grid representation entirely. Our method allows controllable viscosity, and supports general domains through a formulation on discrete meshes.

Particle methods track a fluid's mass through Lagrangian elements. Smoothed particle hydrodynamics (SPH) was introduced to graphics by Desbrun and Gascuel [1996] and used subsequently to simulate water [Müller et al. 2003; Adams et al. 2007]. Enforcing incompressibility in SPH methods is computationally expensive, making them impractical for a large number of particles. Our method satisfies incompressibility automatically as it operates directly in a space of divergence free fields.

Vortex methods use Lagrangian elements such as particles or filaments to track vorticity, and advect these elements through the fluid's velocity [Gamito et al. 1995; Park and Kim 2005; Angelidis et al. 2006; Weißmann and Pinkall 2010]. A formulation using vorticity guarantees incompressibility, but the reconstruction of the velocity field is computationally expensive, typically involving the Biot-Savart formula. We also use a vorticity formulation, hence requiring no explicit enforcement of the incompressibility constraint. However, we use a superposition of global basis functions allowing the representation of arbitrary vorticity fields, whereas Lagrangian elements are limited to vorticity concentrated at points or on curves. Additionally, choosing Laplacian eigenfunctions as a basis allows the velocity field to be recovered trivially, removing the need for complicated and expensive reconstruction.

Symbol	Description
N	Basis dimension.
$\{\Phi_k\}$	Set of velocity basis fields.
$\{\phi_k\}$	Set of vorticity basis fields.
$\mathbf{u}$	Fluid velocity vector field.
$\boldsymbol{\omega}$	Fluid vorticity vector field.
$\omega_1, \omega_2, \ldots, \omega_N$	Basis coefficients.
$\mathbf{w} = [\omega_1 \ \omega_2 \ \ldots \ \omega_N]^T$	Column vector of basis coefficients.
Δ	Vector Laplacian operator.
λ_k	Scalar eigenvalue of k^{th} basis field.
$\text{Adv}(\cdot, \cdot)$	Advection operator.
$\{\mathbf{C}_k\}$	Structure coefficient matrices.
$\mathbf{C}_k[i, j]$	$(i, j)^{th}$ entry of the $\mathbf{C}_k$ matrix.
$\mathbf{a_x}, \mathbf{a_y}, \mathbf{a_z}$	Canonical basis for $\mathbb{R}^3$.

Fig. 3. Nomenclature.

Gupta and Narasimhan represented fluid velocity in a basis of Legendre polynomials allowing analytic evaluation of differential operators [Gupta and Narasimhan 2007]. However, only box-boundary conditions were considered and the velocity basis fields are not strictly divergence free hence requiring a pressure projection step to enforce incompressibility.

Model reduction has been applied to fluid simulation by Treuille et al. [2006]. This technique chooses a reduced velocity basis defined on a mesh through observation of an existing fluid simulator. The resulting run-time performance is fast, but the precomputation time and memory requirements are large. Furthermore, it is unclear how well this technique generalizes to arbitrary flows, as behavior is limited to the examples present in training. Our method can be used directly as a means of dimensionality reduction through choice of the basis dimension N, but it differs from current approaches in many respects. We choose an appropriate velocity basis a priori instead of relying on observation of an existing fluid simulator. Up to a desired scale of vorticity, Laplacian eigenfunctions form a complete basis for divergence free fields. Adding basis functions increases the coverage in a well defined way. In contrast, a data driven basis can only approximate flows that are in some sense "close" to those observed in training, and there is no guarantee that additional training data will substantially increase the span of the resulting PCA basis. Our basis has a natural correspondence with spatial scales of vorticity that is lacking in [Treuille et al. 2006]. Finally, Laplacian eigenfunctions have closed form expressions for some simple domains, in which case the precomputation time and memory requirements are vastly reduced in comparison.

3. LAPLACIAN EIGENFUNCTIONS AS BASIS FIELDS

We express the velocity field of a fluid $\mathbf{u}$ on a domain D as

$$\mathbf{u} = \sum_{i}^{N} \omega_i \Phi_i.$$

where Φ_k are eigenfunctions of the *vector Laplacian* $\Delta = \text{grad}\,(\text{div}\,) - \text{curl}^{\,2}$. When acting on divergence free fields, the vector Laplacian reduces to $\Delta = -\text{curl}^{\,2}$. We require the set of basis

Fig. 4. Laplacian eigenfunctions have a correspondence with spatial scales of vorticity, allowing basis coefficients to be interpreted as a discrete spectrum of vorticity. The right of the spectrum corresponds to larger magnitudes of eigenvalues and smaller scale vortices.

fields Φ_k to be divergence free and satisfy a free slip condition at the boundary. Hence our basis fields are completely characterized by

$$\begin{aligned} \Delta\Phi_k &= \lambda_k \Phi_k \\ \text{div}\,(\Phi_k) &= 0 \\ \Phi_k \cdot \mathbf{n} &= 0 \text{ at } \partial D \end{aligned} \tag{1}$$

where λ_k are eigenvalues and $\mathbf{n}$ is a vector normal to the boundary.

The eigenfunctions of the Laplacian operator Δ are domain dependent. For many simple domains, functions satisfying Eq. 1 have closed form expressions, which are available for example in the physics literature where they describe the magnetic fields of electromagnetic resonators [Cheng 1999]. For instance, on a $\pi \times \pi$ square domain, Laplacian eigenfunctions satisfying Eq. 1 have the closed form expressions

$$\begin{aligned} \Phi_k = \frac{1}{k_1^2 + k_2^2} \big(&k_2 \sin(k_1 x)\cos(k_2 y)\mathbf{a_x} \\ &-k_1 \cos(k_1 x)\sin(k_2 y)\mathbf{a_y}\big), \end{aligned} \tag{2}$$

where $k = (k_1, k_2) \in \mathbb{Z}^2$ is a tuple of integers known as the *vector wave number*. The vector fields Φ_k are Laplacian eigenfunctions with eigenvalues $\lambda_k = -(k_1^2 + k_2^2)$. Examples are plotted in Figure 4. We will continue to use the square domain as a concrete, illustrative example throughout the text, although closed form expressions also exist for many other domains including a 3-D rectangular prism [de Witt 2010], a disc, the surface of a sphere, or a planar region with a wrap around boundary condition.

