Fluid Simulation in Bases of Laplacian Eigenfunctions

by

Tyler de Witt

A thesis submitted in conformity with the requirements
for the degree of Master of Science
Graduate Department of Computer Science
University of Toronto

Abstract

Fluid Simulation in Bases of Laplacian Eigenfunctions

Tyler de Witt
Master of Science
Graduate Department of Computer Science
University of Toronto
2010

We present a novel method for the simulation of incompressible fluids. In contrast to existing grid based and particle methods, we choose a spatial representation of vorticity in a basis of Laplacian eigenfunctions. In this thesis, we show that unique properties of this basis make it useful for computer graphics applications. Particularly, the Navier-Stokes equations reduce to a compact form that is elegant and practical, permitting time integration schemes that operate directly in the reduced space of basis coefficients. These time integration schemes are efficient and energy preserving. For a number of useful geometries, our basis functions are analytic. We extend our method to work on simplicial meshes through the use of discrete exterior calculus.

Contents

Chapter 1

Introduction

I attest to being one of many to have taken pleasure in observing the delicate patterns of cream stirred into black coffee, watching wisps of smoke curl from a cigar in a still room, staring mesmerized at a camp fire or sitting by the ocean for hours watching waves break against the shore. Whatever the reason, fluid motion is naturally captivating. It is no wonder it has been the source of inspiration and study in a diverse range of fields. For centuries, it has piqued the imaginations and curiosity of artists, mathematicians and scientists.

Leonardo da Vinci was fascinated with water. His *Codex Leicester* and *Codex Arundel* manuscripts, dating from the early 16^{th} century, were a collection of scientific writings with a substantial portion devoted to water, its movement, and machines for controlling it. He made observations about everything from erosion to pipe hydraulics. These were all augmented by masterful illustrations such as those in Figure 1.1 showing the formation of eddies as water flows past obstacles.

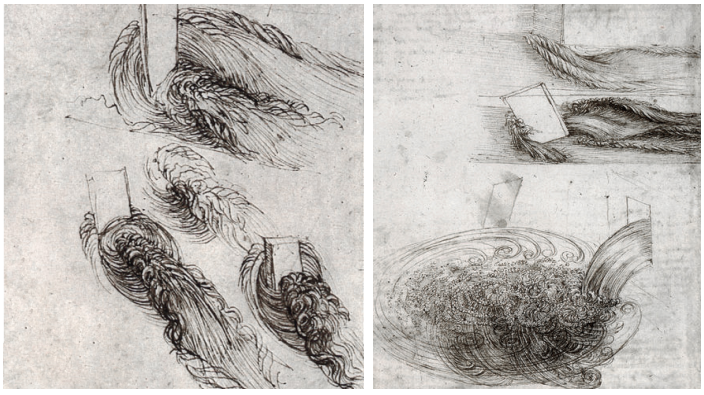


Figure 1.1: Illustrations of turbulent wakes from da Vinci's *Codex Leicester* and *Codex Arundel*.

Mathematicians have taken a special interest in the motion of fluids. While scientists and engineers are concerned foremost with prediction of physical experiments, mathematicians have an affinity for exploring the strange fundamental properties of idealized fluid models, such as the Navier-Stokes equations that describe the motion of an ideal incompressible medium. Such abstractions are extremely interesting in their own right, evidenced by the amount of deep technical work their study has garnered. Existence and uniqueness of the three dimensional Navier-Stokes equations are still among the most important open problems in mathematics. This is all the more fascinating when one considers that these models still

remain rooted in the observation of nature.

In the context of art, fluid motion is present both as a subject and as a physical medium. Techniques such as water color painting or paper marbling rely on fluids to impart a strong character to the resulting piece. An iconic example is Van Gogh's 'Starry Night'. Even as a static painting, the swirling colors of the sky impart a compelling sense of motion.

Figure 1.2: Left: Van Gogh's 'Starry Night' Right: Paper marbling

More recently, animation of fluid motion has become possible, beginning with the hand drawn animations of the 1950s. Joseph Gilland wrote a book detailing the craft, telling stories about the typical workflow of these artists spending many days observing nature, attempting to distill the emergent perceptual features to be incorporated into their work [14]. Michel Gagne self-describes the effects animator as a "scientific magician ... creating lighting, and conjuring fiery whirlpools, all from the tip of a pencil" [14]. Perhaps most striking is the the sheer number of pencil-miles spent animating scenes like an avalanche or a storm, a testament to how compelling fluid motion can inspire such life long dedication.

With the advent of computing, systems for animating fluids have been developed to alleviate some of the tedious effort required. For the most part, these animation systems use variants taken from the mathematical sciences to automate a physical process computationally. Compared to hand drawn workflow, note the strong difference in approach. The traditional animator works top down: the process begins with an artistic intention based on a learned perception of fluid motion, and proceeds to finer and finer tangible detail through illustration. In contrast, computer simulation based on a physical model requires the opposite: one begins with a detailed low level model and hopes that turning the crank will produce the desired emergent behavior.

Here we begin to see a problem. It is very difficult in a bottom up approach for an animator to incorporate global artistic intent. Indeed, expressive control of physically based simulation is recognized as a significant open problem in computer graphics research. It is the prime motivation for the present work, but is too broad to be addressed here in full generality. Instead we narrow our focus as follows.

Assuming that a computer animator's foremost goal is to produce captivating animations, we address *designing* fluid simulation methods with this goal in mind. In this thesis we present a novel method for fluid simulation that is both physically based and has unique qualitative properties that distinct from existing techniques. Briefly, our method comprises a representation of fluid's velocity field in a reduced basis of continuous functions, and a time integration scheme derived from physical laws that works directly with the basis *coefficients*. This paradigm represents a new direction when compared to existing computer graphics fluid

simulators that rely on a grid or particles for tracking physical quantities. In addition to avoiding many of the inherent problems associated with existing methods, our technique has many unique and compelling properties that will require further exploration, some of which we believe may extend to artistic control. In the following chapters, we will describe our method, compare it to previous work, demonstrate its qualities and discuss future research directions.

1.1 Outline

This thesis is structured as follows:

- Chapter 2 reviews some some mathematical preliminaries.
- Chapter 3 presents a general overview of our method and describes its unique properties.
- Chapter 4 compares our work to some relevant previous work in geometric mechanics and the computer graphics literature.
- Chapter 5 describes our method in technical detail.
- Chapter 6 shows visual results and discusses time and memory performance.
- Chapter 7 discusses improvements and future research directions.

Chapter 2

Preliminary Concepts

We begin with a review of some mathematical and physical concepts. The goal in our presentation is to provide intuitive explanations sufficient for the understanding of our method. Our choice of topics selective, favouring intutition over thoroughness, so readers new to these concepts may wish to consult the cited references for further detail. Readers familiar with this material may safely skip this section.

2.1 Function Spaces

A *function space* is a vector space of functions spanned by a set of n basis functions $\{\phi_i\}$. If a function f is an element of this space, it can be described as a linear combination of basis functions weighted by scalar coefficients $\{f_i\}$:

$$f = \sum_i f_i \phi_i.$$

Such representations can be very convenient. Even if the basis functions have complicated expressions, any f is uniquely described by its coefficients $\{f_i\}$ which is just a vector in the Euclidean space $\mathbb{R}^n$.

Depending on the properties of the basis functions, operations that are difficult to perform on f may be much simpler to perform on its basis representation $\{f_i\}$. For example, if the basis is *orthogonal*, then

$$\int f^2 = \sum_i f_i^2.$$

This is the L^2 norm, also called the *energy* of the function due to its meaning in certain physical contexts. The integral on the left hand side may be difficult to evaluate. However, if we know the representation of f in its basis coefficients $\{f_i\}$, the right hand sum is a simple calculation. Christensen [8] is a good reference for more detail about function spaces, functional analysis and Fourier analysis.

2.2 Laplacian Eigenfunctions

When applied to a quantity over a domain, the *Laplacian operator* measures at every point the difference between the quantity and the average of its surroundings. Its exact form differs depending on the context and nature of the quantity acted upon, but the intuition remains intact. For our purposes, we will be most interested in the Laplacian of vector fields in three dimensional space. The *vector Laplacian* acts on a vector field and returns a vector field. It is defined in traditional vector calculus notation as:

$$\begin{aligned}\Delta u &= \nabla(\nabla \cdot u) - \nabla \times (\nabla \times u) \\ &= \operatorname{grad}(\operatorname{div} u) - \operatorname{curl}^2 u\end{aligned}$$

for a vector field u.

For ideal fluid dynamics, the velocity vector field is divergence free. In this case the the first term of the vector Laplacian vanishes. To within a sign, this makes the Laplacian for divergence free fields equivalent to curl^2:

$$\begin{aligned}\Delta u &= \operatorname{curl}^2 u \\ \operatorname{div}(u) &= 0.\end{aligned}$$

An *eigenfunction* of an operator is one that changes only linearly in magnitude when acted upon. An eigenfunction of the Laplace operator satisfies

$$\Delta \phi_k = \lambda_k \phi_k,$$

for basis functions ϕ_k and scalar values λ_k. Over a domain D, *Laplacian eigenfunctions* are the set of functions satisfying the spatial Helmholtz equation

$$\Delta \phi_k - \lambda_k \phi = 0. \tag{2.1}$$

Depending on D and the context in which they are used, Laplacian eigenfunctions have various forms and names. For example, over the unbounded real line, they are the set of sinusoids forming the Fourier basis. Over a sphere, Laplacian eigenfunctions are the spherical harmonics. In physics, they often appear as the resonant acoustic or electromagnetic modes in a cavity, or the vibrational modes of a string.

Laplacian eigenfunctions on a domain are mutually orthogonal, and form a basis for L^2 functions over this domain. Taking the Fourier basis as an example, any L^2 function can be represented as the sum of sinusoids. However, in general it may take an infinite set of basis functions to do so exactly. Instead, we may choose to approximate the signal using a finite number of them. The approximation error can be made very small by using a sufficient number of basis functions, or choosing a set of basis functions that in the first place most resemble typical signals. For example, the Fourier basis is well suited for approximating smooth, periodic functions because the basis functions themselves are smooth and periodic.

To represent vorticity, we will soon make use of Laplacian eigenfunctions of a 2-D rectangular domain. These functions have the form

$$\phi_k = \sin(k_1 x)\sin(k_2 y), \tag{2.2}$$

where k_1, k_2 are positive integers. One can verify that $\Delta\phi_k = \lambda_k\phi_k$ where $\lambda_k = k_1^2 + k_2^2$. These functions are plotted on a $[0,\pi]\times[0,\pi]$ domain in Figure 2.1 for $1 \le k1, k2 \le 3$.

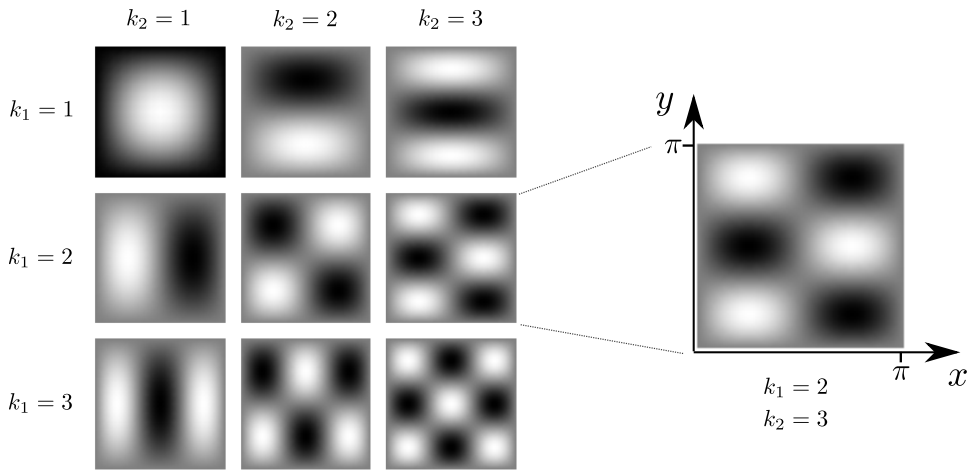


Figure 2.1: Left: Scalar Laplacian eigenfunctions of a 2-D square domain, described Eq. 2.2 for $1 \le k_1, k_2 \le 3$. Black and white correspond to negative and positive values respectively. These functions are also known as the 2-D Fourier Sine Basis. Right: Close up of $\phi_{2,3}$.

2.3 Vector Fields and their Vorticity

The motion of an incompressible medium can be characterized by a divergence free velocity vector field over a domain D. This velocity field u assigns a vector to every point x in D, subject to a zero divergence constraint and a boundary condition on ∂D. In three dimensions, taking x,y and z as spatial coordinates and $\mathbf{a_x}, \mathbf{a_y}, \mathbf{a_z}$ as unit basis vectors, this can be described as

$$\begin{aligned} u(x,y,z) &= u_x(x,y,z)\mathbf{a_x} + u_y(x,y,z)\mathbf{a_y} + u_z(x,y,z)\mathbf{a_z}, \\ \operatorname{div}(u) &= 0, \\ u\cdot n &= 0 \text{ over } \partial D, \end{aligned} \tag{2.3}$$

where n is a unit vector normal to the boundary. The condition $u \cdot n = 0$ is called the *free slip* condition, allowing only a tangential velocity component on ∂D.

The Helmholtz Hodge decomposition theorem guarantees that every vector field can be decomposed into the sum of a divergence free field (also called a *solenoidal* field) and a curl free field (also called *irrotational*). When u is solenoidal, it is uniquely identified by its curl. In fluid dynamics, this quantity is known as the *vorticity*, defined as

$$\omega = \operatorname{curl} u.$$

Physically, in three dimensions vorticity measures the tendency of a fluid to rotate about an axis perpendicular to the velocity field. When restricted to a 2-D domain, for example

the xy plane, this axis must lie along a new perpendicular spatial direction z. In this case, $\omega = \text{curl}\,(u(x, y))$ can only have an $\mathbf{a_z}$ component. As a result, this single component can be interpreted as a scalar function with positive and negative values corresponding to clockwise and counter-clockwise vorticity, respectively. In three dimensions, both ω and u are vector quantities and may point in any direction, but always remain perpendicular to each other.

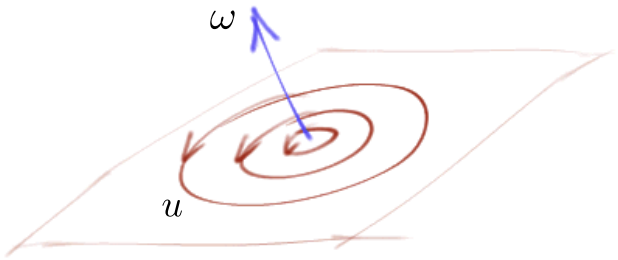


Figure 2.2: The vorticity (blue) of a 2-D vector field (red)

2.4 Dynamical Systems

Dynamical systems describe the time evolution of processes and systems. They encompass two main things: a description of a system's state, and how it changes with time.

Examples of a state include the position of a particle, the rotation angle of a swinging pendulum, or the density field of a gas. The set of all possible states of a system is called the *configuration space*. For physical systems, the state will change continuously and hence the configuration space takes the form of a continuous manifold. Particular states correspond to points on this manifold.

As time progresses, the state changes and traces out a 1-dimensional curve through the configuration space. At any point, we may assign a tangent vector describing the direction of this change. The set of all tangent vectors is called the *tangent space*. A dynamical system assigns an element of the tangent space to every point in configuration space.

Our preceding discussion has been geometric in nature. A dynamical system has a direct analogue as a differential equation

$$\frac{dx}{dt} = f(x, t),$$

where x is a point on the configuration space, and f is a function assigning a tangent vector $\frac{dx}{dt}$ to every point. Along with a specification of the configuration space, this equation completely describes the time evolution of the system. To predict the state x of the system at a future time, we must *integrate* (solve) this equation. Depending on the form of f this may be difficult or impossible to do analytically. In these cases, one must resort to numerical integration. More detail about dynamical systems and their numerical integration can be found in [15] and [19].

2.5 Galerkin Projection

Beginning with a continuous differential equation, the *Galerkin projection* produces a discrete differential equation by approximating variables as series expansions in a desired basis. For example, to solve the nonlinear differential equation

$$\frac{du}{dt} + u^2 = 0$$

we could substitute a representation of u in some basis $\{\phi_k\}$

$$u = \sum_i^\infty u_i \phi_i$$

to obtain

$$\frac{d}{dt} \sum_i^\infty u_i \phi_i + \left(\sum_j^\infty u_j \phi_j \right)^2 = 0.$$

If the series expansions are truncated, the equations become finite dimensional. Solution of the resulting approximation amounts to determining the *basis coefficients* such that the equation is satisfied. Expanding and equating basis function coefficients will often result in a system of equations that is easier to solve than the original differential equation.

Additionally, the solution method is often facilitated by exploiting properties of the basis functions themselves. For example, the basis functions may be orthogonal, making some products zero when integrated globally. The basis functions may have compact support, making their product zero nearly everywhere.

Galerkin methods are a general category with subcategories identified by the particular choice of basis. Finite Element Methods (FEM) are the Galerkin method applied with a basis of piecewise polynomials. These basis functions possess compact support. Due to this property, FEM typically results in a sparse matrix with is solved numerically. Spectral methods are also Galerkin methods, but use a Fourier series basis, enabling the Fast Fourier Transform (FFT) algorithm to be used as a highly efficient method of solution.

2.6 The Euler and Navier-Stokes Fluid Equations

Fluid motion is a dynamical system. It has a configuration space and dynamics described by differential equations. For an incompressible medium, the configuration space is the set of all divergence free vector fields. The time evolution is described by the Euler fluid equation

$$\frac{\partial u}{\partial t} + \nabla_u u = -\nabla p \tag{2.4}$$

where u is the velocity field, and p is the *pressure*, a physical force that 'pushes back' and prevents the medium from being compressed or expanded, ensuring that u remains divergence free. The advection term $\nabla_u u$ is the directional derivative of the velocity field with respect

to itself. This equation states that the velocity field will be self-advected in the direction of a negative pressure gradient.

The Euler fluid equations describe an idealized medium that is *inviscid*, meaning it does not dissipate energy. In reality, all physical media must dissipate at least some energy when they move, due to the shearing motion that occurs in vortices causing molecules to rub against each other. This energy dissipation is a type of friction called *viscosity* and is modelled by adding a dissipative term to the Euler fluid equations, resulting in the Navier-Stokes equations:

$$\frac{\partial u}{\partial t} + \nabla_u u = -\nabla p + \nu \Delta u. \tag{2.5}$$

The kinematic viscosity parameter ν describes the fluids resistance to shearing, which tends to dampen the motion of small vortices. Examples of fluids with high and low viscosity are molasses and alcohol, respectively.

The Euler and Navier-Stokes equations can be restated stated in terms of the fluid's vorticity rather than its velocity. Taking the curl of both sides of Eq. 2.4 and Eq 2.5 and substituting $\omega = \operatorname{curl} u$ results in

$$\frac{\partial \omega}{\partial t} + \nabla_u \omega = \nu \Delta \omega. \tag{2.6}$$

$$\frac{\partial \omega}{\partial t} + \nabla_u \omega = 0. \tag{2.7}$$

In doing so, the pressure term conveniently vanishes as the curl of a gradient is zero. Equation 2.7 is also known as the *Helmholtz form* of the Euler fluid equations.

2.7 Solution Methods to Fluid Equations

The Euler and Navier-Stokes equations describe the dynamics of a continuous velocity field as it changes continuously over time. However, to solve these equations numerically, both time and space must be discretized. Here we categorize several existing solution methods based on their approach to discretizing space.

Eulerian Methods *Eulerian* methods store and compute fluid quantities (such as velocity, pressure and density) at fixed grid locations. The grid may be regular (forming a cubic lattice, for example) or an unstructured polygon mesh. The defining characteristic is that the sampling points remain fixed, and measurements change continuously as quantities flow past these points of reference. The mesh points are also used to define discretizations of the required Laplacian, gradient and advection operators present in the equations. Coupled with a suitable time integration scheme, solving the Navier-Stokes equations typically requires the solution to a linear or nonlinear system of equations at each time integration step.

Eulerian methods suffer from two apparent drawbacks. First, the representation of small scale detail depends on the resolution of the mesh, but as the mesh is refined the size of the system of equations grows rapidly along with computational cost. Second, the use of a mesh necessitates interpolation when evaluating quantities at locations away from grid points. This has a number of drawbacks.

First, interpolation schemes can cause artifical diffusion. This can be understood by thinking of interpolation as a low pass filter. Repeatedly applying interpolation at each time step tends to smooth out small scale detail. Interpolation schemes may also lead to unstable energy growth in the in the velocity field, causing the simulation to 'blow up'.

Eulerian methods can be applied to either the vorticity or velocity formulations of the fluid equations. Representing velocity is the most direct approach, as it is typically this quantity one is most interested in for visualizing a flow or advecting immersed particles. However, a notable drawback is that due to inaccuracies in interpolation and time integration, the velocity field will not remain divergence free. In this case, a correction step is required to remove the portion of the velocity field with non-zero divergence. This compuation is expensive and introduces other problems, such as artificial energy loss. Energy dissipation manifests itself visually as a fluid that is overly viscous, containing less lively turbulent detail.

In contrast, representing the fluid using its vorticity field has the advantage of eliminating the pressure field and guaranteeing that the corresponding velocity field will remain divergence free. However, one must then reconstruct the velocity field from vorticity. This involves inversion of the curl operator $\omega = \text{curl}^{-1} u$ using a formula like the Biot Savart formula

$$u(x) = \frac{1}{4\pi} \int_D \omega(z) \times \frac{x - z}{|x - z|^3} \, dz$$

which in general is difficult and expensive to evaluate.

Lagrangian Methods A second class of methods do not use a fixed mesh. Instead, physical quantities are attached to a number of discrete particles, which act as moving sample points. This is an example of a *Lagrangian* formulation: the frame of reference moves, while the measured physical quantities do not – they are simply carried along with the particles. Remaining properties that depend on interactions between particles (such as pressure) are interpolated anywhere in space by treating the existing particle locations and attributes as sampled data points. Lagrangian methods are often used to represent vorticity, either as particles or filaments. Solution of Eq. 2.7 is accomplished by advecting the discrete vortex elements through the velocity field at each timestep.

Lagrangian methods do not suffer from the inherent diffusion in Eulerian methods caused by continually resampling at grid locations. However, they have their own host of problems. To represent the entire fluid, there must be many particles and they must remain well distributed, necessitating techniques for splitting, merging and destroying particles. Additionally, accounting for viscosity is difficult. Viscosity implies a type of diffusion, which does not fit well with a scheme that stores and moves quantities at discrete points. Quantities cannot be diffused as they can only remain concentrated at discrete points.

When Lagrangian methods are used to represent a velocity field, the same difficulty arises as in an Eulerian scheme: maintaining the divergence free constraint. In this case , it is due to the complexity of interpolation with large numbers of particles. To make computation tractable, particle interactions are generally limited to within a specific distance, introducing global inaccuracies and nonlinearities that accrete to produce a field that has non-zero divergence.

When applied to vorticity, Lagrangian methods overcome the non-zero divergence issue, but still requires the expensive inversion $\omega = \text{curl}^{-1}$ to reconstruct the velocity field. Additionally, the discrete vorticity elements employed in three dimensions are often 1-dimensional *filaments* (as opposed to 0-dimensional particles), necessitating their own geometric discretization. Finally, boundary conditions are also difficult to enforce with vortex elements.

Chapter 3

Overview of our Method

In this chapter we present an overview of our method, focusing on the discretization of field quantities. In contrast to Eulerian and Lagrangian formalisms, our representation uses neither a grid nor particles, but instead relies on a finite basis of continuous analytic functions – the set of Laplacian eigenfunctions. We will show that this representation greatly facilitates the solution of fluid equations and overcomes many of the problems inherent in particle and grid based methods.

3.1 Vorticity and Velocity Fields as Laplacian Eigenfunctions

Let $\{\phi_k\}$ be a set of vorticity fields that are Laplacian eigenfunctions of a domain, as defined in Eq. 2.1. Let $\{\Phi_k\}$ represent the set of associated velocity fields, each satisfying $\operatorname{curl}\Phi_k = \phi_k$ by definition. Any vorticity field ω in L^2 can be represented in the $\{\phi_k\}$ basis as

$$\omega = \sum_i \omega_i \phi_i. \tag{3.1}$$

Similarly, the associated velocity field u satisfying $\omega = \operatorname{curl} u$ can also be represented in the basis Φ_k by taking the curl of Eq. 3.1. Note that corresponding velocity and vorticity fields have the *same coefficients* ω_i in both the Φ_k and ϕ_k representation, due to linearity of the curl operator:

$$\begin{aligned} u = \operatorname{curl}\omega &= \operatorname{curl}\left(\sum_i \omega_i \phi_i\right) \\ &= \sum_i \omega_i \operatorname{curl}\phi_i \\ &= \sum_i \omega_i \Phi_i. \end{aligned} \tag{3.2}$$

Figure 3.1 shows a set $\{\phi_k\}$ and $\{\Phi_k\}$ for a 2-D rectangular domain. We can use a superposition of the velocity fields to represent any field in the span of $\{\Phi_k\}$. Figure 3.2 shows examples of a superpositions. As more basis fields are used, finer scales of vorticity are achievable.

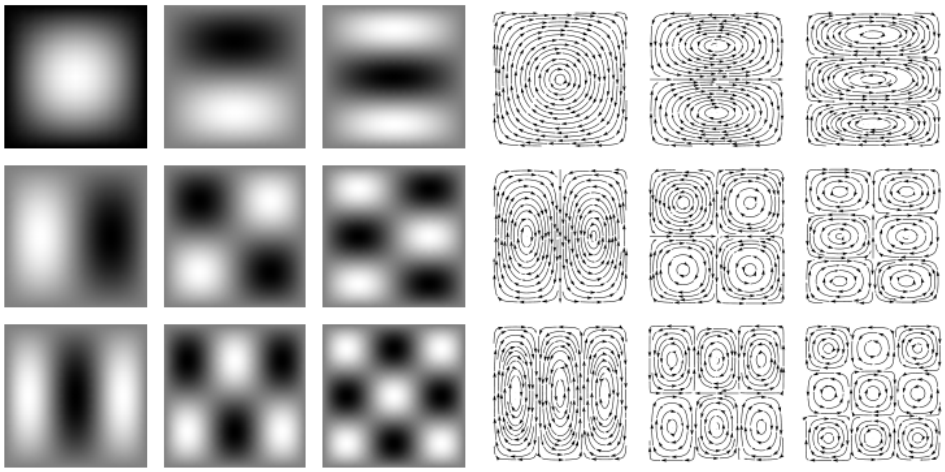


Figure 3.1: Left: Scalar vorticity basis functions $\{\phi_k\}$ described by Eq. 2.2. Black and white correspond to positive (clockwise) and negative (counter-clockwise) vorticity. Right: Corresponding velocity basis fields $\{\Phi_k\}$.

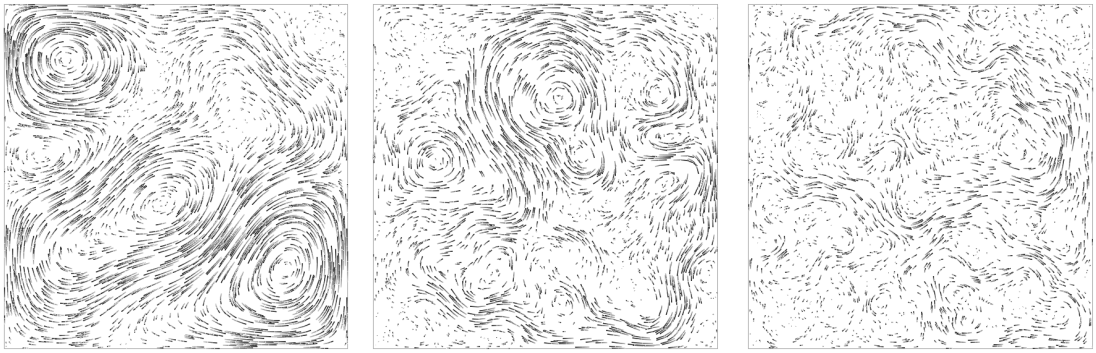


Figure 3.2: Examples of random superpositions of velocity basis fields. Every superposition of basis functions is a divergence free velocity field that respects a free slip boundary condition. Finer scale vorticity is achievable with a larger basis. From left to right: 16, 32, 64 basis functions.

The analytic expressions for ϕ_k were given in Eq 2.2. However, we also require expressions for the velocity basis fields Φ_k that satisfy $\operatorname{curl}\Phi_k = \phi_k$. To do so, we must evaluate the inverse curl operation $\Phi_k = \operatorname{curl}^{-1}\phi_k$ to be able to go back and forth. Herein lies the key to our method. Rather than resorting to the Biot Savart formula, because we work in a basis of Laplacian eigenfunctions curl^{-1} becomes a simple operation:

$$\begin{aligned}\Phi_k &= \text{curl}^{-1}\phi_k \\ &= \text{curl}^{-1}\left(\frac{1}{\lambda_k}\Delta\phi_k\right) \\ &= \text{curl}^{-1}\left(\frac{1}{\lambda_k}\text{curl}^{2}\phi_k\right) \\ &= \frac{1}{\lambda_k}\text{curl}^{-1}\left(\text{curl}^{2}\phi_k\right) \\ &= \frac{1}{\lambda_k}\text{curl}\,\phi_k. \end{aligned} \tag{3.3}$$

Using this result, we can easily derive expressions for $\{\Phi_k\}$ depicted in Figure 3.1

$$\begin{aligned}\Phi_k &= \frac{1}{\lambda_k}\text{curl}\,(\phi_k) \\ \Phi_k &= \frac{1}{k_1^2+k_2^2}\left(k_2\sin(k_1x)\cos(k_2y)\mathbf{a_x}\right. \\ &\qquad \left.-k_1\cos(k_1x)\sin(k_2y)\mathbf{a_y}\right). \end{aligned} \tag{3.4}$$

The Φ_k are themselves Laplacian eigenfunctions since

$$\begin{aligned}\text{curl}\,\Phi_i &= \phi_i \\ \text{curl}^{2}\Phi_i &= \text{curl}\,\phi_i \\ \text{curl}^{2}\Phi_i &= \lambda_i\Phi_i \\ \Delta\Phi_i &= \lambda_i\Phi_i. \end{aligned} \tag{3.5}$$

3.2 Advantages and Useful Properties of Laplacian Eigenfunctions

As noted previously, the coefficients of the vorticity basis representation and velocity basis representation are the same. Any fluid configuration is uniquely described by a set of basis coefficients. This eliminates the need to store the complete vector field, and instead we can work directly in the space of coefficient vectors $\mathbf{w} = [\omega_1\,\omega_2\,\ldots\,\omega_N]$ which is just the Euclidean vector space $\mathbb{R}^N$. Effectively, by choosing an appropriate basis we have reduced the configuration space to one that is more convenient. Any representable vector field in this space remains divergence free and satisfies the boundary conditions, as these properties are built into the $\{\Phi_k\}$ basis functions themselves.

We will exploit this property in Chapter 5 to show that Galerkin projection of the Navier-Stokes equations onto a Laplacian eigenfunction basis produces a form that is elegant and practical. The resulting time integration schemes work directly with the basis coefficients and a set of sparse precomputed matrices. Computationally, many of the time integrators are explicit, avoiding the need to solve a system of equations as is required for implicit methods. This makes them very simple to implement, and trivially parallelizable. As well,

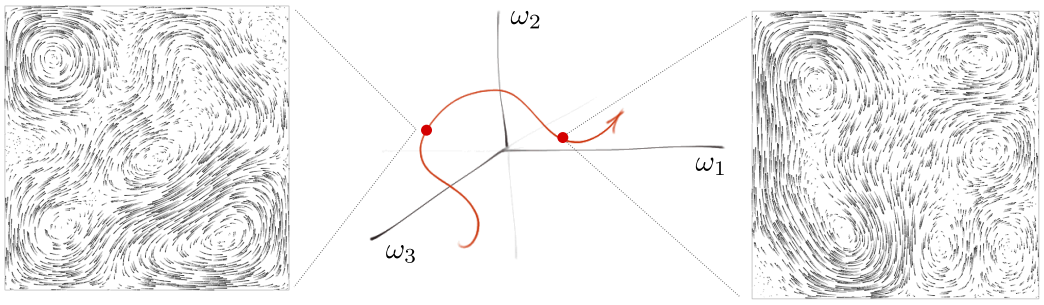


Figure 3.3: The configuration space of basis field coefficients is a Euclidean space. Each point represents a superposition of divergence free fields. A smooth curve describes a continuously changing velocity field: the motion of a fluid. In the figure, we limit the number of coordinate axes to three as an iconic representation. In reality, the configuration space is a high dimensional Euclidean space $\mathbb{R}^N$.

our algorithm scales with the number of basis functions, rather than with the underlying grid resolution.

Laplacian eigenfunctions form an orthogonal basis. Hence, the energy in each velocity basis field is independent of the rest, since $\int_D \Phi_i \Phi_j = 0$ when $i \neq j$. The energy content of a particular basis field Φ_k is the square of its coefficient ω_k^2, and the total physical energy E in a field is proportional to the sum of squares of its basis coefficients $E = \sum_i \omega_i^2$. In a Euclidean configuration space, this is just the distance from the origin of the basis coefficients when seen as a position vector. Orthogonality of the basis allows a simple means to compute and adjust energy. In Chapter 5 we exploit this property for developing energy preserving integration schemes exhibiting zero numerical viscosity. In other words, our method naturally preserves the interesting, turbulent motion of an inviscid fluid.

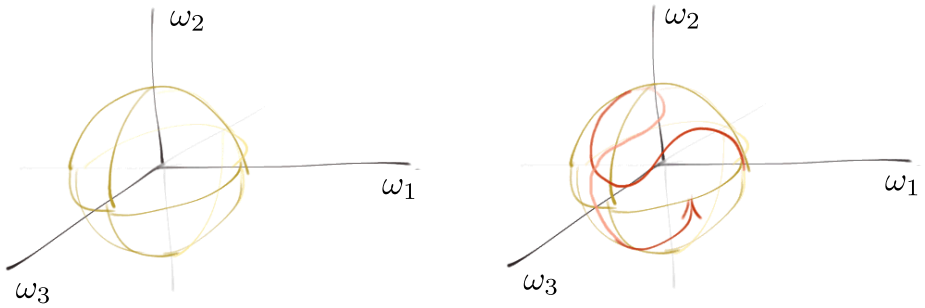


Figure 3.4: Because the basis is orthogonal, the energy in the fluid velocity field is the square of the distance from the origin in the configuration space of coefficients. Spheres represent surfaces of constant energy. The motion of an inviscid fluid will be constrained to one such surface.

Additionally, our basis has a property that works well in combination with orthogonality. As depicted in Fig. 3.1, each velocity basis field corresponds intuitively to vortices of varying scale: larger eigenvalues correspond to basis functions with finer scale turbulence. This allows energy in different scales of turbulence to be finely *controlled* with only trivial additional

computations. In Chapter 5 we show how the viscous term of the Navier-Stokes equations can be easily (but rigorously) incorporated through use of these properties.

For some simple but useful domains such as a rectangle or cube, our formulation is entirely analytic. This allows precise evaluation of the field at arbitrary locations without the need for a mesh or interpolation. This eliminates storage requirements and visual artifacts from the use of a grid. Analyticity also permits us to derive exact expressions for higher order derivatives of the velocity fields, allowing high order advection schemes, and symmetric time reversible integrators.

Chapter 4

Previous Work

We have presented a brief overview and the main contributions of our method. With this overview in mind, we now survey previous work before going into further technical detail. As with many research works in computer graphics, our work is adapted from and inspired by existing concepts in mathematics. Alain Fournier expressed this more succinctly: "In the time honored tradition of computer graphics, we can now go ahead and steal anything we can get [13]." In fairness, we review some previous mathematical works in addition to surveying the computer graphics literature, followed by a comparison to the present work.

4.1 Geometric Mechanics

The geometric viewpoint of fluid dynamics was first touched upon by Poincaré. In his famous 1901 paper[25], he showed that the dynamics of many physical systems shared the same description that Euler had first developed for the rotation of a rigid body. Essentially, Poincaré recognized that the same elegant equations could apply equally well to a variety of dynamical systems whose permitted movements could be described by continuous manifolds with group structure. For a rigid body, this manifold is the rotation group $SO(3)$ whereas for an incompressible fluid it is the more exotic *SDiff*, the infinite dimensional group of volume preserving diffeomorphisms.

Such notions seem quite abstract, but this viewpoint provided remarkable intuition into real physical properties. For example, in his 1969 paper, Arnold considered the curvature of the group manifold *SDiff*, providing fundamental results about the stability of fluid motion [3, 4]. He showed that because the curvature of the manifold is negative, the error in the prediction of a fluids trajectory grows exponentially with time. Arnold used this result to claim that weather prediction is impossible for periods longer than two weeks, essentially because the exponential error growth quickly trumps even the most exaggerated computing power and measurement apparatus.