For our simulation method, we also require the vorticity field $\boldsymbol{\omega} = \text{curl}\,(\mathbf{u})$ and a vorticity basis $\{\phi_k\}$ with $\phi_k = \text{curl}\,\Phi_k$. For example, the vorticity basis fields associated with Eq. 2 are just the curl of the velocity basis functions and given by

$$\phi_k = \sin(k_1 x)\sin(k_2 y)\mathbf{a_z}. \tag{3}$$

One can verify that the ϕ_k are also Laplacian eigenfunctions of the domain. However, as $\mathbf{u}$ and $\boldsymbol{\omega}$ are orthogonal, the vorticity basis functions have only a normal component at the boundary, and hence satisfy

$$\begin{aligned} \Delta\phi_k &= \lambda_k \phi_k \\ \phi_k \times \mathbf{n} &= 0 \text{ at } \partial D. \end{aligned} \tag{4}$$

3.1 Basis Field Properties

We summarize some additional interesting and useful properties of our basis. One can verify that the example expressions of Eqs. 2 and 3 satisfy all the properties listed below.

Velocity-Vorticity Duality. In general, reconstructing a velocity field from a vorticity field is computationally expensive, typically involving the use of the Biot-Savart Law [Angelidis et al. 2006; Weißmann and Pinkall 2010]. The key benefit of a representation in Laplacian eigenfunctions is that the inverse operator curl $^{-1}$ applied to vorticity basis functions yields a simple expression:

$$\begin{aligned} \Phi_k &= \text{curl}^{-1}\phi_k \\ &= \text{curl}^{-1}\left(\frac{1}{\lambda_k}\Delta\phi_k\right) \\ &= \frac{1}{\lambda_k}\text{curl}^{-1}\left(-\text{curl}^{2}\phi_k\right) \\ &= -\frac{1}{\lambda_k}\text{curl}\,\phi_k. \end{aligned} \tag{5}$$

A further important observation is that due to linearity of the curl operator, the expansion of the vorticity $\boldsymbol{\omega}$ in the ϕ_i basis shares the same coefficients as the expansion of the velocity $\mathbf{u}$ in the Φ_k basis

$$\boldsymbol{\omega} = \text{curl}\,\mathbf{u} = \text{curl}\sum_i^N \omega_i \Phi_i = \sum_i^N \omega_i \text{curl}\,\Phi_i = \sum_i^N \omega_i \phi_i.$$

This is notable since a single coefficient vector $\mathbf{w} = [\omega_1 \;\; \omega_2 \;\; \ldots \;\; \omega_N]$ uniquely identifies both the fluid's velocity $\mathbf{u}$ and its vorticity $\boldsymbol{\omega}$. Either field can be easily reconstructed from the basis coefficients ω_i.

Orthogonality. Laplacian eigenfunctions on a domain form an orthogonal set. The total energy of a signal expressed in an orthogonal basis is the sum of the squares of its coefficients by Parseval's identity. The fluid's kinetic energy can thus be calculated as

$$\int_D \|\mathbf{u}\|^2 = \sum_i^N \omega_i^2.$$

Spatial scales of vorticity. As shown in Figure 4, larger eigenvalues of the Laplacian correspond to fields with smaller vortices. Basis coefficients can be interpreted as a discrete spatial spectrum of the fluid with higher "frequencies" corresponding to smaller scales of vorticity. This notion has been previously applied by Stam and Fiume using a Fourier basis to generate procedural stochastic turbulence [Stam and Fiume 1993].

A decomposition into a spectrum of vorticity is important for at least two reasons. First, because computations require our basis to be finite, this ordered structure provides a principle by which to select the finite set. In choosing to truncate the spectrum at some finite N, the error we incur is well defined: we lose the ability to simulate vortices smaller than a given scale. Second, combined with orthogonality, our basis delivers a means of controlling the energy at different scales of vorticity by adjusting the magnitude of the basis coefficients. We use this property in Section 4 to accurately model viscous energy decay. It could also be used to initialize or arbitrarily change a fluid's turbulent spectrum.

Closed form expressions. For some simple domains, the basis fields have closed form expressions. This allows the velocity to be evaluated at any spatial coordinate without the need for a voxelized grid or interpolation. A grid may still be used for visualization, for example to track density or subsample the velocity from the closed form expressions to accelerate particle advection. However, this grid is independent of the simulation, and its resolution may be changed without changing the performance or behavior of the underling simulation. Although the benefits of closed form expressions are limited to simple geometries, a 2-D rectangle and 3-D rectangular cavity both represent typical simulation domains. In Section 8 we compute basis fields numerically for general meshed domains through a discrete vector Laplacian operator.

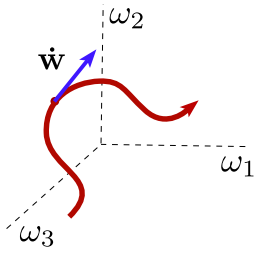

Fig. 5. A fluid's motion is a curve through the Euclidean space of basis coefficients, shown for $N = 3$ as an illustrative example. From the Navier-Stokes equations, we derive an expression for the tangent vector $\dot{\mathbf{w}}$ as a function of the current state vector.

4. DYNAMICS

A fluid's velocity field will change continuously over time according to physical laws. In our basis representation, this can be described by the continuous change of the coefficient vector $\mathbf{w}$. In this section we derive an expression for the time derivative $\dot{\mathbf{w}}$ in terms of the basis coefficients only.

The vorticity formulation of the Navier-Stokes equation is

$$\dot{\boldsymbol{\omega}} = \text{Adv}(\mathbf{u}, \boldsymbol{\omega}) + \nu\Delta\boldsymbol{\omega} + \text{curl}\,(\mathbf{f}) \tag{6}$$

where $\boldsymbol{\omega} = \text{curl}\,\mathbf{u}$ and $\mathbf{f}$ are external forces. For notational convenience, we choose $\text{Adv}(\cdot,\cdot)$ to represent the advection term, which is defined as $\text{Adv}(\mathbf{u}, \boldsymbol{\omega}) := \text{curl}\,(\boldsymbol{\omega} \times \mathbf{u})$.

We perform Galerkin projection of the Navier-Stokes equations onto a Laplacian eigenfunction basis by substituting the expansions $\boldsymbol{\omega} = \sum_i \omega_i\phi_i$, $\mathbf{u} = \sum_j \omega_j\Phi_j$ and $\dot{\boldsymbol{\omega}} = \sum_k \dot{\omega}_k\phi_k$ into Eq. 6 and rearranging terms through linearity of operators

$$\sum_k \dot{\omega}_k\phi_k = \sum_i^N\sum_j^N \omega_i\,\omega_j\text{Adv}(\Phi_i,\phi_j) + \nu\sum_i^N \Delta\omega_i\phi_i + \text{curl}\,(\mathbf{f}).$$

We discuss each right hand term separately.