In the recent mathematical control theory literature, Agrachev and Sarychev employed representations of vorticity in a basis of Laplacian eigenfunctions to prove controllability theorems for incompressible fluids [1]. It was this particular work that first inspired our efforts, although a similar procedure dates back at least to the works of Yudovich [36]. For two dimensional fluids, Yudovich was able to use functional basis representations to prove existence and uniqueness theorems that extend to the infinite dimensional case. Unfortunately, the function spaces considered in two dimensions do not share the same properties as their three dimensional counterparts, limiting their ability to be generalized.

In the past several decades, Marsden was a prominent researcher in the fields of geometric mechanics and dynamical systems. In particular, Marsden was a pioneer in the theory and tools for a *discrete* counterpart to continuous differential geometry and mechanics [18]. A discrete exterior calculus (DEC) was developed in cooperation with researchers including Desbrun and Hirani [9, 16] . DEC operates on simplicial meshes and is designed to preserve, as much as possible, the same structures and invariants that hold in a continuous setting. The results find applications for example in electromagnetism, computational fluid dynamics and computer graphics [22]. DEC is of particular importance to our work, as in Section 5.5 we use it as a recipe to extend our method to irregular simplicial meshes.

4.2 Computer Graphics Literature

4.2.1 Precursors to Physically Based Methods

The first fluid simulation techniques in graphics were not strongly physical. This was not primarily due to any lack of insight, but was instead necessitated by the goal of producing at least some type of visual result given the computational restrictions of the time. Some techniques employed simple parametric models like those of Max [20] which modulated a sum of sinusoids to represent ocean waves.[1] Others include the earliest use of a particle system, which was designed to emulate fire like phenomena [26].

Models became increasingly physically based. Kass and Miller simulated waves through an approximation of shallow water equations [17], which is itself a well established approximation to the Navier-Stokes equations. However, this method was fairly limited in that it was only capable of producing a height field, ruling out the simulation of phenomena such as breaking waves.

Physically based models of turbulence were used by Stam [32] and Shinya [27] to produce complex swirling flows. However, these stochastic models were unable to produce deterministic motion from initial conditions or to react to outside forces.

4.2.2 Eulerian Grid Based Methods

Grid based fluid techniques in graphics are numerous. We review a few notable ones. In 1996, Foster and Metaxas presented one of the first physically based fluid simulation methods in graphics, which solved the full Navier-Stokes equations using finite differences on a three dimensional grid [12]. Their implementation took into account solid obstacles, boundaries and the liquid/air interface. In comparison to previous techniques, this method was very general and had a solid physical foundation. Its main limitation was numerical instability. Due to the use of an explicit time integrator, without carefully monitoring of the step size the total energy of the system had the potential to grow uncontrollably.

In 1999, Stam presented an unconditionally stable solution method to the Navier-Stokes equations [30]. Instead of an explicit solver, Stam introduced a semi-implicit method which guaranteed the energy in the velocity field could never grow. The result was a practical method for fluid simulation that has become very popular. Although unconditionally stable, the main drawback of this method is the opposite of [12] – it dissipates too much energy.

[1] As a functional representation, that is similar in spirit to the present work.

Technically, the algorithm exhibits artificial numerical viscosity, which manifests visually as the dampening of small scale turbulent detail.

A number of works dealt with improving the energy behavior of this algorithm. Besides refining the grid, high order advection can improve accuracy to reduce the amount of dissipation. Fedkiw introduced to graphics the technique of *vorticity confinement*, a method used in computational fluid dynamics to reinject energy into vortical regions [11]. This method improves energy behavior, but does not fix the underlying problem as it is limited to amplifying existing vortices; in particular it cannot produce new vorticity that would otherwise have formed in a truly inviscid medium. Mullen et al. took a notably different approach to the energy dissipation problem. Rather than treating the symptom, they presented a novel Eulerian simulation method inspired by recent work in discrete exterior calculus and structure preserving fluid integrators [22]. The resulting method exhibits zero energy dissipation. However, it still exhibits numerical *diffusion*, a phenomenon that is unavoidable when working in an Eulerian framework due to the iterative interpolation and resampling.

4.2.3 Particle Methods

As discussed previously, Lagrangian particle methods do not suffer from the inherent diffusion in Eulerian methods caused by continually resampling at grid locations. They are well suited for simulating water like phenomena, as the mass particles do not diffuse, maintaining a sharp boundary in droplets and spray. Additionally, all the particles are concentrated in the water filled region. In contrast, a grid based method may waste many grid cells on large air filled regions that to not contribute any detail to the simulation.

Particle systems have a long history in the graphics literature. Early works use particle systems as ad hoc groups of independent vehicles imbued with a designed behavior, such as the work by Reeves [26].

More recent work adopts a more physical outlook. A notable work is that of Desbrun and Gascuel [10] who extended Smoothed Particle Hydrodynamics (SPH), a technique used by cosmologists for simulating the interaction of cosmic matter. The contribution in this work was to provide a sound physical formulation of particle based sampling and interaction forces, and the design of appropriate interpolation (smoothing) kernel functions for simulation of highly deformable substances. A number of follow up works by other researchers aimed at improving the computation performance and incompressibility [23, 29].

4.2.4 Vortex Methods

Vortex methods are Lagrangian methods employing vortex particles or filaments as discrete elements. The key advantage to such methods is the elimination of the divergence free constraint, since it is satisfied implicitly. Vortex methods are very well suited for simulating fluid phenomena in situations where the domain is entirely filled, such as wispy smoke in an air filled room. However, they are much less suited to phenomena like water, as it is not easy to incorporate a moving air-water boundary into the formulation.

Park and Kim were among the first to simulate fluids using vorticity tracked by particles [24]. In two dimensions this is appropriate as vorticity is scalar valued. In three dimensions however, vorticity is a vector. In reality vorticity in three dimensions is more accurately represented by filaments, a fact introduced and developed in the graphics literature by Angelidis et al. [2]. Recent work has expanded on filament representation, improving its computa-

tional efficiency and enabling the production of vortex sheets from interaction with obstacles [35].

4.2.5 Model Reduction

The computational complexity of many physically based simulation techniques do not scale well. For example, a three dimensional grid based fluid simulation method may require $O(N^3)$ time as an $N \times N \times N$ grid is refined. However, the increase in grid resolution may not lead to the same gains in perceived simulation quality. Model reduction is an attractive prospect which seeks to shrink the complexity of the configuration space, while attempting to maintain the same quality level. Model reduction has been applied to many areas in graphics including contact detection in deformable bodies [5], acoustic transfer and sound synthesis [6], and precomputed radiance transfer [28].

To a lesser extent, model reduction has also been applied to fluids. Stam presented an interactive fluid simulation method exploiting a Fourier basis representation [31]. This method is very efficient, but the use of a Fourier basis imposes the restriction of periodic 'wrap around' boundary conditions, and implies a rectangular domain.

Treuille et al. have simulated fluids in a low dimensional basis of divergence free fields, achieving real time simulation rates [33]. This method performs a time series analysis of an existing grid based fluid simulation to obtain an orthogonal basis of velocity fields and 'learn' the dynamics. The drawback of this technique is that the precomputation is very expensive. Additionally, the resulting learned model is linear, and cannot extend to flows that were not observed during training.

4.3 Comparison to Present Work

Here we comment on the relationship of our work to that previously reviewed. The mathematical concepts underlying our present work are not novel. Our main contribution is to recognize and apply them in a novel practical setting. Similarly, our use of discrete exterior calculus in Chapter 5.5 is purely as a tool to increase our work's range of applications.

Our method is related to but distinct from spectral Galerkin methods. We perform Galerkin projection of the Navier-Stokes equations onto a basis of Laplacian eigenfunctions. In some sense, this is similar to a spectral method. It differs because for the domains we consider, the Laplacian eigenfunctions are not always a Fourier series basis so a fast transform method akin to the Fast Fourier Transform is not always available. Additionally, we seek to determine the resulting coefficients analytically rather than through a numerical method.

The novelty of our method in computer graphics, however, is considerable. It is differentiated from particle and grid based methods since at its purest, it does not use either of these constructs. Although we use a simplicial mesh in our DEC formulation of Chapter 5.5, this is more as an extension to the core concepts rather than an intrinsic part of our work. We use a vorticity formulation, but it differs greatly from existing vortex methods because we do not use discrete Lagrangian elements. Additionally, through our choice of basis we avoid having to perform expensive inversion of $u = \text{curl}^{-1}\omega$ through the Biot Savart formula.

Our work is closest in spirit to the model reduction of Treuille et al. [33]. At their core, both use a finite dimensional basis of divergence free fields for representing fluids, and calculate dynamics directly within this reduced space. Beyond this similarity there are many differences. Trueille et al.'s method does not generate dynamics on its own, but relies on

observations of an existing fluid simulation to learn a linear model for the dynamics. In contrast, we derive dynamics directly from the Navier-Stokes equations. As a result, the precomputation time is vastly reduced and the resulting dynamics do not depend on any existing simulation. In addition, the dynamics of our method are nonlinear, producing more accurate (and perhaps more interesting) behavior.

Finally, our method is the first to make use of continuous analytic expressions for describing fluid velocity field and its dynamics. All previous fluid simulation methods in graphics discretize space. For simple (yet practical) domains, we instead use a finite number of analytic, continuous functions. We also derive exact analytic expressions for the time derivatives of a system's path through configuration space. Analyticity provides a number of immediate advantages such as reducing storage requirements by allowing quantities to be computed on the fly, rather than existing in memory. It also allows analytic derivation of higher order space and time derivatives, permitting very accurate advection schemes. We believe analyticity has potential to enabling well founded analysis of meta-techniques (for example, control policies) that use our fluid simulation as a foundation.

Chapter 5

Fluid Simulation in Bases of Laplacian Eigenfunctions

Having outlined the benefits of our method in relation to existing work, we will now turn to the technical detail. In this chapter we discuss the dynamics of fluids represented in Laplacian eigenfunctions and derive time integration schemes from the Euler and Navier-Stokes fluid equations. As a preface to this discussion, we begin with an analogy to a rotating rigid body to build some intuition about the geometric viewpoint of fluids, an interpretation championed by mathematicians such as Poincaré and Arnold.

5.1 Dynamics: Analogy to Rigid Body Rotation

Euler's equations for the rotation of a rigid body in the absence of torques or external forces are

$$\begin{aligned} I_1 \frac{\partial \omega_1}{\partial t} + (I_3 - I_2)\omega_2\omega_3 = 0, \\ I_2 \frac{\partial \omega_2}{\partial t} + (I_1 - I_3)\omega_3\omega_1 = 0, \\ I_3 \frac{\partial \omega_3}{\partial t} + (I_2 - I_1)\omega_1\omega_2 = 0 \end{aligned} \tag{5.1}$$

where I_k are the principal moments of inertia, and ω_k is the angular velocity along each of three principal axes. These equations describe a dynamical system in which the tangent vector $\frac{d\omega_k}{dt}$ (describing the change in angular momentum) is a a quadratic function of the existing angular velocities $\{\omega_i\}$.

Note that when the body is rotating about a single axis (for example, only ω_1 is nonzero), the time derivatives satisfy $\frac{\partial \omega}{\partial t} = 0$ and the system is in dynamic equilibrium: it will continue spinning about this single axis indefinitely. As soon as more than one component of the angular velocity is nonzero, the motion is no longer stationary as angular momentum is then continually transferred among the three principal axes.

In the absence of friction, this transfer of angular momentum will be perpetual and the total energy will be conserved. Imagining a three dimensional Euclidean space with ω_k along each of three axes, the path taken in this space must lie on a surface of constant energy. This is the configuration space of the system. The surface will be an ellipsoid, whose dimensions

are determined by the moments of inertia. The basis is orthogonal, and for appropriate scaling this surface will be spherical.

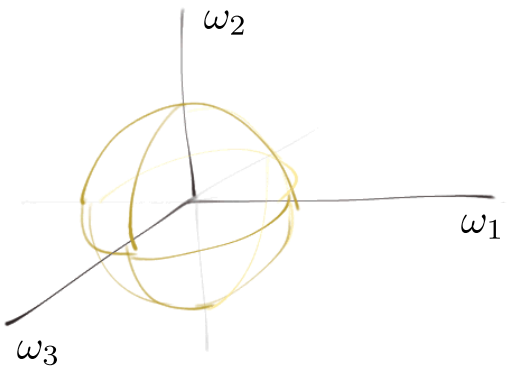


Figure 5.1: The configuration space of a rotating rigid body – or an incompressible fluid. For a rigid body, $\{\omega_k\}$ represent angular momentum along the object's three principal axes. The configuration space of an incompressible fluid is the infinite dimensional analog, with $\{\omega_k\}$ representing energy in velocity fields which are Laplacian eigenfunctions. In both cases, surfaces of constant energy for non-dissipative systems are spherical.

Poincaré had the insight that the motion of other dynamical systems could be described in much the same way [19]. In the case of a fluid, the primary difference is that instead of three principal axes, there are infinitely many. The axes correspond to the Laplacian eigenfunction basis fields described in Chapter 3. Many of the properties described for the rigid body carry over. Fluid motion in a single basis field is a stationary flow. The presence of multiple basis fields will break the equilibrium and induce quadratic energy transfer to other components, producing the turbulent, swirly behavior characteristic of an inviscid fluid. The configuration space is a surface of constant energy embedded in a high dimensional Euclidean space. With appropriate scaling the basis functions are orthonormal, making the surfaces of constant energy high dimensional spheres.

5.1.1 Derivation of Analytic Advection Operator

We begin by restating the Helmholtz formulation of the Euler fluid equation

$$\frac{\partial \omega}{\partial t} + \nabla_u \omega = 0.$$

This equation describes the motion of an inviscid, incompressible fluid in terms of its vorticity. Intuitively, it states that the vorticity field ω will be advected by the velocity field u. The nonlinear advection operator ∇_u is the directional derivative with respect to the vector field u. The operation of differentiating a vector field with respect to another vector field is a special case of a Lie derivative, notated as $\mathcal{L}_u w$. This is equivalent to the Jacobi-Lie bracket of vector fields $[u, \omega]$. In three dimensional space, it can be shown that this reduces to $\operatorname{curl}(w \times u)$ in traditional vector calculus notation. All these representations are equivalent:

$$\nabla_u \omega = \mathcal{L}_u w = [u, \omega] = \text{curl}\,(\omega \times u).$$

We adopt the bracket notation for simplicity

$$\frac{\partial \omega}{\partial t} = [u, \omega]. \tag{5.2}$$

5.1.2 Galerkin Projection

We project $\frac{\partial \omega}{\partial t}$, ω and u onto the Laplacian eigenfunction bases $\{\phi_k\}$, $\{\phi_i\}$ and $\{\Phi_j\}$:

$$\begin{aligned} \frac{\partial \omega}{\partial t} &= \sum_k \omega_k \phi_k \\ u &= \sum_i \omega_i \Phi_i \\ \omega &= \sum_j \omega_j \phi_j. \end{aligned}$$

Substituting these expansions into Eq. 5.2 one obtains

$$\frac{\partial}{\partial t} \sum_k \omega_k \phi_k = \left[\sum_i \omega_i \Phi_i, \sum_j \omega_j \phi_j \right].$$

Because the Jacobi-Lie bracket is a linear operator, this reduces to

$$\sum_k \frac{\partial \omega_k}{\partial t} \phi_k = \sum_i^\infty \sum_j^\infty \omega_i\, \omega_j [\Phi_i, \phi_j]. \tag{5.3}$$

Following the approach of [1], for now we assume that $\{\phi_k\}, \{\Phi_k\}$ are closed under the bracket $[\Phi_i, \phi_j]$. In other words, the result of the bracket evaluation can be expressed exactly in the $\{\phi_k\}$ basis

$$[\,\Phi_i, \phi_j] = \sum_k C_{i,j}^k \phi_k. \tag{5.4}$$

We will discuss in Section 5.2 how to perform this evaluation. The key point for the current discussion is that the coefficients $C_{i,j}^k$ can be *precomputed* for a given domain. In the mathematical literature, they are known as the *structure coefficients* [19]. Combining with Eq. 5.7, once the $C_{i,j}^k$ have been precomputed, the k^{th} basis function coefficient of $\frac{\partial \omega}{\partial t}$ is

$$\frac{\partial \omega_k}{\partial t} = \sum_i^\infty \sum_j^\infty \omega_i\, \omega_j C_{i,j}^k. \tag{5.5}$$

We begin to see the similarities to the rigid body equations introduced in the preface. Eq. 5.5 describes the change in basis coefficients ω in terms of quadratic functions of the existing coefficients, weighted by the precomputed factors $C_{i,j}^k$. In the rigid body case, the constant weighted factors were ratios of the moments of inertia. Here they describe the precomputed interaction of the i^{th} and j^{th} basis fields for a particular domain.

5.1.3 Incorporating Viscosity

Our derivation began with the Euler fluid equations, ignoring viscosity to keep the exposition as clear as possible. However, it is not difficult to follow a similar approach for the Navier-Stokes equations taking into account the viscous term. The Navier-Stokes equations in vorticity form are

$$\frac{\partial \omega}{\partial t} + \nabla_u \omega = \nu \Delta \omega. \tag{5.6}$$

As the viscous term involves a Laplacian, an expansion in a Laplacian eigenfunction basis is particularly convenient:

$$\begin{aligned} \nu \Delta \omega &= \nu \Delta \sum_g \omega_g \phi_g \\ &= \nu \sum_g \lambda_g \omega_g \phi_g. \end{aligned}$$

Galerkin projection of the Navier-Stokes equations yields

$$\sum_k \frac{\partial \omega_k}{\partial t} \phi_k = \sum_i^\infty \sum_j^\infty \omega_i \omega_j [\Phi_i, \phi_j] - \nu \sum_g \lambda_g \omega_g \phi_g \tag{5.7}$$

and applying Eq. 5.4 yields the dynamics equations for each coefficient ω_k

$$\frac{\partial \omega_k}{\partial t} = \sum_i^\infty \sum_j^\infty \omega_i \omega_j C_{i,j}^k - \nu \lambda_k \omega_k.$$

Considering only the viscous term and ignoring the structure coefficients, we see that the effect of viscosity produces a first order linear differential equation in each component ω_k

$$\frac{\partial \omega_k}{\partial t} = -\nu \lambda_k \omega_k.$$

This means that each basis coefficient will decay exponentially with a time constant proportional to its eigenvalue

$$\omega_k(t) = \omega_k(0) e^{-\nu \lambda_k t}.$$

This is physically plausible, since larger eigenvalues correspond to smaller vortices which should decay faster.

5.1.4 Discretization

The previous derivation did not make any approximations, and Eq. 5.5 is still an exact continuous expression assuming an infinite number of basis functions are employed. However, to permit computation, a finite dimensional approximation is required. We truncate the basis expansion to n coefficients yielding

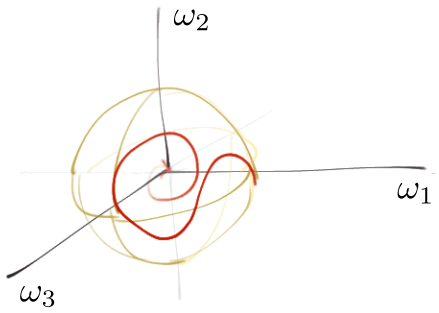


Figure 5.2: As a viscous fluid dissipates energy, its coefficients will decay with a time constant proportional to the eigenvalue of the associated Laplacian eigenfunction. In consequence the path will no longer remain on surfaces of constant energy, but instead spiral towards the origin.

$$\frac{\partial \omega_k}{\partial t} = \sum_i^n \sum_j^n \omega_i \omega_j C_{i,j}^k - \nu \lambda_k \omega_k.$$

This can be simplified to a matrix equation by treating ω as a column vector $\mathbf{w}$ of basis coefficients, $C_{i,j}^k$ as a set $\{\mathbf{C}_k\}$ of n-by-n square matrices, and $\mathbf{D}$ as a diagonal matrix with eigenvalues λ_k along the diagonal

$$\frac{\partial \mathbf{w}_k}{\partial t} = \mathbf{w}^T \mathbf{C}_k \mathbf{w} - \nu \mathbf{D} \mathbf{w}. \tag{5.8}$$

The $\frac{\partial \mathbf{w}_k}{\partial t}$ is hence a quadratic function of the components of the current state vector $\mathbf{w}$. The coefficients of each quadratic term $\omega_i \omega_j$ are stored in the i, j position of matrix $\mathbf{C}_k$.

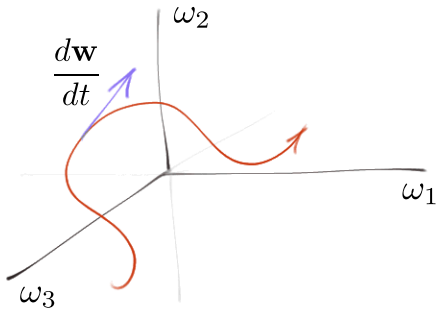


Figure 5.3: We derive an exact expression for the tangent vector in coefficient space as a function of the current state vector.

In the next section we show how to compute the entries of $\mathbf{C}_k$ matrices to satisfy

$$[\Phi_i, \phi_j] = \sum_k^n \mathbf{C}_k[i, j] \phi_k. \tag{5.9}$$

5.2 Basis Fields and Lie Bracket Evaluation: Analytic Solutions

For domains admitting analytic expressions, our method necessitates two main tasks. First, the Laplacian eigenfunctions themselves must be determined for a particular domain. Second, the Lie bracket for pairs of basis functions $[\Phi_i, \phi_j]$ must be evaluated, determining the entries in the $\mathbf{C}_k$ matrices. We will discuss these two problems in general terms and illustrate through the example of a 2-D rectangular domain. Derivations for additional analytic geometries are included in Appendix A.

Determining the basis of Laplacian eigenfunctions amounts to solving the homogeneous spatial Helmholtz equation

$$\Delta\omega - \lambda\omega = 0 \tag{5.10}$$

on a domain D under given boundary conditions. In our case, this is the velocity boundary condition $u \cdot n = 0$ called the *free slip* condition, allowing only a tangential velocity component at the boundary. Under these constraints, analytic solutions to Eq. 5.10 are in general difficult to find. Fortunately, this is a historically well studied problem in other fields such as physics, as analytic solutions for simple geometries were all that was available prior to recent advances in computing.

In particular, much can be gleaned from the solutions to spatial Helmholtz problem in electromagnetics, due to the similarity of boundary conditions. Finding the resonant modes of waveguides and metallic cavities is a problem parallel to our own and we can leverage the results to help solve Eq. 5.10. For example, in Appendix A, we use the expressions for the resonant modes of the electric field in a rectangular cavity.

Having determined the basis in analytic form, $[\Phi_i, \phi_j]$ can be evaluated. This can be done by directly evaluating $\operatorname{curl}(\phi_j \times \Phi_i)$. However, it is sometimes more convenient to work only with either the vorticity basis $\{\phi_k\}$ or the velocity basis $\{\Phi_k\}$. In the following we therefore present some additional equivalent expressions. To do so, we will make use of an identity which holds when both a and b are divergence free vector fields

$$\operatorname{curl}(a \times b) = b \times \operatorname{curl} a.$$

The bracket expansion can be rewritten entirely in terms of $\{\Phi_k\}$:

$$\begin{aligned}
[\Phi_i, \phi_j] &= \operatorname{curl}(\phi_j \times \Phi_i) \\
&= \Phi_i \times \operatorname{curl}(\phi_j) \\
&= \Phi_i \times \lambda_j \Phi_j \\
&= \lambda_j (\Phi_i \times \Phi_j)
\end{aligned} \tag{5.11}$$

where we also employed the substitution $\operatorname{curl}\phi_j = \lambda_j \Phi_j$ from Eq. 3.3. We can instead substitute $\Phi_i = \frac{1}{\lambda_i}\operatorname{curl}\phi_i$ to obtain an expression in terms of $\{\phi_k\}$:

$$[\Phi_i, \phi_j] = \frac{1}{\lambda_i}(\operatorname{curl}\phi_i \times \operatorname{curl}\phi_j). \tag{5.12}$$

Evaluating the bracket analytically using any of these methods is tedious but straightforward. What remains is to factor the result into a linear combination of basis functions

$\{\phi_k\}$. We require closure and sparseness, so that the resulting be stored in the $\{\mathbf{C}_k\}$ matrices of coefficients. Properties of the Jacobi-Lie bracket guarantee closure of $\{\phi_k\}$ under this operation. This ensures that the result will in fact be a divergence free field. However, there is no guarantee that the result is sparse. In the geometries we have examined to date whose basis functions are sinusoidal in nature, this in fact holds. In these cases, the bracket of basis functions gives rise to products of sinusoids which are themselves bandlimited. In fact, for any two functions f and g bandlimited by n and m respectively, their product will be bandlimited by $n + m$. However, it is unknown to us if the bracket for Laplacian eigenfunctions of arbitrary geometries always reduces to products of bandlimited functions.

5.2.1 Properties of $\mathbf{C}_k$ Matrices

The structure coefficient matrices $\mathbf{C}_k$ have a number of useful properties. Most importantly, these matrices are sparse. For the analytic geometries we've considered, only 1%-4% of the entries are nonzero. Not every pair of basis fields interacts to produce growth or decay in every other component.

Due to the cross product being an antisymmetric operator, we can see from Eq. 5.12 that these matrices will possess a type of antisymmetry. The entries are not exactly antisymmetric, due to the $\frac{1}{\lambda_i}$ factor. Instead we have

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

We can exploit this property to compute only *half* of the matrix entries and mirror the rest with this formula. Antisymmetry also implies that the diagonal entries must be zero. This makes sense intuitively since each basis field represents a stationary flow and does not interact with itself.

5.2.2 Lie Bracket Evaluation: 2-D Rectangular Domain

In Section 3, we introduced the basis fields for a bounded 2-D rectangle with free slip boundary condition. Here we provide an example of analytic bracket evaluation for pairs of these basis functions. For a 2-D $[0,\pi]\times[0,\pi]$ rectangle with vanishing vorticity boundary condition, the Laplacian eigenfunctions of vorticity take the form

$$\phi_k = \sin(k_1 x)\sin(k_2 y)\mathbf{a_z} \tag{5.13}$$

where $k = (k_1, k_2) \in \mathbb{Z}^2$ is a tuple which we call a *vector wave number*. The function ϕ_k satisfies $\Delta\phi_k = \lambda_k\phi_k$ for the eigenvalue $\lambda_k = k_1^2 + k_2^2$. The corresponding velocity basis fields $\{\Phi_k\}$ are

$$\begin{aligned}\Phi_k &= \operatorname{curl}\left(\Delta^{-1}\phi_k\right)\\ &= \frac{1}{k_1^2+k_2^2}\left(k_2\sin(k_1x)\cos(k_2y)\mathbf{a_x}\right.\\ &\qquad\left.-k_1\cos(k_1x)\sin(k_2y)\mathbf{a_y}\right).\end{aligned}$$

Because the expressions for ϕ_k are simpler than Φ_k, we choose to evaluate the bracket using Eq. 5.12. We begin by expanding curl (ϕ_a), recalling that a and b are vector wave numbers (a_1, a_2) and (b_1, b_2)

$$\begin{aligned}\operatorname{curl}\phi_a &= a_2\sin(a_1x)\cos(a_2y)\mathbf{a_x} - a_1\cos(a_1x)\sin(a_2y)\mathbf{a_y}\\ \operatorname{curl}\phi_b &= b_2\sin(b_1x)\cos(b_2y)\mathbf{a_x} - b_1\cos(b_1x)\sin(b_2y)\mathbf{a_y}.\end{aligned}$$

Next we evaluate their cross product

$$\begin{aligned}(\operatorname{curl}\phi_a \times \operatorname{curl}\phi_b)_x &= 0\\ (\operatorname{curl}\phi_a \times \operatorname{curl}\phi_b)_y &= 0\\ (\operatorname{curl}\phi_a \times \operatorname{curl}\phi_b)_z &= a_1b_2\cos(a_1x)\cos(b_2y)\sin(a_2x)\sin(b_1y)\\ &\quad - a_2b_1\cos(a_2x)\cos(b_1y)\sin(a_1x)\sin(b_2y).\end{aligned}$$

Note that in the end only the $\mathbf{a_z}$ component remains nonzero since the cross product produces a vector perpendicular to both operands. This is hopeful, since we expect the $\{\phi_k\}$ to span the result. The trigonometric identity $\cos(\alpha)\sin(\beta) = \frac{1}{2}\sin(\alpha+\beta) - \frac{1}{2}\sin(\alpha-\beta)$ allows simplification to a suitable form:

$$\begin{aligned}(\operatorname{curl}\phi_a \times \operatorname{curl}\phi_b)_z = \frac{1}{4}(&a_2b_1\sin((a_1-a_2)x)\sin((b_1-b_2)y)\\ &- a_1b_2\sin((a_1-a_2)x)\sin((b_1-b_2)y)\\ &+ a_2b_1\sin((a_1+a_2)x)\sin((b_1-b_2)y)\\ &+ a_1b_2\sin((a_1+a_2)x)\sin((b_1-b_2)y)\\ &- a_2b_1\sin((a_1-a_2)x)\sin((b_1+b_2)y)\\ &- a_1b_2\sin((a_1-a_2)x)\sin((b_1+b_2)y)\\ &- a_2b_1\sin((a_1+a_2)x)\sin((b_1+b_2)y)\\ &+ a_1b_2\sin((a_1+a_2)x)\sin((b_1+b_2)y))\\ (\operatorname{curl}\phi_a \times \operatorname{curl}\phi_b)_z = \frac{1}{4}(&(a_1b_2 - a_2b_1)\phi_{a_1+a_2,b_1+b_2}\\ &+ (a_1b_2 + a_2b_1)\phi_{a_1+a_2,b_1-b_2}\\ &- (a_1b_2 + a_2b_1)\phi_{a_1-a_2,b_1+b_2}\\ &-(a_1b_2 - a_2b_1)\phi_{a_1-a_2,b_1-b_2})\end{aligned}$$

which is indeed spanned by $\{\phi_k\}$. The result is a sum of basis fields of different wave number, the coefficients of which are added to the $\{\mathbf{C}_k\}$ matrices:

$$\begin{aligned}\mathbf{C}_{a_1+a_2,b_1+b_2}[a,b] &= \frac{1}{4}(a_1b_2 - a_2b_1)\\ \mathbf{C}_{a_1+a_2,b_1-b_2}[a,b] &= \frac{1}{4}(a_1b_2 + a_2b_1)\\ \mathbf{C}_{a_1-a_2,b_1+b_2}[a,b] &= -\frac{1}{4}(a_1b_2 + a_2b_1)\\ \mathbf{C}_{a_1-a_2,b_1-b_2}[a,b] &= -\frac{1}{4}(a_1b_2 - a_2b_1).\end{aligned}$$

Although the initial derivation is tedious, the result is a very simple procedure for computing the $\mathbf{C}_k$ for use in Eq. 5.8: iterate over all pairs of mode numbers a, b, adding the result to the $\mathbf{C}_k$ entries accordingly.

5.3 Time Discretization: Numerical Integration Schemes

In this section we describe a number of methods for time integration of Eq. 5.8. We begin by outlining a few unique properties of our formulation that enable various integration schemes.

Equation 5.8 provides an exact expression for the tangent vector $\frac{\partial \mathbf{w}_k}{\partial t}$ representing the time evolution of vorticity. We can also analytically calculate higher order time derivatives of $\mathbf{w}_k$ by differentiating Eq. 5.8. For example, ignoring the viscous term for clarity, the second and third time derivatives are

$$\begin{aligned}\frac{\partial^2 \mathbf{w}_k}{\partial t^2} &= \frac{\partial \mathbf{w}}{\partial t}^T \mathbf{C}_k \mathbf{w} + \mathbf{w}^T \mathbf{C}_k \frac{\partial \mathbf{w}}{\partial t}. \\ &= \left(\mathbf{w} + \frac{\partial \mathbf{w}}{\partial t}\right)^T \mathbf{C}_k \left(\mathbf{w} + \frac{\partial \mathbf{w}}{\partial t}\right).\end{aligned}$$

and

$$\frac{\partial^3 \mathbf{w}_k}{\partial t^3} = \left(\mathbf{w} + 2\frac{\partial \mathbf{w}}{\partial t} + \frac{\partial^2 \mathbf{w}}{\partial t^2}\right)^T \mathbf{C}_k \left(\mathbf{w} + 2\frac{\partial \mathbf{w}}{\partial t} + \frac{\partial^2 \mathbf{w}}{\partial t^2}\right).$$

Likewise, the higher-order derivatives follow the same pattern, with the coefficients following those of a binomial expansion. The procedure is identical when the viscous term is included, although the algebra is slightly more cumbersome. The main point is that accurate time derivatives of arbitrary order are available to us, and can be exploited in a number of time integration schemes.

A second observation relates to energy conservation. Under the Laplacian eigenfunction basis, surfaces of constant energy correspond to N dimensional spheres in a Euclidean space. In the absence of dissipation, the path must be constrained to this surface. Knowing this allows us to impose a correction to any time integration scheme to ensure ω stays on the

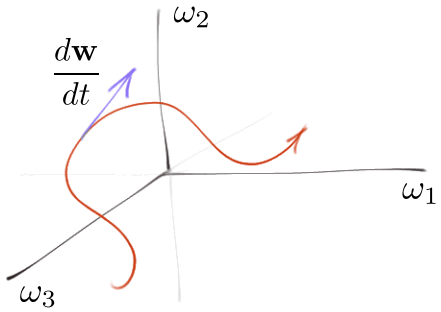


Figure 5.4: The tangent vector $\frac{\partial \mathbf{w}_k}{\partial t}$ representing the time evolution of vorticity.

surface. Computationally, this is very simple because projection onto a sphere is just a scaled renormalizing of the coefficient vector. Hence many of the time integration schemes presented follow a 'step and project' scheme.[1]

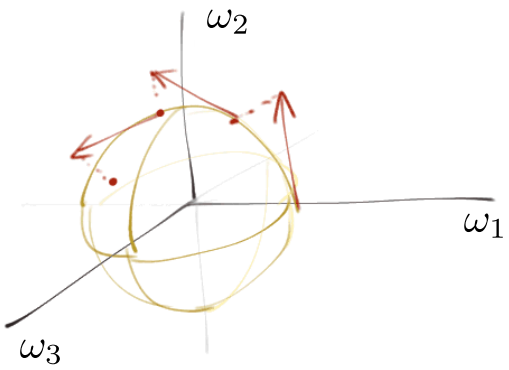


Figure 5.5: We can evaluate the tangent vector exactly, follow this direction for a timestep and reproject to the manifold of constant energy, an N dimensional spherical surface.

5.3.1 Euler Integration

We first consider the simplest case of an explicit Euler integration scheme (not to be confused with the Euler fluid equations):

$$\mathbf{w}_k[t + \Delta t] = \mathbf{w}_k[t] + \Delta t \frac{\partial \mathbf{w}_k}{\partial t}[t].$$

This scheme simply evaluates the velocity at the current position, and follows this direction for a small timestep. [2]

Note that we have an *exact* expression for the velocity at the current position in configuration space. This vector will always be tangent to the manifold of constant energy. However, following this tangent vector over a timestep causes the state to leave this manifold. As discussed, we can correct for this by projecting back to a sphere through normalization

$$\mathbf{w} = \frac{1}{|\mathbf{w}|}\mathbf{w}.$$

Viscosity is incorporated through exponential decay of the diagonal matrix $\mathbf{D}$ of basis field eigenvalues

[1] Note that the meaning of our projection step is unrelated to the projection step in the Stam's stable fluids algorithm [30]. In Stam's work, projection was performed in order to enforce the zero divergence constraint, and required the solution of a linear system. It did not preserve energy. In our case, all possible velocity fields are guaranteed to be divergence free. The projection step here is instead to preserve energy, and requires only a trivial calculation.