Advection. The $\text{Adv}(\Phi_i,\phi_j)$ terms represent the nonlinear advection of basis fields. As will be detailed in Section 6, we precompute these terms and the vorticity basis coefficients of the result are stored in a set of matrices $\mathbf{C}_k$. After equating coefficients, the contribution of the self advection term can be written as

$$\dot{\omega}_k = \sum_i^N\sum_j^N \omega_i\,\omega_j\mathbf{C}_k[i,j], \tag{7}$$

summarized in matrix form as

$$\dot{\omega}_k = \mathbf{w}^T\mathbf{C}_k\mathbf{w},$$

where $\mathbf{C}_k$ are precomputed matrices and $\mathbf{w}$ is the column vector of basis coefficients $[\omega_1\ \ \omega_2\ \ \ldots\ \ \omega_N]^T$.

Viscosity. Because ϕ_k are Laplacian eigenfunctions, the viscous term becomes $\nu\sum_i \Delta\omega_k\phi_k = \nu\sum_k \lambda_k\omega_k\phi_k$. The effect of viscosity on each basis coefficient is hence described by the linear first order differential equation

$$\dot{\omega}_k = \nu\lambda_k\omega_k$$

which conveniently has the closed form solution $\dot{\omega}_k(t) = \omega_k(0)e^{\nu\lambda_k t}$. This says that the magnitude of each basis coefficient decays with a time constant equal to the eigenvalue, which is physically correct, as small vortices dissipate faster than large vortices.

External forces. External forces can be incorporated by projecting $\text{curl}\,(\mathbf{f})$ on the vorticity basis, to obtain coefficients $f_i = \langle\text{curl}\,(\mathbf{f}),\phi_i\rangle$ satisfying $\text{curl}\,(\mathbf{f}) = \sum_i^N f_i\phi_i$. Due to Eq. 5, f_i can be equivalently obtained by directly projecting $\mathbf{f}$ to the velocity basis basis $f_i = \langle\mathbf{f},\Phi_i\rangle$, which often involves less computation. The contribution to $\dot{\omega}_k$ is then

$$\dot{\omega}_k = f_k.$$

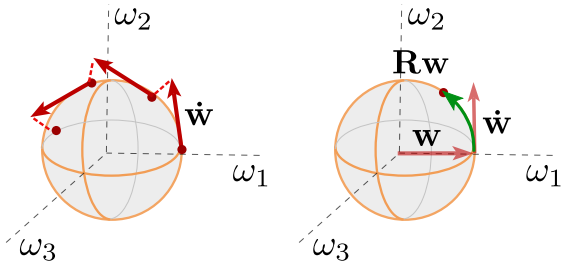

Fig. 6. A basis of Laplacian eigenfunctions is orthogonal, hence surfaces of constant energy are spheres in coefficient space. Left: Unconstrained explicit timesteps are subsequently projected to the manifold of constant energy by normalizing the position vector. Right: Although more expensive, it is also possible to derive an N-dimensional rotation matrix that constrains integration exactly to the state manifold.

Time Evolution Equation. In total, the time derivative of each basis coefficient is

$$\dot{\omega}_k = \mathbf{w}^T\mathbf{C}_k\mathbf{w} + \nu\lambda_k\omega_k + f_k. \tag{8}$$

5. TIME INTEGRATION

Any standard numerical integration scheme can be applied to integrate Eq. 8 forward in time. However, for computer graphics applications speed and energy stability are important requirements. We first describe our preferred integration scheme that meets these two requirements, and then discuss other available techniques.

Our basis is orthogonal allowing kinetic energy to be calculated as a sum of squared coefficients. Additionally, orthogonality implies that surfaces of constant energy in the Euclidean space of coefficients are spheres. An inviscid fluid preserves kinetic energy, and should trace out a path on such a sphere. We choose a fast explicit integrator (such as forward Euler or Runge-Kutta method) to first perform an unconstrained timestep, followed by renormalization to enforce the energy constraint as depicted in Figure 6. Renormalizing to preserve the kinetic energy is a technique available in any fluid simulation method and is not particular to our approach. However, when employing grid based velocity fields it is often undesirable as it can lead to visual artifacts. We have not observed such artifacts, possibly because our basis fields are globally supported and energy is never dissipated locally through a pressure projection step as for example in [Stam 1999].

The effect of viscosity and projected forces will change the kinetic energy, so these terms are integrated following the energy renormalization. Physical viscosity is achieved by decaying each coefficient exponentially as described in Section 4. Our integration scheme is summarized in the pseudo-code of Alg. 1.

Run time complexity. Computation is dominated by the evaluation of matrix vector products, making the run time complexity $O(z)$, where z is the total number of non-zero entries in all the $\{\mathbf{C}_k\}$ combined. In general, $\{\mathbf{C}_k\}$ are dense and z is $O(N^3)$, leading to a computational complexity similar to that of [Treuille et al. 2006]. However, for some domains where closed form expressions are available including a 2-D rectangle and 3-D cavity, the regular-

$e_1 = \sum_i^N \mathbf{w}[i]^2$ // Store kinetic energy of velocity field
for k = *1 to* N **do**
  $\dot{\mathbf{w}}[k] = \mathbf{w}^T \mathbf{C}_k \mathbf{w}$ // Matrix vector product
$\mathbf{w}$ += $\dot{\mathbf{w}} \Delta t$ // Explicit Euler integration
$e_2 = \sum_i^N \mathbf{w}[i]^2$ // Calculate energy after time step
$\mathbf{w}$ *= $\sqrt{e_1/e_2}$ // Renormalize energy
for k = *1 to* N **do**
  $\mathbf{w}[k]$ *= $\exp(\lambda_k \Delta t)$ // Dissipate energy for viscosity
  $\mathbf{w}[k]$ += $\mathbf{f}[k]$ // External forces

Algorithm 1: *Pseudo-code for our fluid simulator. Time integration is explicit and does not require the solution to a linear system.*

ity of the boundary leads to very sparse $\{\mathbf{C}_k\}$ matrices making the theoretical complexity $O(N^2)$ and effectively $\approx O(N)$ for practical ranges of N.

Additional Integration Schemes. Eq. 8 is a symbolic expression for the first time derivative of vorticity. Differentiating this expression produces closed form expressions for time derivatives of arbitrary order. These can be useful for alternate integration schemes to improve accuracy or allow time reversibility. However, considering that stability has already been enforced it may not be a concern for graphics applications. Greater accuracy could also be easily achieved through high order explicit schemes using a small timestep.