[2] Square bracket notation is used to denote a discrete time sequence $\mathbf{w}_k[t]$ as opposed to a continuous function $\mathbf{w}_k(t)$.

$$\mathbf{w}_D = \exp(-\nu \Delta t \mathbf{D})\mathbf{w}.$$

Combining these steps, our integration scheme is:

1. Calculate current velocity direction and follow it for a small time interval

$$\frac{\partial \mathbf{w}_k}{\partial t} = \mathbf{w}^T \mathbf{C}_k \mathbf{w}$$
$$\mathbf{w}_k = \mathbf{w}_k[t] + \Delta t \frac{\partial \mathbf{w}_k}{\partial t}[t],$$

2. Renormalize to preserve energy:

$$\mathbf{w}_N = \frac{1}{|\mathbf{w}_a|}\mathbf{w},$$

3. Account for viscosity by dissipating energy in each coefficient:

$$\mathbf{w}[t + \Delta t] = \exp(-\nu \Delta t \mathbf{D})\mathbf{w}_N.$$

5.3.2 Higher Order Explicit Methods

The procedure just described can be improved by altering the first step to use higher order schemes (such as a Runge Kutta method) or using higher order time derivatives to produce a Taylor expansion

$$\begin{aligned}\mathbf{w}_k[t + \Delta t] = \mathbf{w}_k[t] &+ \Delta t \left(\mathbf{w}[t]^T \mathbf{C}_k \mathbf{w}[t]\right) \\ &+ \frac{1}{2!}(\Delta t)^2 \left(\mathbf{w}[t] + \frac{\partial \mathbf{w}[t]}{\partial t}\right)^T \mathbf{C}_k \left(\mathbf{w}[t] + \frac{\partial \mathbf{w}[t]}{\partial t}\right) \\ &+ \frac{1}{3!}(\Delta t)^3 \left(\mathbf{w}[t] + 2\frac{\partial \mathbf{w}[t]}{\partial t} + \frac{\partial^2 \mathbf{w}[t]}{\partial t^2}\right)^T \mathbf{C}_k \left(\mathbf{w}[t] + 2\frac{\partial \mathbf{w}[t]}{\partial t} + \frac{\partial^2 \mathbf{w}[t]}{\partial t^2}\right) + \cdots .\end{aligned}$$

5.3.3 Time Reversible Verlet Integrator

Using the available expressions for the second time derivative, a Verlet integrator can be built that is fourth order accurate in Δt:

$$\mathbf{w}_k[t + \Delta t] = 2\mathbf{w}_k[t] - \mathbf{w}_k[t - \Delta t] + \frac{\partial^2 \mathbf{w}_k}{\partial t^2}(\Delta t)^2 .$$

This Verlet integrator is also symmetric, allowing time to be run in reverse direction. Note that the first time derivative $\frac{\partial \mathbf{w}_k}{\partial t}$ is not explicitly part of the formula, as it is approximated by the difference between the position vectors. Over time, the velocity approximation will become worse due to accumulating error in the integrator. This is in contrast to previous integrators where it is only the vorticity *position* (or state) vector that accumulates error since $\frac{\partial \mathbf{w}_k}{\partial t}$ is recalculated explicitly at each time step.

Error in velocity is the price that is paid for time reversibility. More accurate time reversible integrators could be derived by including additional even order time derivatives in the Verlet integrator.

5.3.4 Exponential Map

As an alternative to the 'step and project' scheme in explicit methods, it is also possible to derive an N dimensional rotation matrix that will constrain the motion to the constant energy sphere, approximating the true geodesic motion of the Euler fluid equations near the current state.

The Lie algebra of $SO(N)$, the N dimensional rotation group, can be represented as $N \times N$ skew symmetric matrices. An element $\mathbf{r}$ of this Lie algebra is uniquely identified by the current state $\mathbf{w}$ and tangent vector $\frac{\partial \mathbf{w}}{\partial t}$ through the relation

$$\mathbf{r}_{i,j} = \left(\mathbf{w} \wedge \frac{\partial \mathbf{w}}{\partial t} \right)_{i,j}$$

where the right hand side is the i^{th}, j^{th} component of the wedge product between the state and tangent vector. $\mathbf{r}$ is indeed antisymmetric due to properties of the wedge product. This element of the the Lie algebra can be mapped back to the group $SO(N)$ through the exponential map

$$\mathbf{w}[t + \Delta t] = \exp(\Delta t\, \mathbf{r})\, \mathbf{w}$$
$$\mathbf{w}[t + \Delta t] = \mathbf{R}(\Delta t)\mathbf{w}$$

where the rotation matrix $\mathbf{R}$ can be computed using a truncated matrix exponential series

$$\mathbf{R}(\Delta t) = \exp(\Delta t\, \mathbf{r}) = \sum_k^{\infty} \frac{1}{k!} t^k \mathbf{r}^k.$$

Since $\mathbf{R}$ is unitary, the resulting vector will have identical energy. This method is interesting 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 'step and project' method. However, its practical utility is limited as the computation is more expensive than an explicit method, and does not greatly improve accuracy for small timesteps when compared to an Euler method with reprojection. This method could be used in cases where the accuracy is highly valued regardless of computational expense.

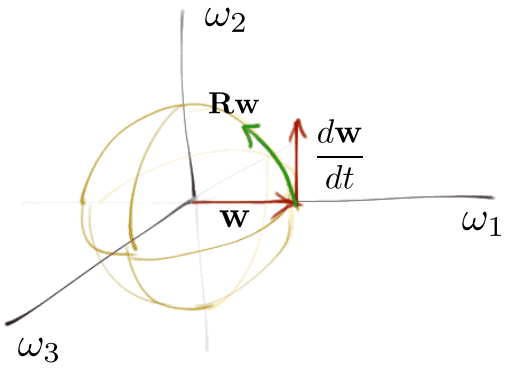


Figure 5.6: An N dimensional rotation matrix $\mathbf{R}$ can be calculated as a function of $\mathbf{w}$ and $\frac{\mathbf{dw}}{\mathbf{dt}}$ through the exponential map. Applied to the current state vector, R constrains motion exactly to a spherical surface of constant energy.

5.4 Dynamics of Advected Quantities

Our method produces a velocity field, which can be used to advect particles or densities in standard ways. However, for simple domains, we have analytic expressions for the velocity basis fields and their spatial derivatives. We have shown previously that we can obtain exact time derivatives for the change in the velocity field. Having both spatial and time derivatives gives us the tools to derive unique advection schemes.

To illustrate, consider the path taken by a massless particle immersed in a 2-D fluid. Denote the particle's position by $p(t)$. The particle's velocity is $p'(t) = u(p(t), t)$ which we write in coordinate form in 2-D as:

$$p'_x(t) = u_x(p_x(t), p_y(t), t)$$
$$p'_y(t) = u_y(p_x(t), p_y(t), t).$$

Note that u is the time varying fluid velocity field, making it a function of t as well as the particle's position. The particle's acceleration is the time derivative of this expression. Since both the particle's position $p(t)$ and the velocity field u vary with time, we use the chain rule to obtain the total derivative. In 2-D coordinates, this yields:

$$p''_x(t) = \frac{\partial u_x}{\partial p_x}\frac{\partial p_x}{\partial t} + \frac{\partial u_x}{\partial p_y}\frac{\partial p_y}{\partial t} + \frac{\partial u_x}{\partial t}$$
$$p''_y(t) = \frac{\partial u_y}{\partial p_x}\frac{\partial p_x}{\partial t} + \frac{\partial u_y}{\partial p_y}\frac{\partial p_y}{\partial t} + \frac{\partial u_y}{\partial t}.$$

All of these terms are available to us. $\frac{\partial p_x}{\partial t}$ and $\frac{\partial p_x}{\partial t}$ are just the velocity vectors at a point. $\frac{\partial u_x}{\partial p_x}$ are the spatial derivatives of the velocity field, which can be obtained by differentiating analytic basis functions. The time derivative terms $\frac{\partial u_x}{\partial t}$ are obtained by first calculating time derivative in the vorticity basis $\frac{\partial \mathbf{w}_k}{\partial t}$ as outlined in 5.3 and then expanding in the velocity basis.

Using a second order Taylor expansion as an example, we can derive a time integration scheme for particle advection as

$$p[t + \Delta t] = p[t] + p'(t)\Delta t + \frac{1}{2}p''(t)(\Delta t)^2.$$

Note that through the $p''(t)$ term, not only does this scheme take into account acceleration information of the particle, but incorporates the time variation of the velocity field itself. Previous fluid simulation techniques in graphics always alternate between update of the velocity field and particles/density, treating the velocity field as static during advection.

The example illustrated is for a second order Taylor expansion, but the same procedure could be applied to obtain an expansion of arbitrary order. We have shown in Section 5.3 how to obtain arbitrary time derivatives of the velocity field. Assuming the functions are smooth, we can continue to differentiate analytic basis functions spatially. Mixed partials can be handled by first calculating the time derivative in the vorticity basis as outlined, and then expanding in spatial derivatives of the velocity field.

Symmetric Integrators for Particles

A symmetric, time reversible Verlet integrator was described for time integration of the vorticity. The same may be applied to particles transported by the fluid

$$p[t + \Delta t] = 2p[t] - p[t - \Delta t] + p''(t)(\Delta t)^2.$$

As previously noted, a Verlet integrator does not explicitly calculate velocity, but rather approximates it through current and past positions and updates it based on the acceleration. The approximated velocity will drift from the actual velocity as error accumulates. Visually, this becomes very noticeable as the particle's velocity will no longer correspond to the underlying fluid velocity field, resulting in motion that is no longer divergence free.

The integrator remains time reversible precisely because the velocity is encoded in consecutive particle positions. Hence, accumulation error in velocity is the price that is paid for time reversibility. This can be mitigated by using small time steps or using higher order Verlet integrators, taking into account even order time derivatives of the particle's position.

5.5 Discrete Meshes for Irregular Geometries

Until now, our formulation has been entirely analytic. This is possible under simple geometries, in which analytic expressions for Laplacian eigenfunctions are available. However, to describe flows on arbitrary geometries, we must resort to a discrete numerical approach to approximate the basis functions and their interactions. As meshes are a common representation of geometry in computer graphics, we aim to reformulate our algorithm to work with arbitrary discrete meshes, while retaining the desirable structure and properties from the analytic case.

Differential operators can be discretized (approximated) in a variety of ways. Finite differences are a simple means to do so, but offer no guarantee of preserving the inherent geometric structure of the physical model. Instead, we opt instead to employ *discrete exterior calculus* (DEC) [9, 16]. *Exterior calculus* (EC) is the modern language of differential geometry: a calculus of continuous manifolds. It defines continuous operators such as the exterior derivative and the Hodge star which operate on differential forms defined over smooth manifolds. DEC is an extension of EC to discrete domains. Instead of manifolds, quantities are represented on *simplicial meshes*. Discrete operators are defined that share many of the important properties of their continuous counterparts. As a result, DEC has been shown to preserve geometric structure of the continuous models in many problem areas, including electromagnetism and fluid dynamics [**?**, 22]. It is a natural choice for transitioning our model to the discrete case. For a detailed introduction to DEC, see [9]. For DEC applied to fluid simulation in computer graphics, see the previous work on energy preserving fluids [22] and circulation preserving simplicial fluids [**?**].

We will begin by modelling fluid quantities and operators using EC and DEC in similar fashion to the approaches in previous work [**?**, 22]. Next, we show how these tools can be used in the context of our method. Namely, we show how to obtain basis fields that are eigenfunctions of a discrete Laplacian operator, and how to discretize the advection operator to compute the entries of the $\mathbf{C}_k$ matrices.

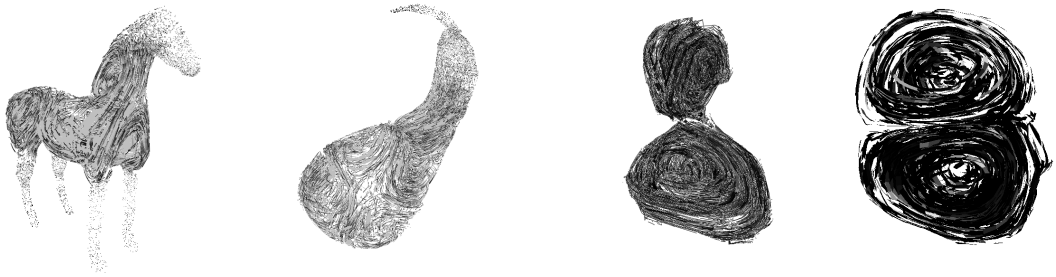


Figure 5.7: Examples of basis fields on 3D irregular tetrahedral meshes.

5.5.1 Fluid Quantities as Differential Forms

Continuous differential forms are quantities that can be integrated. Discrete differential forms are represented by their integral over a discrete domain. Instead of representing velocity as a vector field, we instead store the integral of velocity flux across the boundary. In 2-D, that boundary is an edge, so velocity is a 1-form. In 3-D, velocity flux is integrated over faces, making velocity a 2-form. In this fashion, differential forms are simply values attached to

edges and faces. No additional information needs to be stored, as the direction of the velocity is defined intrinsically through the geometry of the mesh itself. As vorticity is orthogonal to velocity, it lives naturally on the dual mesh, and is similarly represented as an integrated quantity over faces. Table 5.1 summarizes the differential forms.

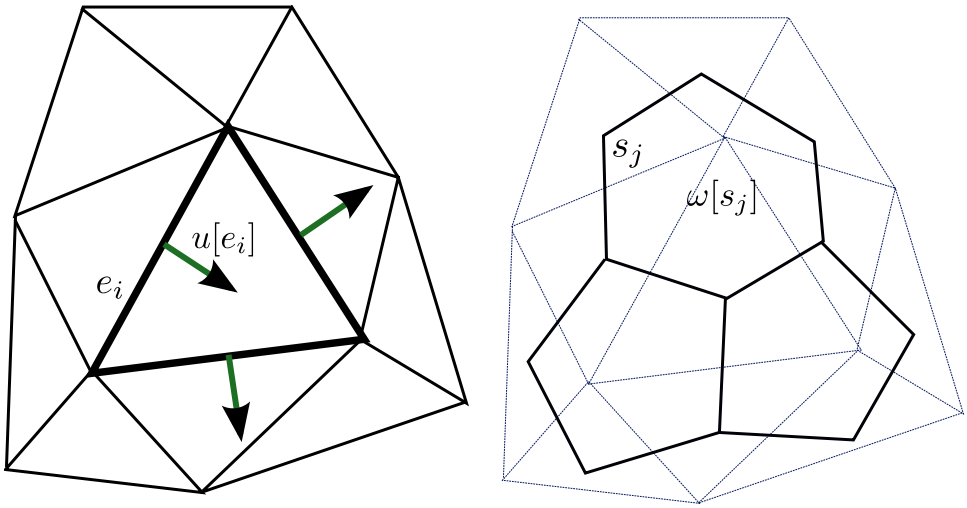


Figure 5.8: Velocity and vorticity as differential forms on a 2-D simplicial mesh. Velocity exists on edges, representing the flux across the edge boundary. Vorticity lives on dual faces, representing the flux through the face.

Table 5.1: Fluid quantities in DEC

Quantity	**Symbol**	**2-D**	**3-D**
Vorticity	ω	Dual 2-form	Dual 2-form
Velocity	u	Primary 1-form	Primary 2-form

5.5.2 DEC Operators and their Implementation

Here we describe exterior calculus operators and how their discrete equivalents can be implemented as matrices or algorithms. Table 5.2 summarizes a relation between traditional vector calculus operators and their DEC equivalents.

Hodge Star Operator In n dimensional space, the discrete Hodge star operator $*$ maps a primal k form to a dual $(n-k)$ form. The *diagonal Hodge star operator* is one possible implementation of this mapping defined as the ratio between the primal and dual volume elements.

Exterior Derivative Operator The continuous exterior derivative operator satisfies Stokes theorem:

Table 5.2: DEC Operator equivalents

Description	Vector Calculus	DEC
Curl of a vector field	$\omega = \mathrm{curl}\,(u)$	$\omega = d * u$
Divergence of a vector field	$\mathrm{div}\,(u)$	du
Cross Product	$u \times \omega$	$\omega = u \wedge \omega$
Laplacian	$\omega = \mathrm{curl}^{\,2}\psi = \Delta\psi$	$\omega = d * d * \psi$

$$\int_c du = \int_{\partial c} u$$

which says that for a k-form u and a domain c, du evaluated over a simplex is equal to u evaluated on the boundary of c. The discrete exterior derivative operator is designed to satisfy a similar discrete version of Stokes theorem. Essentially, the domain c becomes a simplex, and u a k-form represented by quantities on the boundary simplicial elements of c. To determine du, we sum the k-form quantities around the boundary of each $k+1$ simplex (being careful to account for orientation) and store the result on $k+1$ simplicies.

Divergence When applied to an $n-1$ form in an n dimensional space, the divergence operator is just d, the exterior derivative. For example, in 2-D dimensions it sums the fluxes through the edges (1-forms) to calculate the divergence of each triangle (2-form). In 3-D, it sums fluxes of faces to calculate the divergence of each tetrahedron. This makes calculating the divergence of a velocity field in DEC very simple, as u is represented as integrated velocity flux on edges or faces, and du amounts to summing the fluxes on the boundary of a volume cell.

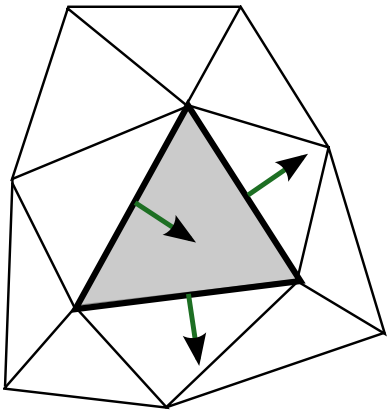


Figure 5.9: The divergence du of a triangle is calculated by summing the fluxes of the surrounding edges.

Curl The curl operator is a combination of preceding operators, defined as $d*$ in DEC. We illustrate its implementation on a 2-D simplicial mesh. Since $\omega = \mathrm{curl}\,u$, we expect $d * u$ to

produce a dual 2 form, a value stored on dual faces. As depicted in Figure 5.10a, for each dual face, we consider the adjacent primal edges $e_1, \ldots, e_6$ and scale their flux values by the length of dual edges. This is the action of the $*$ operator. Next, we sum these values and assign the result to the dual face s_i.

We also saw that for Laplacian eigenfunctions, we can recover the velocity through $u = \frac{1}{\lambda}\text{curl}\,\omega$; thus applying $d*$ to ω should yield a 1-form. As depicted in Figure 5.10b, for each primal edge, we consider the two adjacent dual faces and scale their vorticity values by the area of each face. This is the action of the Hodge star $*$ operator. Then we take the oriented sum of these values and assign the result to the primal edge.

The described procedures are straightforward operations. $d*$ applied to a primal 1-form and $d*$ applied to a dual 2-form can each be represented by a matrix acting on a column vector of form values. Hence for a given mesh, these matrix operators can be precomputed. If all ω and u values over the mesh are represented as column vectors $\mathbf{u}$ and $\mathbf{w}$ respectively, then the $d*$ operators can be conveniently represented as sparse matrices $\mathbf{M}_{\omega,u}$ and $\mathbf{M}_{u,\omega}$

$$\omega = d * u$$
$$\mathbf{w} = \mathbf{M}_{\omega,u}\mathbf{u}$$

$$u = \frac{1}{\lambda} d * \omega$$
$$\mathbf{u} = \mathbf{M}_{u,\omega}\mathbf{w}.$$

Note that multiplication by $\mathbf{M}_{\omega,u}$ or $\mathbf{M}_{u,\omega}$ represents global action of the operator on the *entire* mesh.