A final integration scheme that is theoretically interesting involves the calculation of an N dimensional rotation matrix $\mathbf{R}$, which, when applied to the coefficient vector $\mathbf{w}$, constrains its motion exactly to the constant energy N-sphere. This approximates the true geodesic motion of the Euler fluid equations near the current state. The position vector $\boldsymbol{\omega}$ and the tangent vector $\dot{\boldsymbol{\omega}}$ span an $N-1$ dimensional rotation plane that uniquely identifies an $N \times N$ skew symmetric matrix ξ. This matrix $\mathfrak{g}$ is an element of $\mathfrak{so}(N)$, the Lie algebra of the N-dimensional rotation group SO(N). Multiplying by Δt and exponentiating the matrix yields the $N \times N$ rotation matrix

$$\mathbf{R} = \exp(\Delta t \xi).$$

This method is more expensive than explicit integration with renormalization, and we have found that in comparison it offers very little gain in accuracy for small timesteps. However, it is of interest because it preserves the geometric viewpoint of a fluid as a high dimensional rotation group, and provides a more rigorous way of enforcing energy preservation compared to the renormalization correction step.

6. PRECOMPUTATION OF ADVECTION OPERATOR

The operator $\text{Adv}(\mathbf{u}, \boldsymbol{\omega}) := \text{curl}\,(\boldsymbol{\omega} \times \mathbf{u})$ represents the advection of a fluid's vorticity by its velocity field. It has many equivalent expressions, including the the Lie derivative $\mathcal{L}_{\mathbf{u}}\boldsymbol{\omega}$, or the Jacobi-Lie bracket of vector fields $-[\mathbf{u}, \boldsymbol{\omega}]$,

$$\text{Adv}(\Phi_i, \phi_j) := \mathcal{L}_{\Phi_i}\phi_j = -[\Phi_i, \phi_j] = \text{curl}\,(\phi_j \times \Phi_i).$$

In our context, all the preceding expressions are equivalent, and any can be used to evaluate the advection of pairs of basis fields. For domains admitting closed form expressions for Laplacian eigenfunctions, the evaluation can be performed symbolically and is hence exact. For discrete domains, it can be approximated numerically on a mesh as described in Section 8.

for i, j = *1 to* N **do**
  $p = \text{Adv}(\phi_i, \Phi_j)$ // Project the result onto finite basis
  for k = *1 to* N **do**
    $\mathbf{C}_k[i,j] = \text{Proj}(p, \phi_k)$

Algorithm 2: *Pseudocode for precomputing entries of* $\mathbf{C}_k$ *matrices.*

For every pair of basis functions we evaluate the advection operator and express the result in the finite ϕ_k basis. The basis coefficients of this projection are the *structure coefficients* that form the $\{\mathbf{C}_k\}$ matrices and satisfy

$$\text{Adv}(\Phi_i, \phi_j) = \sum_k \mathbf{C}_k[i,j]\phi_k.$$

The Laplacian eigenfunction basis is closed under the Jacobi-Lie bracket. Hence, we expect the result to factor perfectly into a linear combination of vorticity basis functions.

For simulation, our basis must necessarily be finite dimensional. Despite closure, the advection operator may produce coefficients beyond the chosen finite bandlimit N which cannot be stored. This is unavoidable, as the nonlinear advection operator necessitates products of functions. Considering for example the Fourier basis, the multiplication of two N bandlimited functions is in general bandlimited by $2N$. Physically this represents the cascading of energy to ever higher scales of turbulence.

Projecting the result of the advection operator to our finite dimensional basis amounts to truncating the coefficients beyond the bandlimit N. However, this truncation is physically motivated, since in a real fluid the vortices will eventually reach a small enough scale and dissipate quickly through viscosity. A pseudo-code listing of the precomputation procedure is shown in Algorithm 2.

Properties of $\mathbf{C}_k$ matrices. Because the Jacobi-Lie bracket and vector cross product are anti-symmetric operators, the structure coefficient matrices have the property

$$\frac{1}{\lambda_i}\mathbf{C}_k[i,j] = -\frac{1}{\lambda_j}\mathbf{C}_k[j,i].$$

The antisymmetry reflects an important property of the our basis functions. The self advection $\text{Adv}(\Phi_k, \phi_k)$ of a vorticity basis field ϕ_k by its velocity Φ_k is identically zero, and hence $\dot{\mathbf{u}} = 0$, meaning that each velocity basis field is a stationary flow. This is analogous to the stable rotation of a rigid body about a principal axis [Arnold 1966]. To illustrate the preceding discussion, the evaluation of the structure coefficients in closed form for a 2-D rectangle is provided in Appendix A as an example.

7. EXTERNAL FORCES

As discussed in Section 4, external forces can be incorporated by projecting $\mathbf{f}$ to the velocity basis basis $f_i = \langle \mathbf{f}, \Phi_i \rangle$. The inner product for vector valued $\mathbf{f}$ and Φ_i is defined by the summation of dot product of vectors at every point x within the domain

$$\langle \mathbf{f}, \Phi_i \rangle = \sum_x \mathbf{f}(x) \cdot \Phi_i(x).$$

We make use of external forces to allow immersed moving obstacles and to incorporate a simple buoyancy model.

7.1 Moving Obstacles

The eigenfunctions of the Laplacian operator are defined by their domain and boundary conditions, making the velocity and vorticity basis fields domain dependent. Static irregular boundaries and obstacles are supported in our method through precomputation on general meshes as will be discussed in Section 8. However, moving obstacles change the shape and boundary conditions of the domain dynamically, and hence require special consideration.

Our goal is to satisfy the internal boundary conditions of immersed objects at all times. This requirement can be simply stated: in addition to remaining divergence free, the fluid velocity at an object's boundary should be equal to the normal component of the boundary's velocity. This satisfies the free-slip condition when the object is at rest, and equates normal components of the fluid and boundary velocity when the object is in motion.

Our solution is as follows. At each time-step we project the difference from the desired normal component onto the velocity basis Φ_k and subtract the result from the current state vector $\mathbf{w}$. The result is a divergence free field that best satisfies the desired boundary conditions. Note that this method is not perfect, as the projected forces only approximate the desired forces to the extent that the basis fields can resolve them. In other words, to handle obstacles with small spatial features, one must increase N to use basis functions of a sufficiently high spatial frequency. However, for coarse objects, we have found this method to provide reasonable accuracy, and it is efficient enough to perform interactively without requiring precomputation.

Treuille et al. [2006] also correct the normal velocity component through projection to a divergence free field, and our technique is similar in this respect. However, in their case an additional set of fields they name the *boundary basis* are employed that are chosen based on the object's geometry to best correct for normal velocity components. The boundary basis allows the free-slip constraint to be more accurately enforced in the vicinity of the boundary, but adds substantially to memory and precomputation expense. It also does nothing to improve the quality of object-fluid interaction since the underlying simulation basis, to which the boundary basis must be numerically projected, remains unchanged. In contrast, our basis fields exhibit a spectrum of spatial scales (akin to a Fourier Series) allowing some guarantee of resolving obstacle features with similar length scales. Although our method does not perfectly resolve the boundary, it avoids the use of multiple bases for simulation and boundaries as well as the associated expensive precomputation and memory requirements.

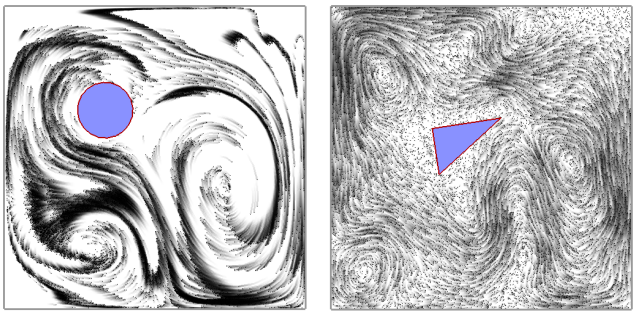

Fig. 7. Irregular obstacles interact with the fluid simulation by projecting contact forces to the velocity basis to best satisfy boundary conditions.

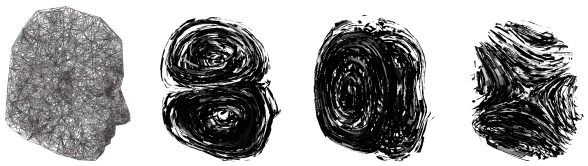

Fig. 8. Velocity basis fields for a tetrahedral mesh obtained through eigendecomposition of a discrete Laplacian operator.

7.2 Buoyancy

In some of our examples we incorporate a simple buoyancy model. Smoke density or particle density are subsampled onto a grid. Buoyancy forces at each grid centre are calculated through the Boussinesq approximation. These forces are projected to the velocity basis through pointwise multiplication.

8. FORMULATION ON MESHED DOMAINS

Simple geometries admit basis fields with closed form expressions. However, our method also supports discrete domains defined on a mesh. For this, we require a set of basis fields defined on the mesh that are eigenfunctions of a discrete Laplacian operator, as well as a means to precompute their advection numerically. Discrete exterior calculus (DEC) provides a principled means of describing operators and quantities on simplicial meshes [Desbrun et al. 2005]. It has been applied to fluid simulation in previous work, and we use a discrete formulation on tetrahedral meshes analogous to [Mullen et al. 2009; Elcott et al. 2007]. Regular voxel meshes are also supported as a special case of this discretization.

Discrete Basis Fields. Through DEC we define the discrete Laplacian operator $\Delta = -\text{curl}^{\,2} = -d * d*$ which has a representation as a sparse, symmetric matrix. We compute the eigendecomposition of this matrix to produce the discrete velocity and vorticity basis fields. The velocity basis fields satisfy a free slip boundary condition and are divergence free, due to constraints imposed implicitly through the Laplacian operator matrix. For example, to enforce a free-slip velocity boundary condition, we omit (set to zero) the rows of the discrete Laplacian Δ that calculate velocity flux on boundary faces. Defined as above, Δ admits only divergence free solutions in its eigendecomposition. Hence the fields produced by the eigensolver satisfy the conditions of Eq. 1 in a discrete setting. Examples of basis fields for a tetrahedral mesh are shown in Figure 9.

Discrete Advection Operator. We also employ DEC to approximate the advection operator $\text{Adv}(\cdot,\cdot)$ using appropriate discretizations. This evaluation is similar to that employed in [Mullen et al. 2009].

Other than the discrete representation and computations described above, the rest of our fluid simulation method remains the same. The operation and performance of the time integration scheme described in Section 5 does not change, since it operates only with the basis coefficients. However, additional expenses in the case of meshed domains include the storage of discrete basis fields, and the reconstruction of the velocity field through summation. As we show in Section 9, these costs are reasonable for typical operating parameters, but can become large for simulations employing very fine meshes and large basis dimensionality.

					Memory		Runtime			
Domain Type	Domain	Basis Dim.	Mesh Ele.	Precomp. (s)	$\mathbf{C}_k$ (Mb)	Basis Fields † (Mb)	Advection (ms)	Vel. Reconst (cached)† (ms)	Vel. Reconst (closed form) (ms) ††	External Forces ††† (ms)
		N	T	$O(N^2)$	$O(N^2)$	$O(TN)$	$\approx O(N)$	$O(TN)$	$O(TN)$	$O(TN)$
Closed form	3-D Cube	81		3.6	0.05	66	18	26		
	3-D Cube	172		15	0.3	141	37	56		
	3-D Cube	325		55	1.1	267	70	105		
	3-D Cube	540		149	3.3	443	125	172		
Closed form	3-D Cube	81		3.6	0.05	n/a	18		30	
	3-D Cube	172		15	0.3		37		67	
	3-D Cube	325		55	1.1		70		126	
	3-D Cube	540		149	3.3		125		216	
		N	T	$O(TN^2)$	$O(N^3)$	$O(TN)$	$O(N^3)$	$O(TN)$		$O(TN)$
Tets	Head	64	$\approx 24^3$	436	0.9	26	20	10		
	Head	128	$\approx 24^3$	1689	7.2	52	42	20		
Voxels	Armadillo	16	32^3	18	0.01	13	3.5	5		22
	Armadillo	32	32^3	47	0.1	26	7	10		43
	Armadillo	64	32^3	185	0.9	52	15	20		88
	Armadillo	128	32^3	1109	7.8	105	39	41		172
Voxels	Bunny	32	32^3	79	0.11	26	8	10		22
	Bunny	64	32^3	269	0.95	52	17	19		43
	Bunny	128	32^3	1305	7.8	105	40	39		90
	Bunny	256	32^3	6911	62.4	210	130	87		174

Fig. 10. Precomputation time, storage requirements and runtime performance performed on a single CPU core. N is the basis dimensionality and T is the number of mesh elements.
† Closed form velocity bases cached on a 32^3 grid.
†† Per 1000 closed form evaluations.
††† Buoyancy forces calculated on a 16^3 subsampled grid.

9. RESULTS

The storage requirements, precomputation time, and runtime performance of our simulation method for both closed form and meshed domains are presented in Figure 10. All experiments were performed on a single CPU core. Time integration was performed using an explicit fourth order Runge Kutta method. Closed form domains are limited in their boundaries, but have notable advantages in terms of runtimes, precomputation and memory requirements. For examples including external forces (such as buoyancy or moving obstacles), the cost of projecting the forces on to the basis is noted. This cost is proportional to the mesh resolution and the number of basis fields. In the case of the bunny, a subsampled 16^3 density grid is used for the buoyancy force calculations.