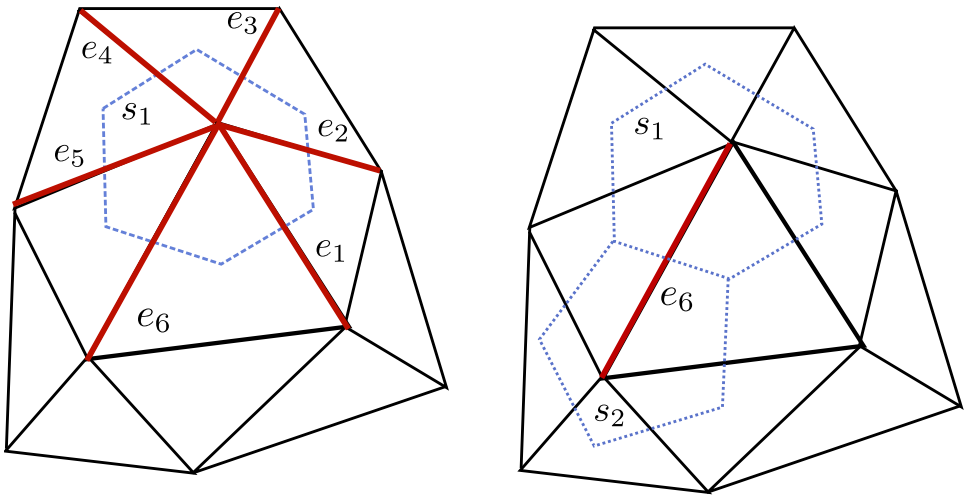


Figure 5.10: Calculating $\omega = d * u$ and $u = \frac{1}{\lambda} d * \omega$ on a 2-D simplicial mesh.

Laplacian The Laplace operator in exterior calculus is defined as

$$\Delta = \delta d + d\delta = *d * d + d * d * .$$

Since the divergence of the velocity and vorticity fields is zero, $d\omega = du = 0$ and the first term in the definition vanishes. Hence for divergence free fields, the Laplacian operator is just $d * d*$, a product of the two $d*$ matrix operators introduced above:

$$\omega = \lambda d * d * \omega$$
$$\mathbf{w} = \lambda \mathbf{M}_{\omega,u} \mathbf{M}_{u,\omega} \mathbf{w}.$$

Note that if ϕ_k is a Laplacian eigenfunction then $\Delta\phi_k = d * d * \phi_k = \lambda_k \phi_k$, and $\Phi_k = \text{curl}^{-1}\phi_k = \frac{1}{\lambda_k} d * \phi_k$ as in the continuous case.

Wedge Product The wedge product is required for computing the advection operator between basis fields. In the most general sense the wedge product maps a k and n form to a $k + n$ form. The discrete formulation of the wedge product is rather complicated. For our purposes, we consider only wedge products in three dimensions, which is equivalent to the traditional vector cross product. The underlying concept is that we evaluate the flux through a dual face of the cross product $u_i \times u_j$.

On an irregular mesh, the cross product can be approximated as follows. Consider Figure 5.11. Edges are represented by e_i. Unit vectors normal to edge e_i are denoted by n_i, satisfying $n_i \cdot e_i = 0$ and $|n_i| = 1$. The support area $a_{i,j}$ is the area of the quadrilateral formed by the centroid of the triangular face and the midpoints of edges e_i, e_j. The union of all $a_{i,j}$ is the dual face s. The velocity 1-form assigned to each primal edge is denoted by $u_a[e_i]$ or $u_b[e_i]$.

To evaluate the flux of $u_b \times u_a$ through s, consider every pair of adjacent edges e_i, e_j emanating from a vertex and evaluate the sum:

$$\sum a_{i,j} \left(u_a[e_i] u_b[e_j] - u_b[e_i] u_a[e_j] \right) (n_i \times n_j)$$

Essentially, this is a low order approximation to the cross product weighted by the support area subtended by each pair of edges. The same procedure applies in three dimensions, except we consider every adjacent pair of primal faces to a primal edge.

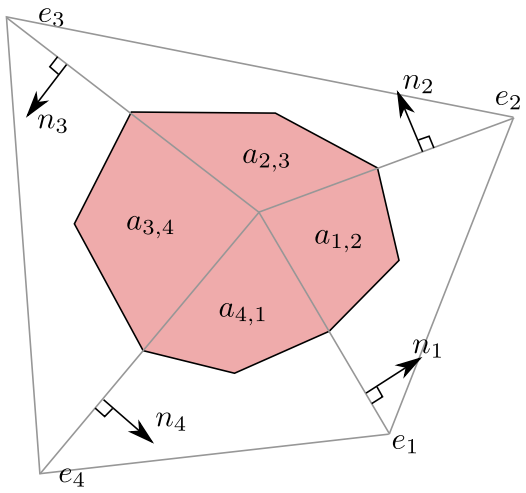


Figure 5.11: Evaluation of advection operator of 2-D simplicial mesh

5.5.3 Discrete Laplacian Eigenfunctions

As previously explained, we adopt the following discrete Laplace operator for divergence free fields:

$$\Delta = d * d * .$$

For a given simplicial mesh, this operator is represented as a sparse symmetric matrix obtained by multiplication of the matrix representations of the d and Hodge star $*$ operators. Computing the basis fields amounts to finding the eigenvectors of this matrix. Boundary conditions are enforceable by explicitly setting entries in the operator matrix. For example, to enforce a free-slip velocity boundary condition, we omit (set to zero) the rows of the first d operator that calculate vorticity as a function of surrounding edges (or in the 3-D case, faces.)

Choice of Eigensolver Many techniques exist to compute the eigendecomposition of a matrix. Our choice of eigensolver is motived by the properties of our matrix and the practical requirements of our setting. The discretized Laplace operator is real symmetric and sparse. We require a method that works well with sparse representations, as the storage requirements for dense representations would be prohibitive for large meshes. Similarly, computing a complete set of eigenvalues for a large mesh would be very expensive. For interactive fluid simulations, our method will typically be used with a moderate number of basis functions. Hence, we require a method that can calculate only a subset of the eigenvalues, ordered from smallest to largest.

A good candidate for these requirements is the Arnoldi method implemented in ARPACK. This is an iterative method making it suitable for sparse matrix representations. This method can also return a small number of eigenvalues near an desired point in the spectrum. To obtain an ordered set of positive eigenvalues, we begin by searching near zero, and gradually increase the search point until the desired number of eigenvectors have been computed.

Note that the eigensolver will converge very rapidly if the initial search point is very near an eigenvalue. Hence, once a set of eigenvectors has been computed, it suffices to store only the *eigenvalues* for future program runs, as these can be used as initial search points for the eigensolver, greatly accelerating eigenvector computation.

A further consideration in the choice of solver is the qualitative difference in the orthogonal basis fields. Just as an orthogonal basis can be oriented arbitrarily in $\mathbb{R}^3$, the orthogonal velocity basis fields sets will vary from solver to solver. Some have produced sets of beautiful patterns, much different than the boxed vortex shapes of the Fourier Sine basis calculated analytically.

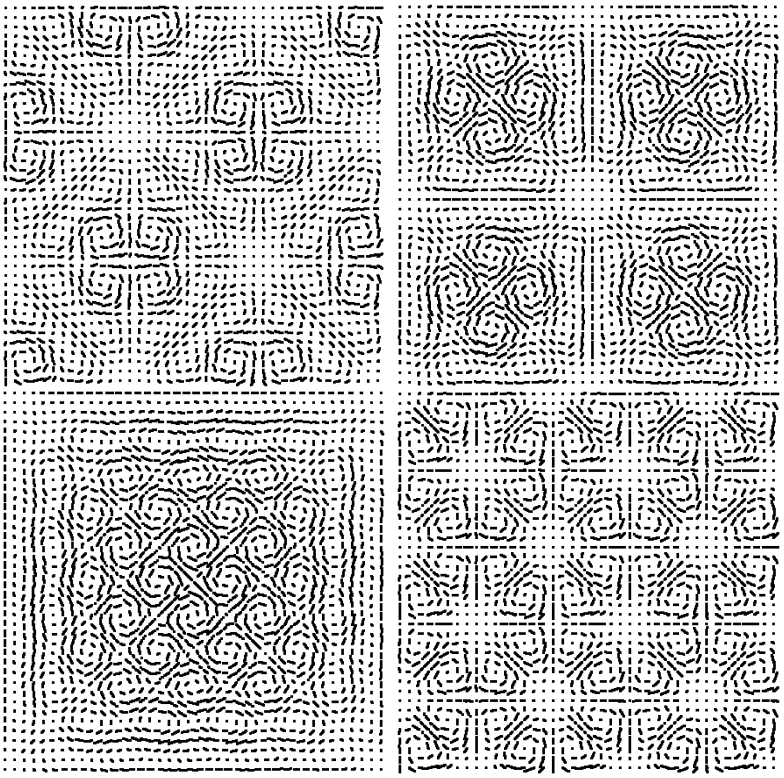


Figure 5.12: Examples of orthogonal basis fields produced by a Jacobi Davison eigensolver. Compared to the Fourier Sine basis, these basis fields are more irregular and qualitatively interesting.

5.5.4 Discrete Advection Operator

We repeat a derivation similar to Chapter 5 reformulated in exterior calculus, and explain how the wedge product implementation of Section 5.5.2 can be used to compute it for pairs of basis fields.

We restate the Helmholtz vorticity formulation of the Euler equations:

$$\frac{\partial \omega}{\partial t} = \mathcal{L}_u \omega.$$

The Lie derivative can be expressed in exterior calculus using Cartan's formula:

$$\mathcal{L}_u \omega = i_u d\omega + d i_u \omega.$$

where i_u is the *contraction operator* defined in exterior calculus as $i_a b = *(*a \wedge b)$. Using this definition, we can expand the expression to

$$\mathcal{L}_u \omega = *(*d\omega \wedge u) + d * (*\omega \wedge u).$$

The first term $i_u d\omega$ vanishes since ω is a divergence free field and $d\omega = 0$. The second term can be simplified by distributing the d and $*$ operators over the wedge product and using the Leibniz rule to obtain

$$\mathcal{L}_u \omega = u \wedge d * \omega.$$

Adopting the bracket notation and considering u and ω as Laplacian eigenfunctions we have

$$\begin{aligned}[\Phi_i, \phi_j] &= \Phi_i \wedge d * \phi_j \\ &= \Phi_i \wedge \lambda_j \Phi_j \\ &= \lambda_j \left(\Phi_i \wedge \Phi_j\right).\end{aligned}$$

since $d * \phi_j = \operatorname{curl} \phi_j = \lambda_j \Phi_j$. This is analagous to the $\lambda_j(\Phi_i \times \Phi_j)$ term in Equation 5.11. This gives us some confidence as derivations in traditional notation and in exterior calculus agree.

For every pair of velocity basis fields, we can now evaluate the bracket using the discrete wedge product operator described previously in Section 5.2. The resulting coefficients form the $\mathbf{C}_k$ matrices and satisfy

$$\lambda_i(\Phi_j \times \Phi_i) = \sum_k \mathbf{C}_k[i, j]\Phi_k. \tag{5.14}$$

Comparison to Analytic Advection Operator The wedge product implementation we have described is only a first order approximation, since it assumes the fields to be piece-wise constant over the support volumes. The approximation error incurred requires consideration. Two basis fields whose wedge product is exactly zero in the analytic case may produce a small non zero value when calculated numerically. The result is that we do not obtain exact sparseness in the projected coefficients. Instead, some of the coefficients will be relatively larger than the average, but all will have a non-zero component.

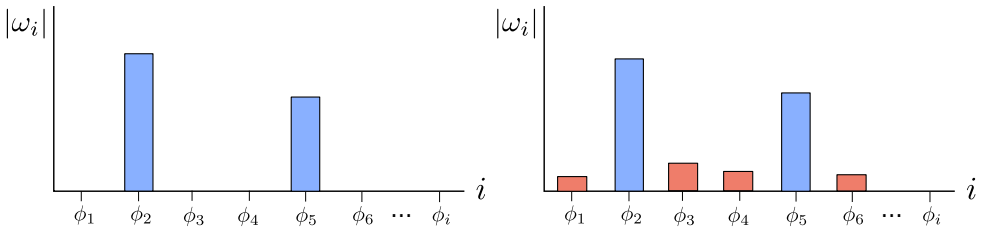


Figure 5.13: A figurative comparison of the spectrum of the advection operator. Left: When performed analytically, the result is exactly sparse. Right: Error in the discrete approximation of the advection operator produces a spectrum that resembles the analytic case, but is not perfectly sparse.

This is problematic for our simulation method for two reasons. First, the $\mathbf{C}_k$ will no longer be sparse, increasing the memory and computing expense. Second, and more importantly, the small but inaccurate non-zero entries cause interaction between basis fields that should not occur. The effect is a type of diffusion. Energy tends to spread out quickly among all basis components, smoothing out pronounced variations. Visually, this manifests itself as a vorticity field that does not change as much as it should.

A simple mitigation strategy is to apply a threshold to the spectrum in order to keep the peaks and eliminate the small non zero values. Choosing the threshold is not always straightforward, as the difference between peaks and the noise floor depends on the resolution and quality of the mesh. Currently we employ a strategy that sets the threshold to keep n of the the largest values, rejecting the remaining ones. The parameter n can be chosen by a user, with typical values between 2 and 5 as this is the sparseness observed in analytic cases.

A better solution to the problem would be to use a higher order approximation of the wedge product, or develop a discrete wedge product that preserves sparseness.

Chapter 6

Results

In this section we briefly describe our implementation, and then present performance tables and visual results of our algorithm.

6.1 Implementation Notes

We have implemented our fluid simulation technique in a combination of Python and C++. Python is a dynamically-typed interpreted scripting language that allows for fast and flexible development. It allows custom C++ extensions, and bindings exist for numerous compiled shared libraries. We make use of bindings to OpenGL for drawing primitives, ARPACK for eigendecomposition, and the Numpy/Scipy numerical libraries for matrix operations.

At its core, our algorithm performs sparse matrix vector multiplication to calculate $\frac{\partial \mathbf{w}_k}{\partial t} = \mathbf{w}^T \mathbf{C}_k \mathbf{w}$. We store $\mathbf{C}_k$ in compressed sparse row format (CSR) format, allowing efficient computation of matrix vector products, and making the storage footprint very small, as only 1%-4% of the entries are nonzero. Additionally, each of the k components of $\frac{\partial \mathbf{w}_k}{\partial t}$ can be calculated independently, making the algorithm inherently parallel. However, in our current implementation we do not parallelize computation.

6.2 Results

In this section we present performance tables, figures, still images and animations demonstrating our method and highlighting its unique features. Where appropriate, we choose to use 2-D fluid simulations, as they often best illustrate the point. We also present some full color ray traced animations for 3-D simulations, and a creative performance based fluid animation. Still images from animations are included inline. Video files are available at http://www.dgp.toronto.edu/people/tyler/msc.

The simplest case A minimum of three basis functions is needed to observe interesting dynamic behavior. Figure 6.1 shows a vortex moving clockwise around the domain as it is carried by the flow of a larger vortex. This confirms what is described mathematically by the Helmholtz version of the Euler equations: the vorticity is advected by the flow. Observing the basis coefficients, we can see that energy oscillates between the 2nd and 3rd basis functions while the 1st basis function remains constant.

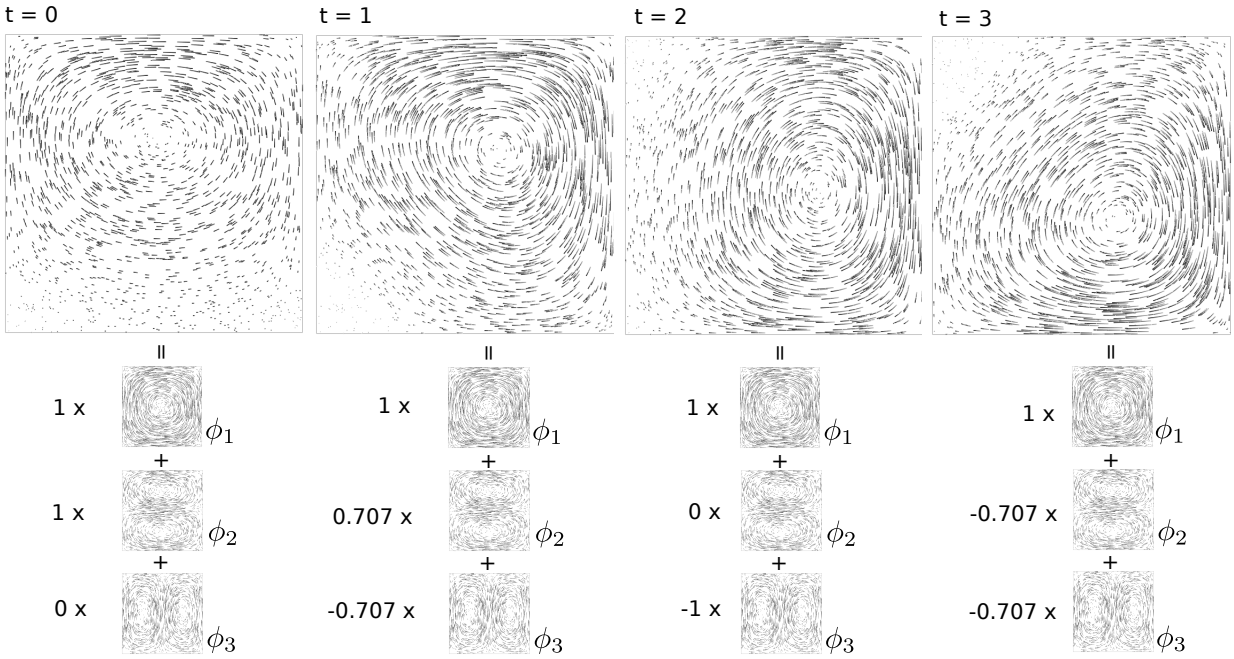


Figure 6.1: Vorticity is advected by the flow. Using only three basis functions, we see the smaller vortices advected clockwise by the large vortex. The basis field coefficients are shown at each timestep.

Effect of number of basis fields An infinite number of basis fields is required to perfectly describe fluid motion, but we must choose finitely many of them for computation. Basis fields with larger eigenvalues produce fields with higher 'frequency', which in our context corresponds to fields with smaller and smaller vortices. We choose to truncate the spectrum keeping all basis fields with eigenvalue smaller than a certain value. Hence a larger number of basis fields are needed to capture the motion of small vortices. This is a reasonable way to perform truncation, since at some point vortices of a small enough scale would not be observable in nature for even very small viscosities.

Figure 6.2 shows snapshots from flows simulated with varying number of basis fields chosen in this fashion. Note that even for low dimensional bases, the flows produced are still smooth and the center of the vortices can still be tight; the dimensionality only limits the minimum scale of a vortex.