Velocity Reconstruction. For discrete meshes, velocity field reconstruction requires summation of stored basis fields. This is proportional to the mesh resolution and the dimension of the basis. On closed form domains, there are two alternatives for velocity reconstruction. The basis fields may be pre-evaluated on a mesh and stored, just as in the discrete case. Alternatively, they may be computed on demand. Closed form evaluation is proportional to the number of basis functions and the number of advected quantities. Each alternative has its strengths. Caching the basis fields uses memory, but saves computation when many quantities are being advected through the field (density or millions of particles). If only a few particles need to be advected (leaves in wind, for example), then evaluating closed form expressions is accurate and fast and does not have additional memory requirements. A column in Table 10 lists the cost of 1000 closed form evaluations.

The supplemental videos demonstrate simulations performed on a variety of domains with varying basis dimensionality. A comparison to the stable fluids algorithm is included as a rough qualitative validation. We demonstrate flow on some simple tetrahedral meshes; however we chose a structured voxelized grid for the bunny example only to facilitate implementation. A robust tetrahedral mesh implementation would have similar performance characteristics and alleviate the boundary "stair case" artifacts. The effects of basis dimensionality are illustrated through the bunny example. Modes with small eigenvalue capture the low frequency motion of the fluid. Notably, the bunny's ears do not begin to be resolved until after the 64th mode. In addition to using the bottom of the spectrum to capture the large scale motion, one may choose additional modes from much higher parts of the spectrum to incorporate smaller scales. This demonstrates the benefit of a basis that exhibits a spectrum of scales. Note that these high frequency modes interact and decay physically, in contrast to other post-processing turbulence models.

10. CURRENT LIMITATIONS

Our method is most applicable to gaseous phenomena and situations when the domain is entirely filled by fluid. Currently it is not readily adaptable to typical liquid simulations that require a constantly changing fluid domain with a free surface.

We have shown that interesting dynamics can be captured in a reasonably sized basis dimension and simulated interactively. However, various issues prevent it from scaling well to very large basis dimension or grid resolutions. For irregular domains, the runtime is in general $O(N^3)$. Large mesh resolutions also require large

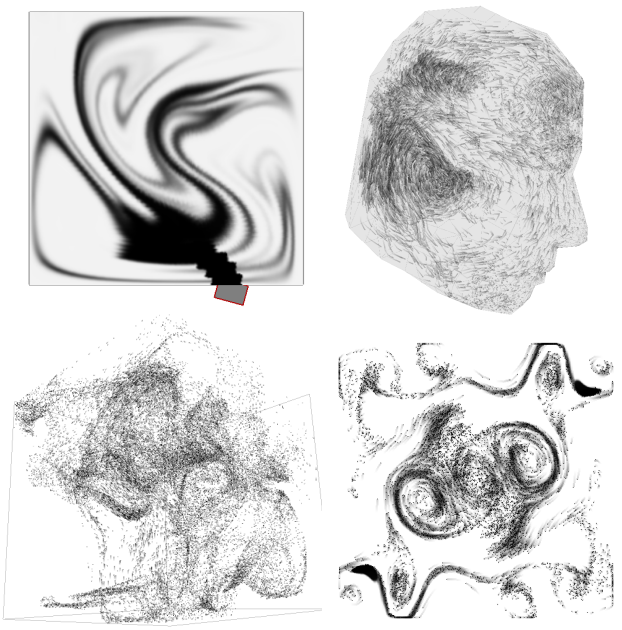

Fig. 9. Top-left: An interactive painting application benefits from an accurate viscous fluid simulation that is independent of the density grid. Top-right: Fluid simulation within a tetrahedral model of a head. Bottom-left: A turbulent 3-D flow in a cubic domain. Bottom-right: Initializing an inviscid simulation with basis functions that are spatially symmetric produces a perpetual symmetric flow.

precomputation times and storage for the basis fields. For discrete meshes, the cost of reconstructing the velocity field and projecting external forces grows linearly with the basis dimension and mesh resolution.

Many of these issues are not present for domains with closed form expressions. However, in this case the shape of the boundary is limited. Also, when advecting many particles or projecting many forces the velocity basis fields must still be cached as the cost of closed form evaluations become prohibitive.

11. FUTURE WORK

Artistic Control. We believe our method has potential to be exploited for the expressive control of fluid phenomena. We have shown how to continuously change the basis coefficients to simulate the physical motion of a fluid. However, any smooth curve through coefficient space, physical or not, may be perceived as "fluid-like" as it represents a continuously changing volume preserving flow that respects all boundary conditions. In addition to constructing completely arbitrary flows, perturbing existing physical paths offers a means to deviate from physics while quantifying this deviation. Due to orthogonality of the basis and its correspondence with vorticity of varying scales, we have a unique mechanism for spectral energy control. This could be used to implement time-varying filters to amplify or attenuate parts of the spectrum, such as achieving crescendos of turbulence or gradual calming. Again, we have a means of quantifying the deviation from non-physical energy behavior, as we have shown how to decay the spectrum according to physical viscosity.

Space-time control for fluids has been attempted previously in [Treuille et al. 2003; McNamara et al. 2004; Fattal and Lischinski 2004]. Many of these methods can be expensive because the optimization scales sharply with the size of the grid, making them impractical for interesting domains. A low dimensional basis offers a good setting to implement control policies that would be intractable in higher dimensions as demonstrated for example by Barbič et al. [Barbič et al. 2009]. Our method's availability of closed form expressions for time derivatives could also prove useful in optimization algorithms.

Our method is fast enough to be interactive, and is very memory efficient and well formulated on rectangular domains due to the available closed form expressions. This makes it particularly attractive for use in image based settings such as painting applications that simulate fluid phenomena, as we briefly demonstrate in the video. Additional potential uses in this vein include texture synthesis and non-photorealistic rendering.

Improvements to our Method. Boundaries of moving obstacles are handled only approximately and could benefit from alternate methods. We have presented a fast and stable integration scheme; however, additional time integrators could be explored, particularly symmetric integrators to allow time reversibility as was achieved in [Mullen et al. 2009]. Time reversibility could prove useful in fluid control applications, as was demonstrated for rigid bodies by Twigg and James [2008]. We have evaluated the advection operator symbolically for closed form expressions on rectangular 2-D and 3-D domains. The same could be done for additional geometries, such as a 2-D disk or a spherical surface. Also, different boundary conditions (for example, a wrap-around boundary condition) remain to be considered, which could prove useful for tilings of fluid simulation domains [Wicke et al. 2009].

Other Applications. Divergence free fields have many potential uses besides simulating natural phenomena. Fluid motion describes the optimal transport in an incompressible medium, and can be used to quantify volume preserving deformations. This has uses in image analysis and shape deformation. We plan to consider how the unique properties of our method could be exploited in these fields. In particular, the elegant formulation on rectangular domains could make it useful for medical image registration. Additionally, the availability of closed form expressions and flexibility in choosing the basis dimension make it an accurate and tractable model for optimization methods. This could be useful for the inverse modelling of real fluid flows for the purpose of parameter estimation, for example to estimate viscosity from sampled velocity measurements.

12. CONCLUSION

We have presented a fluid simulation method that uses eigenfunctions of the vector Laplacian as bases. We have described many of its unique properties and its use as a practical means of fluid simulation for computer graphics. The orthogonality of the basis functions and their correspondence to a spectrum of vorticity scales enables energy control at varying turbulent scales. We have used this property to enforce stability of integrators and simulate physical viscosity. Flexibility in choosing basis dimensionality and the ability to integrate directly in a space of basis coefficients permits computational efficiency, enabling interactive performance. The existence of closed form solutions for simple domains allows symbolic evaluation of the advection operator and the ability to sparsely evaluate velocities on demand.

We have demonstrated some of the useful properties of our method, but many exciting avenues remain to be explored. We plan to investigate its use for the the expressive control of fluid motion, such as spectral energy control and space time optimization. We

also believe there is potential for our method to be exploited in other research areas such as medical imaging and inverse flow modelling.

APPENDIX

A. EVALUATING ADVECTION IN CLOSED FORM FOR 2-D DOMAIN

We evaluate the advection operator for pairs of basis functions defined on a 2-D rectangular domain, defined in Eqs. 3 and 2. We evaluate $\mathrm{Adv}(\Phi_i, \phi_j) = \mathrm{curl}\,(\phi_j \times \Phi_i)$ recalling that i, j are vector wave numbers $i = (i_1, i_2), j = (j_1, j_2)$ and the eigenvalues $\lambda_i = -(i_1^2 + i_2^2)$. This simplifies to

$$\mathrm{Adv}(\Phi_i, \phi_j) = \left(\frac{1}{\lambda_i} i_1 j_2 \cos(i_1 x)\cos(j_2 y)\sin(j_1 x)\sin(i_2 y) - \frac{1}{\lambda_i} i_2 j_1 \cos(j_1 x)\cos(i_2 y)\sin(i_1 x)\sin(j_2 y) \right) \mathbf{a_z}.$$

The trigonometric identity $\cos(\alpha)\sin(\beta) = \frac{1}{2}\sin(\alpha+\beta) - \frac{1}{2}\sin(\alpha-\beta)$ allows factoring to a suitable expression which is indeed spanned by $\{\phi_k\}$:

$$\begin{aligned}\mathrm{Adv}(\Phi_i, \phi_j) = \frac{1}{4\lambda_i} \big(&(i_1 j_2 - i_2 j_1)\phi_{i_1+j_1, i_2+j_2} \\ &- (i_1 j_2 + i_2 j_1)\phi_{i_1+j_1, i_2-j_2} \\ &+ (i_1 j_2 + i_2 j_1)\phi_{i_1-j_1, i_2+j_2} \\ &-(i_1 j_2 - i_2 j_1)\phi_{i_1-j_1, i_2-j_2} \big).\end{aligned}$$

The resulting coefficients are stored in the $\{\mathbf{C}_k\}$ matrices

$$\begin{aligned}\mathbf{C}_{i_1+j_1, i_2+j_2}[i,j] &= -\frac{1}{4(i_1^2+i_2^2)}(i_1 j_2 - i_2 j_1) \\ \mathbf{C}_{i_1+j_1, i_2-j_2}[i,j] &= \frac{1}{4(i_1^2+i_2^2)}(i_1 j_2 + i_2 j_1) \\ \mathbf{C}_{i_1-j_1, i_2+j_2}[i,j] &= -\frac{1}{4(i_1^2+i_2^2)}(i_1 j_2 + i_2 j_1) \\ \mathbf{C}_{i_1-j_1, i_2-j_2}[i,j] &= \frac{1}{4(i_1^2-i_2^2)}(i_1 j_2 - i_2 j_1).\end{aligned}$$

This result demonstrates closure of the advection operator. The indices i, j are meant figuratively, as they represent tuples of integers. A suitable re-mapping from (i_1, i_2) and (j_1, j_2) to positive integers is necessary in an implementation. When outside of the storable finite range, they are discarded as described previously. Note the sums of indices $i_1 + j_1$ and $i_2 + j_2$, which reflect the doubling in bandlimit due to multiplication of sinusoidal functions.

REFERENCES

ADAMS, B., PAULY, M., KEISER, R., AND GUIBAS, L. J. 2007. Adaptively sampled particle fluids. In *ACM SIGGRAPH 2007 papers*. SIGGRAPH '07. ACM, New York, NY, USA.

AGRACHEV, A. A. AND SARYCHEV, A. V. 2005. Navier-Stokes Equations: Controllability by Means of Low Modes Forcing. *Journal of Mathematical Fluid Mechanics 7,* 1 (March), 108–152.

ANGELIDIS, A., NEYRET, F., SINGH, K., AND NOWROUZEZAHRAI, D. 2006. A controllable, fast and stable basis for vortex based smoke simulation. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*. SCA '06. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 25–32.

ARNOLD, V. I. 1966. Sur la géométrie différentielle des groupes de lie de dimension infinie et ses applications à l'hydrodynamique des fluides parfaits. *Ann. Inst. Fourier (Grenoble) 16*.

BARBIČ, J., DA SILVA, M., AND POPOVIĆ, J. 2009. Deformable object animation using reduced optimal control. In *ACM SIGGRAPH 2009 papers*. SIGGRAPH '09. ACM, New York, NY, USA, 53:1–53:9.

BRIDSON, R., HOURIHAM, J., AND NORDENSTAM, M. 2007. Curl-noise for procedural fluid flow. In *ACM SIGGRAPH 2007 papers*. SIGGRAPH '07. ACM, New York, NY, USA.

CHENG, D. K. 1999. *Field and Wave Electromagnetics*. Addison-Wesley, Reading, Mass.

DE WITT, T. 2010. Fluid Simulation in Bases of Laplacian Eigenfunctions. M.S. thesis, University of Toronto, Toronto, ON, Canada.