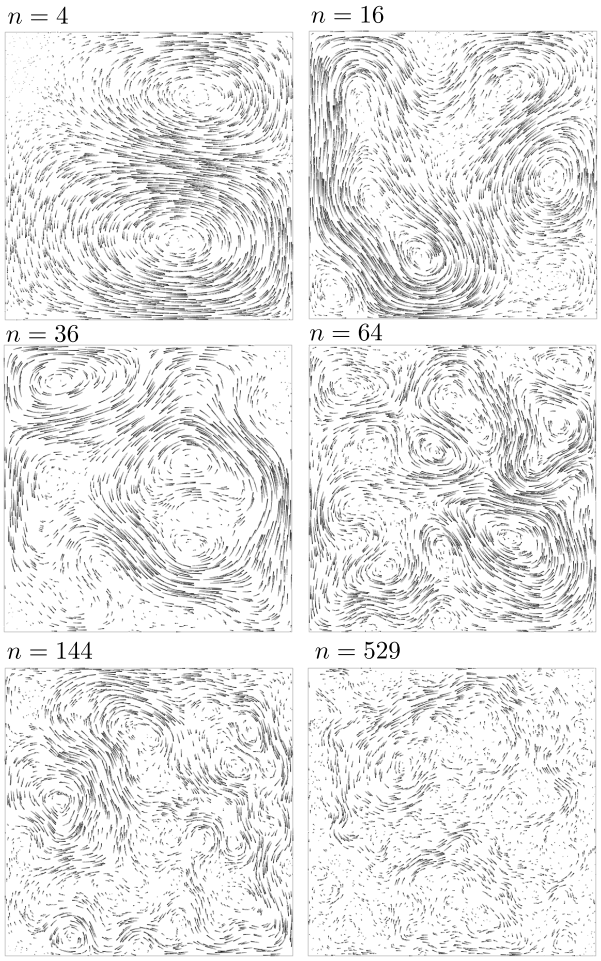


Figure 6.2: Left to right, top to bottom: Snapshots from flows simulated with 4, 16, 36, 64, 144 and 529 basis functions respectively. More basis fields permit simulation of smaller scale vortices.

Analyticity prevents grid artifacts. Using analytic expressions for basis fields allows them to be evaluated to numerical precision at any spatial coordinate. Figure 6.3 shows the absence of observable grid artifacts when analytic basis functions are employed. For comparison, the example on the left of the figure stores its the velocity field is on a MAC (Marker and cell) grid and linearly interpolates, as is often used in existing Eulerian fluid simulations.

A further advantage of analytic basis fields is that they do not need to be stored in memory. Velocity is computed on the fly as a function of the particle positions. This could be advantageous in parallel architectures where performance bottlenecks are often related to memory access. It also makes rendering cost proportional to the number of particles.

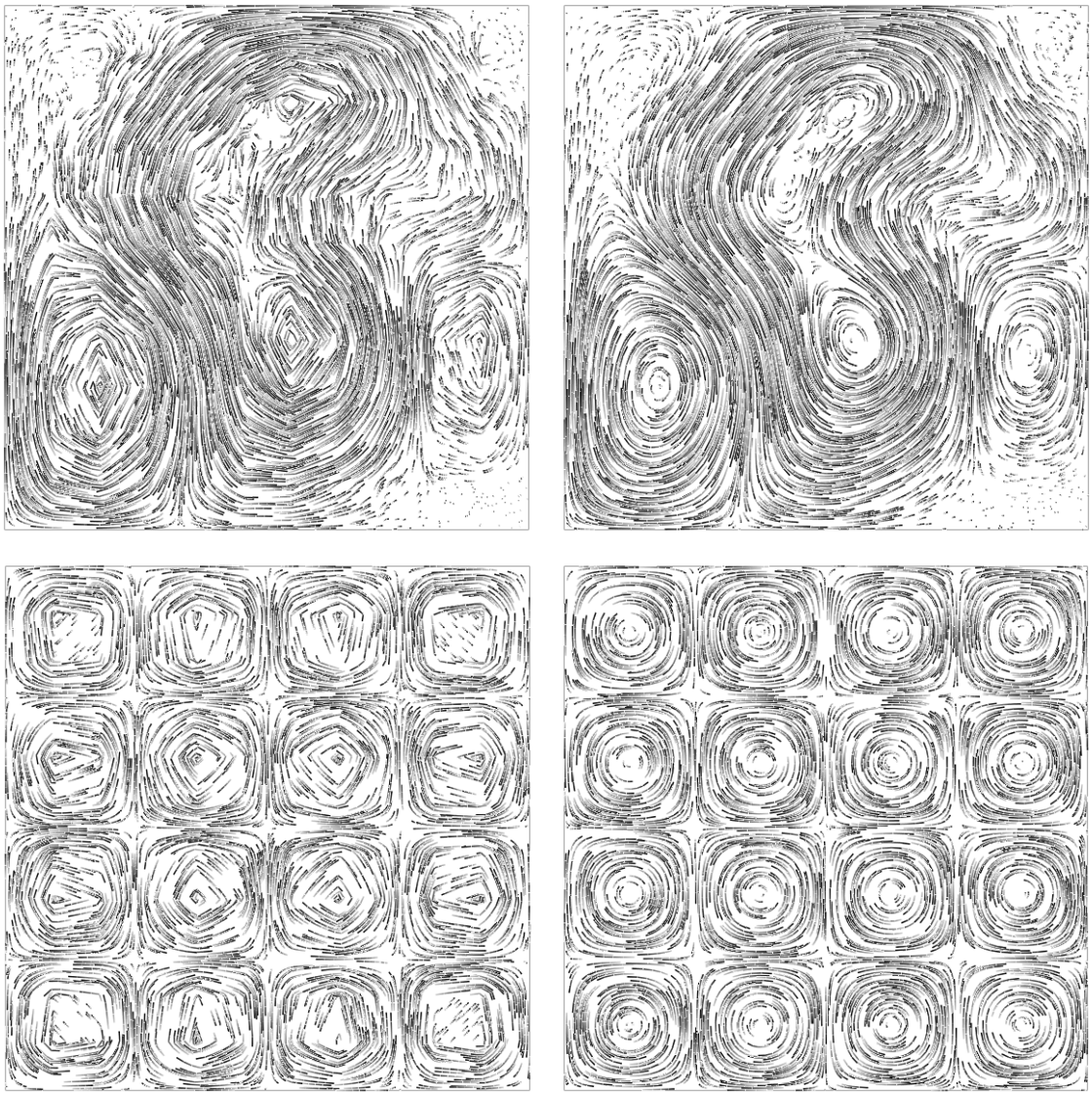


Figure 6.3: Top left, bottom left: fluid simulation on 16x16 MAC grid with linear interpolation of velocity field. Grid artifacts are most noticeable at the centers of vortices where the curvature is high. Top right, bottom right: fluid simulation using analytic basis functions for velocity field. The velocity field remains smooth and accurate in the areas of high curvature.

Controllable Viscosity In Chapter 5, we showed how to accurately simulate physical viscosity by decaying the energy in basis fields with a time constant proportional to their eigenvalue. Accounting for viscosity in our model is trivial and does not require precomputation. It can be controlled at run time by changing a single parameter.

Figure 6.4 shows storyboards of fluid animations for varying viscosity. The top row is a simulation with zero viscosity. As no energy is dissipated, this animation will continue perpetually. The second row is for a kinematic viscosity ν equal to that of water. The 3rd row for a viscosity roughly 30 times greater, comparable to that of corn oil.

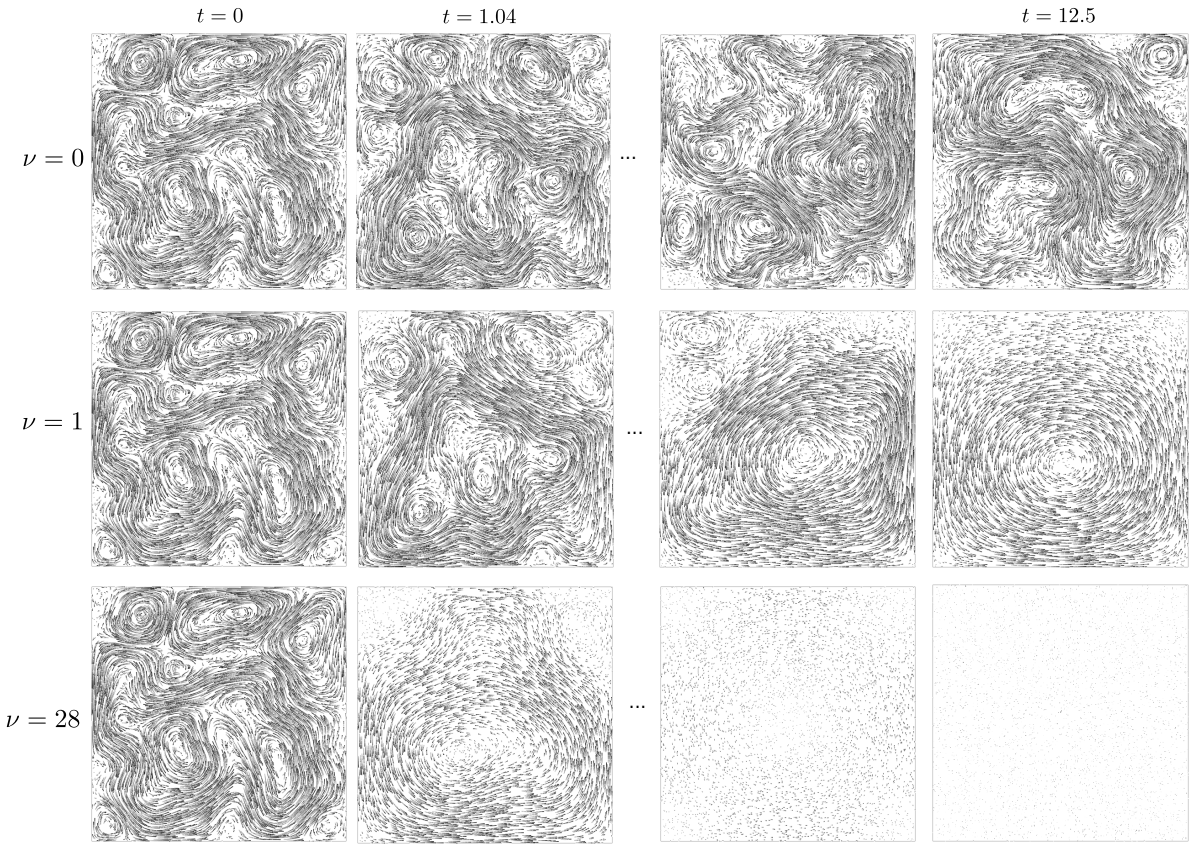


Figure 6.4: Viscosity is easily controllable in our model. Each row corresponds to a storyboard with a different viscosity. From top to bottom: $\nu = 0$ (inviscid), $\nu = 0.001$ (water), $\nu = 0.030$ (vegetable oil). Units for ν are $\frac{m^2}{s} \times 10^{-6}$.

Analytic 3-D cubic domain Analytic basis functions and their interactions have been derived for a 3-D cubic domain. This preserves all properties of analyticity previously discussed in the 2-D case, such as the lack of grid artifacts. Figure 6.5 shows some examples of individual basis fields. They resemble those of the 2-D rectangle, but are repeated along each of three axes. Figure 6.6 shows an animation on a 3D domain using analytic basis functions.

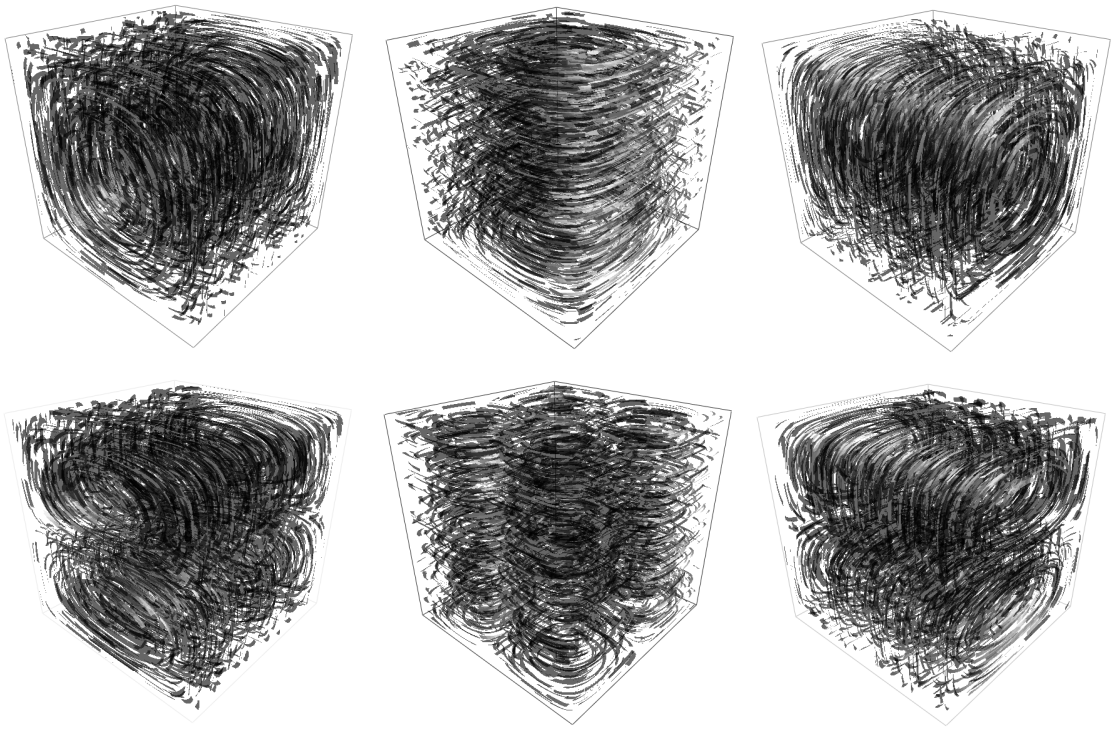


Figure 6.5: 3D analytic basis fields.

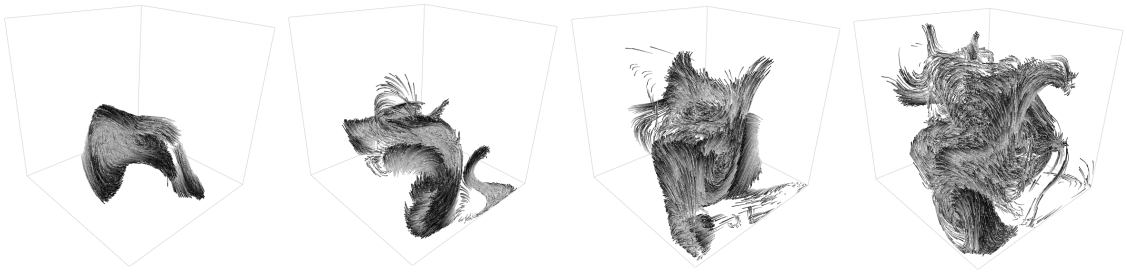


Figure 6.6: Fluid simulation on a cubic domain using 376 basis functions.

2-D Simplicial Mesh Using DEC, we extended our algorithm to work with triangular meshes. The benefits of analyticity are lost, but allows for complex boundary conditions and immersed obstacles. Figure 6.7 shows examples of flows on 2-D triangular meshes,

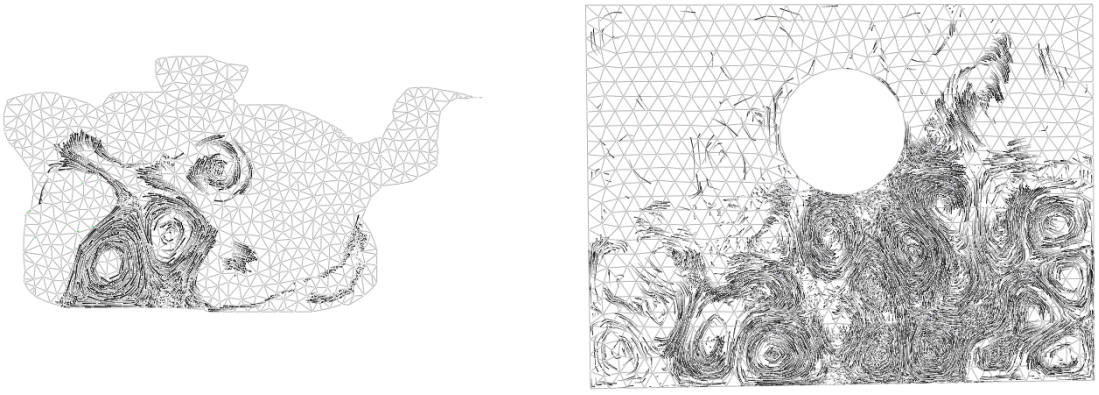


Figure 6.7: Fluid simulation on 2D irregular triangular mesh. Left: a complex boundary. Right: an immersed obstacle.

3-D Simplicial Mesh Analogously, we also used DEC to extend our algorithm to work with tetrahedral meshes. Figures 6.8 - 6.10 show examples of individual basis fields calculated numerically on complex geometries. Figures 6.11, 6.12 shows snapshots from animations.

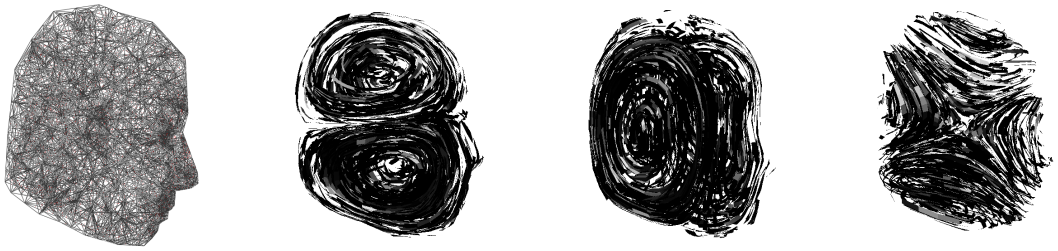


Figure 6.8: Basis fields for 3D tetrahedral model of a head.

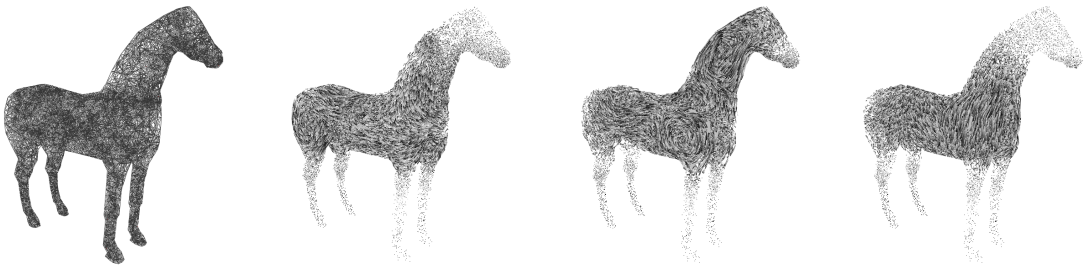


Figure 6.9: Basis fields for 3D tetrahedral model of a horse.