DESBRUN, M., KANSO, E., AND TONG, Y. 2005. Discrete differential forms for computational modeling. In *ACM SIGGRAPH 2005 Courses*. SIGGRAPH '05. ACM, New York, NY, USA.

DESBRUN, M. AND PAULE GASCUEL, M. 1996. Smoothed particles: A new paradigm for animating highly deformable bodies. In *In Computer Animation and Simulation 96 (Proceedings of EG Workshop on Animation and Simulation*. Springer-Verlag, 61–76.

ELCOTT, S., TONG, Y., KANSO, E., SCHRÖDER, P., AND DESBRUN, M. 2007. Stable, circulation-preserving, simplicial fluids. *ACM Trans. Graph. 26*.

FATTAL, R. AND LISCHINSKI, D. 2004. Target-driven smoke animation. In *ACM SIGGRAPH 2004 Papers*. SIGGRAPH '04. ACM, New York, NY, USA, 441–448.

FEDKIW, R., STAM, J., AND JENSEN, H. W. 2001. Visual simulation of smoke. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. SIGGRAPH '01. ACM, New York, NY, USA, 15–22.

FOSTER, N. AND METAXAS, D. 1996. Realistic animation of liquids. *Graph. Models Image Process. 58*, 471–483.

GAMITO, M. N., LOPES, P. F., AND GOMES, M. R. 1995. Two-dimensional simulation of gaseous phenomena using vortex particles. In *In Proceedings of the 6th Eurographics Workshop on Computer Animation and Simulation*. Springer-Verlag, 3–15.

GUPTA, M. AND NARASIMHAN, S. G. 2007. Legendre fluids: a unified framework for analytic reduced space modeling and rendering of participating media. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*. SCA '07. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 17–25.

GUSTAFSON, K. AND HARTMAN, R. 1983. Divergence-Free Bases for Finite Element Schemes in Hydrodynamics. *SIAM Journal on Numerical Analysis 20*, 697–721.

LENTINE, M., ZHENG, W., AND FEDKIW, R. 2010. A novel algorithm for incompressible flow using only a coarse grid projection. In *ACM SIGGRAPH 2010 papers*. SIGGRAPH '10. ACM, New York, NY, USA, 114:1–114:9.

LOSASSO, F., GIBOU, F., AND FEDKIW, R. 2004. Simulating water and smoke with an octree data structure. In *ACM SIGGRAPH 2004 Papers*. SIGGRAPH '04. ACM, New York, NY, USA, 457–462.

MARSDEN, J. E. AND RATIU, T. S. 1999. *Introduction to Mechanics and Symmetry*, Second ed. Number 17 in Texts in Applied Mathematics. Springer-Verlag New York, New York, NY.

MCNAMARA, A., TREUILLE, A., POPOVIĆ, Z., AND STAM, J. 2004. Fluid control using the adjoint method. In *ACM SIGGRAPH 2004 Papers*. SIGGRAPH '04. ACM, New York, NY, USA, 449–456.

MULLEN, P., CRANE, K., PAVLOV, D., TONG, Y., AND DESBRUN, M. 2009. Energy-preserving integrators for fluid animation. In *ACM SIGGRAPH 2009 papers*. SIGGRAPH '09. ACM, New York, NY, USA, 38:1–38:8.

MÜLLER, M., CHARYPAR, D., AND GROSS, M. 2003. Particle-based fluid simulation for interactive applications. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*. SCA '03. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 154–159.

ORSZAG, S. A. 1969. Numerical methods for the simulation of turbulence. *Phys. Fluids 12*, 250–257.

ORSZAG, S. A. AND PATTERSON, G. 1972. Numerical simulation of three-dimensional homogeneous isotropic turbulence. *Phys. Rev. Lett. 28*, 76–79.

PARK, S. I. AND KIM, M. J. 2005. Vortex fluid for gaseous phenomena. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*. SCA '05. ACM, New York, NY, USA, 261–270.

POINCARÉ, H. 1901. Sur une forme nouvelle des équations de la méchanique. *C.R. Acad. Sci. 132*, 369–371.

ROGALLO, R., MOIN, P., AND REYNOLDS, W. 1981. Numerical experiments in homogeneous turbulence. *NASA TM-81315.*.

SELLE, A., FEDKIW, R., KIM, B., LIU, Y., AND ROSSIGNAC, J. 2008. An unconditionally stable MacCormack method. *J. Sci. Comput. 35*, 350–371.

SILBERMAN, I. 1954. Planetary Waves in the Atmosphere. *Journal of Meteorology 11*, 27–34.

STAM, J. 1999. Stable fluids. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*. SIGGRAPH '99. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 121–128.

STAM, J. 2002. A simple fluid solver based on the FFT. *J. Graph. Tools 6*, 43–52.

STAM, J. AND FIUME, E. 1993. Turbulent wind fields for gaseous phenomena. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*. SIGGRAPH '93. ACM, New York, NY, USA, 369–376.

TREUILLE, A., LEWIS, A., AND POPOVIĆ, Z. 2006. Model reduction for real-time fluids. In *ACM SIGGRAPH 2006 Papers*. SIGGRAPH '06. ACM, New York, NY, USA, 826–834.

TREUILLE, A., MCNAMARA, A., POPOVIĆ, Z., AND STAM, J. 2003. Keyframe control of smoke simulations. In *ACM SIGGRAPH 2003 Papers*. SIGGRAPH '03. ACM, New York, NY, USA, 716–723.

TWIGG, C. D. AND JAMES, D. L. 2008. Backward steps in rigid body simulation. In *ACM SIGGRAPH 2008 papers*. SIGGRAPH '08. ACM, New York, NY, USA, 25:1–25:10.

WEISSMANN, S. AND PINKALL, U. 2010. Filament-based smoke with vortex shedding and variational reconnection. In *ACM SIGGRAPH 2010 papers*. SIGGRAPH '10. ACM, New York, NY, USA, 115:1–115:12.

WICKE, M., STANTON, M., AND TREUILLE, A. 2009. Modular bases for fluid dynamics. In *ACM SIGGRAPH 2009 papers*. SIGGRAPH '09. ACM, New York, NY, USA, 39:1–39:8.

YUDOVICH, V. I. 1963. Non-stationary flow of an ideal incompressible liquid. *USSR Computational Mathematics and Mathematical Physics 3,* 6, 1407 – 1456.

ZHU, Y. AND BRIDSON, R. 2005. Animating sand as a fluid. In *ACM SIGGRAPH 2005 Papers*. SIGGRAPH '05. ACM, New York, NY, USA, 965–972.

Received July 2011; accepted July 2011