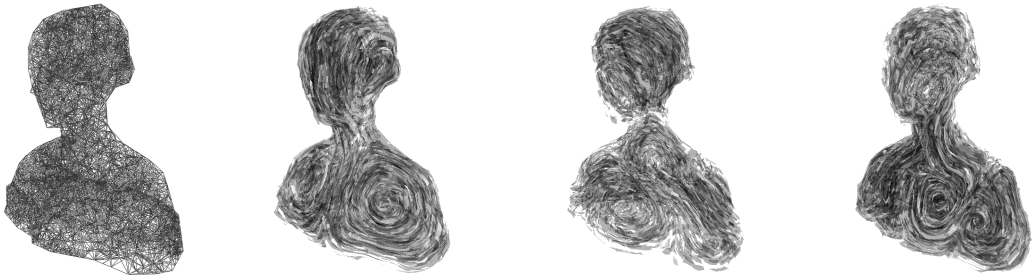


Figure 6.10: Basis fields for 3D tetrahedral model of a bust.

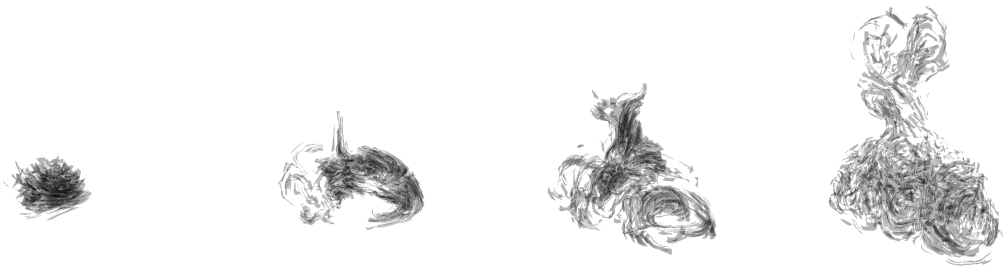


Figure 6.11: Animation within bust model.

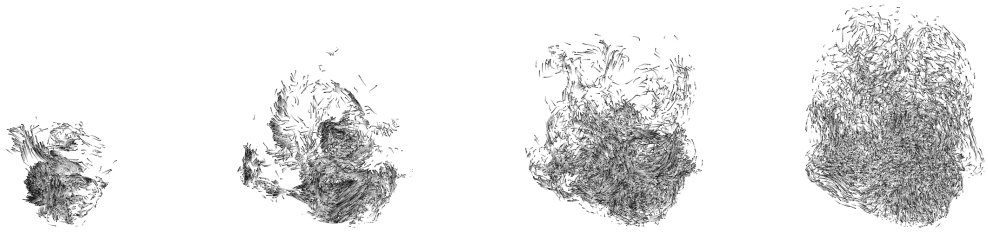


Figure 6.12: Animation within head model.

Smoke simulation Figure 6.13 shows images rendered through volumetric ray tracing of density fields created by radial basis functions at advected particle positions. Not counting rendering costs, in both of these examples each simulation step took approximately 10ms, after a precomputation of less than 10 minutes.

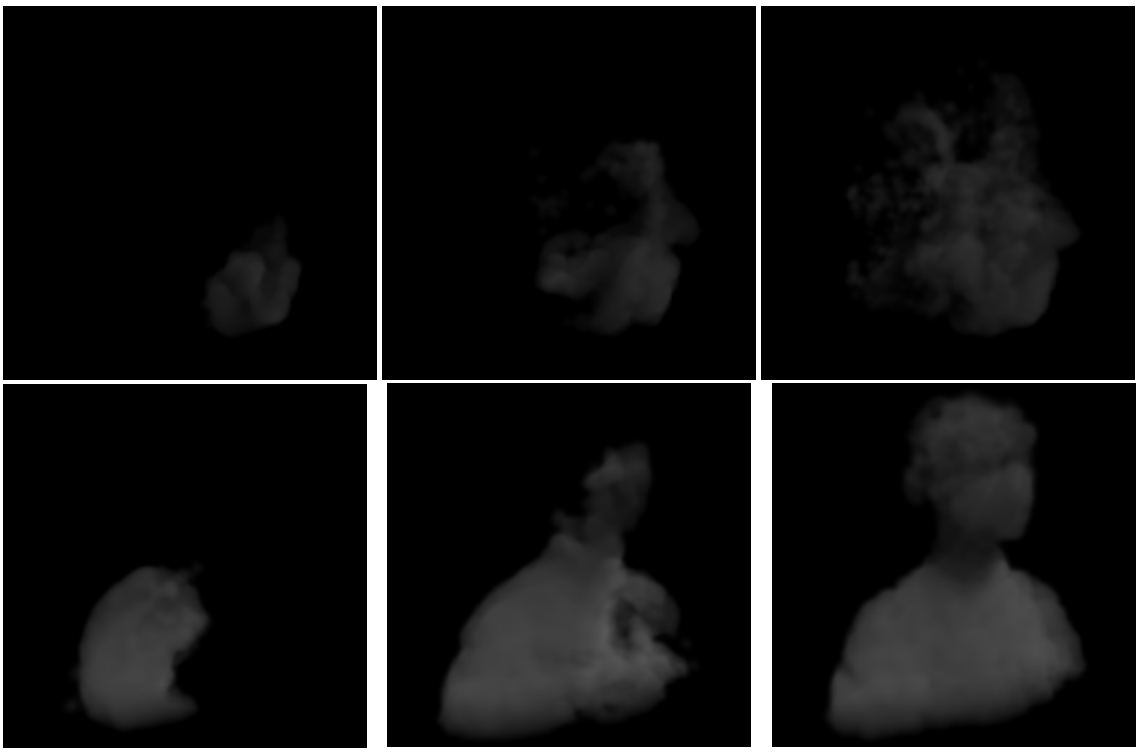


Figure 6.13: Smoke filling a head(top row) and bust(bottom row)

Snapshots from Artistic Animation Some basis fields exhibit symmetry in one or more spatial directions. The dynamics amongst these basis fields will produce fluid motion that remains symmetric. By initializing an inviscid flow with a few of these basis functions, the resulting dynamics will change perpetually, but always retain an element of symmetry.

We use this property along with a few liberal adjustments to design an artistic fluid simulation. For example, one modification we employed is to clamp particle positions to a 'margin' around the boundary of the fluid simulation domain. Particles will tend to accumulate into dense layers, which are continually accumulated and lifted off, creating a pleasing visualization of the vortices in the field.

Additionally, we allow an adjustable ratio between particle velocity and the rate at which dynamics progress. This allows fine tuning of the degree to which particles 'trace out' the vortices. A large ratio allows the particles to move quickly through the vortices revealing their structure, while the actual evolution of the vorticity is relatively slower.

Snapshots from the animation are shown in figure 6.14.

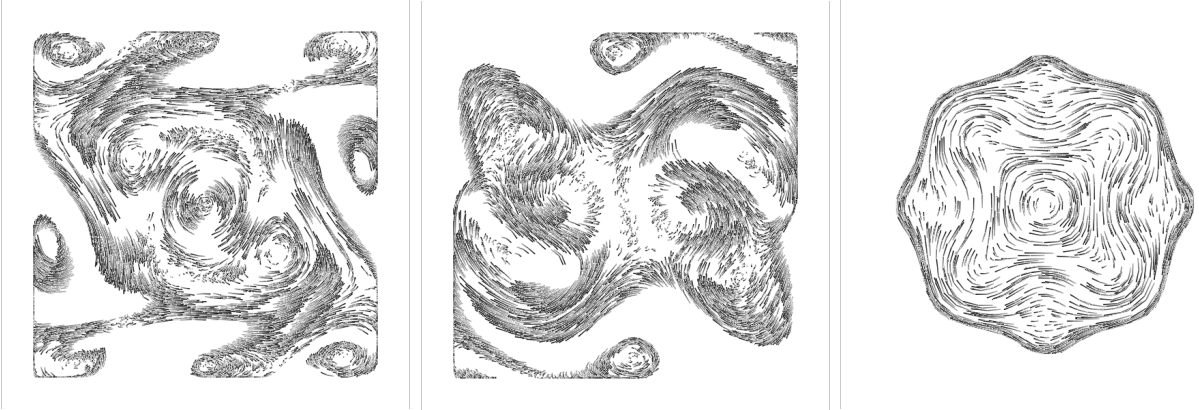


Figure 6.14: Snapshots from a artistic fluid simulation that makes use of symmetric basis functions to produce a perpetual, symmetric flow.

6.3 Performance

Domain	Precomputation	$\mathbf{C}_k$ Storage	Simulation Step	Advect Particles
Analytic	$O(n^2)$	$O(n^2)$	$O(z) + O(n^2) \approx O(n^2)$	$O(np)$

Figure 6.15: Performance and storage complexity for n basis functions, p particles, and z non-zero structure coefficients in a completely analytic simulation.

For a mesh-free analytic simulation, the table in Figure 6.15 shows theoretical performance and storage complexity for n basis functions, p particles and a total of z non-zero entries in the set of sparse $\mathbf{C_k}$ matrices. Simulation time is dominated by the n vector-matrix-vector products required to compute each component of the tangent vector

$$\frac{\partial \mathbf{w}_k}{\partial t} = \mathbf{w}^T \mathbf{C}_k \mathbf{w}.$$

The simulation time is $O(z) + O(n^2)$. The first term $O(z)$ is due to sparse matrix vector products $\mathbf{C}_k\mathbf{w}$, the total of which are proportional to the number of non-zero entries. The $O(n^2)$ term is due to n evaluations of the remaining vector dot products of length n. Note that the number of non-zero entries z is a function of n. For the analytic domains considered, the precomputed advection of each pair contributes to a constant number of non-zero coefficients, making $z(n) = O(n^2)$ and the entire simulation time $O(n^2)$.

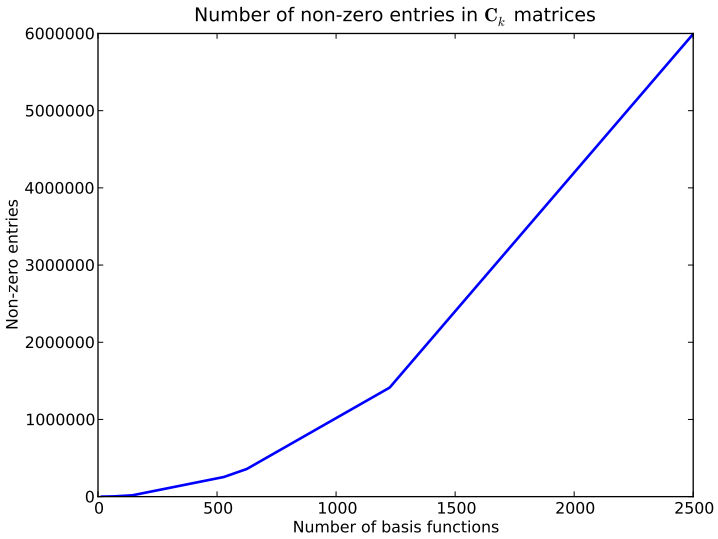


Figure 6.16: The number of non-zero structure coefficients in the $\mathbf{C}_k$ matrices determines the simulations performance. The number of non-zero entries as a function of the number of basis functions n is $O(n^2)$.

Despite this worst-case polynomial time complexity, practical simulation times are measured in milliseconds and grow nearly linearly when employing less than 1000 basis functions. Furthermore, the simulation time is far outweighed by particle advection and rendering costs. Table 6.17 shows experimental results for an analytic 2-D domain, and Figure 6.18 shows a

graph of simulation time versus the number of basis functions. Note that for purely analytic basis fields, particle advection is proportional to the dimension of the basis since for each particle n basis functions must be evaluated and summed. Precomputation time for analytic domains is limited to evaluating the non-zero entries of the $\mathbf{C}_k$ matrices through symbolic formulas.

Domain	Basis Dim.	Non-zero $\mathbf{C}_k$ entries	Precomputation (s)	$\mathbf{C}_k$ Storage (kb)	Simulation Step (ms)	Advect Particles(ms)
2-D Rect	16	128	0.016	5	3.4	12.2
2-D Rect	64	3040	0.05	24	13	50
2-D Rect	144	17136	0.85	133	35	115
2-D Rect	529	254774	12.5	1990	120	400
2-D Rect	625	358368	16.5	2800	145	476
2-D Rect	1225	1412494	65	11032	311	921
2-D Rect	2500	5994050	310	46818	803	1907

Figure 6.17: Experimental performance and memory usage for 2-D analytic domain. Bottom: Graph of simulation time as a function of the basis dimensionality. Despite a theoretical worst-case time complexity of $O(n^2)$, for all practical purposes the simulation time scales nearly linearly when employing less than 1000 basis functions.

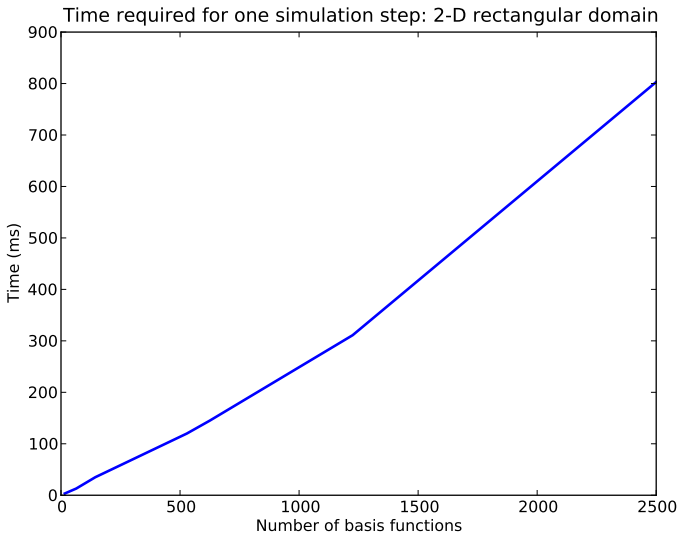


Figure 6.18: Simulation time as a function of basis dimensionality. Despite a theoretical time complexity of $O(n^2)$, for all practical purposes the simulation time scales nearly linearly when employing less than 1000 basis functions.

When both the number of particles and the basis dimension are large, the cost of analytic particle advection becomes prohibitive. Alternatively, even when using an analytic domain, the velocity basis fields may be sampled and stored. This removes some of the benefits of analyticity; the particle velocities are now interpolated on a grid, but they may be looked up in constant time, enabling particle advection in $O(p)$ time. Note that in this case, the

sampled grid is only used for looking up particle velocity. It does not change the run time performance of the simulation. Table 6.19 shows experimental results for an analytic 3-D cubic domain with velocity basis fields sampled and stored on a 32^3 grid.

Domain	Basis Dim.	Precompute (s)	$\mathbf{C}_k$ (kb)	Basis fields (Mb)	Simulation Step (ms)	Advect Particles (ms)
3-D Cube	28	0.4	2.5	32	1.7	0.72
3-D Cube	81	3.6	29	20	4.7	0.68
3-D Cube	176	15.7	160	44	10.7	0.68
3-D Cube	325	54	600	81	21.5	0.67
3-D Cube	833	355	4300	208	71.2	0.67
3-D Cube	1701	1487	19000	425	225	0.67
3-D Cube	2300	2760	35000	575	375	0.67

Figure 6.19: Performance and memory usage for 3-D analytic domain. Particle velocity is interpolated on a 32^3 grid, necessitating storage of sampled velocity fields, but making particle advection $O(p)$ time instead of $O(np)$ time.

For simulation on a discrete simplicial mesh, Table 6.20 shows theoretical performance and storage complexity for n basis functions and t mesh elements. Precomputation for meshed domains has two steps: the search for eigenvectors (eigendecomposition) and bracket evaluation. In this table we ignore the eigendecomposition cost, as it is difficult to estimate and likely less of a factor than bracket calculation. Bracket evaluation requires the computation over a mesh with t elements for every pair of n basis functions, making it $O(tn^2)$. For a number of different meshes and basis dimensions, Table 6.21, 6.23 and Figures 6.22, 6.25 and 6.24 show tables and graphs of experimental precomputation times, memory usage and run time performance.

Domain	Precomputation	Simulation Step	$\mathbf{C}_k$ Storage	Basis Field Storage
Simplicial	$O(tn^2)$	$\approx O(n^2)$	$O(n^2)$	$O(tn)$

Figure 6.20: Performance and storage complexity for n basis functions on a simplicial mesh with t elements. Precomputation time is assumed to be dominated by bracket evaluation, as opposed to eigenvalue decomposition.

Domain	Tets	Basis Dim.	Eigendecomp (s)	Bracket Eval (s)	**Total Precompute (s)**
Bust	10k	16	67.6	13.0	80.6
Bust	10k	32	113	43.5	156.5
Bust	10k	64	191	196	387
Bust	10k	128	340	1020	1360
Head	15k	16	64.3	18.8	83.1
Head	15k	32	95.0	61.8	156.8
Head	15k	64	176	260	436
Head	15k	128	319	1370	1689
Horse	20k	16	143	23.3	166.3
Horse	20k	32	252	84.5	336.5
Horse	20k	64	416	373	789
Horse	20k	128	707	1970	2677

Figure 6.21: Precomputation time for tetrahedral meshes.

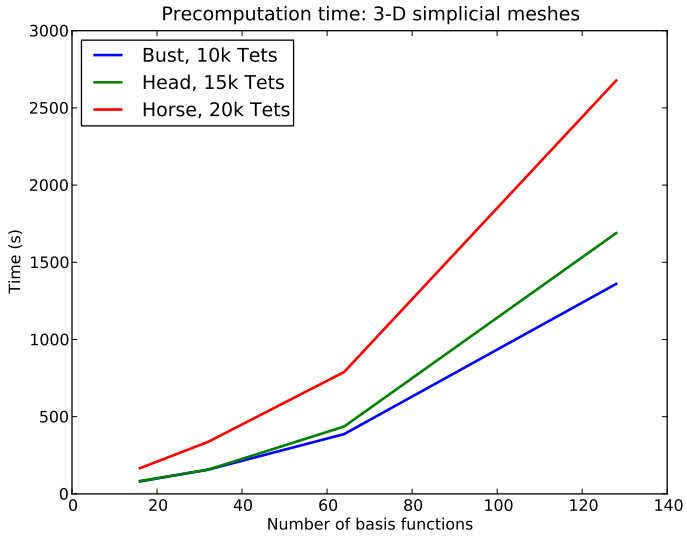


Figure 6.22: Precomputation time for tetrehedral meshes. The theoretical time complexity is $O(n^2)$.

Domain	Tets	Basis Dim.	$\mathbf{C}_k$ (kb)	Basis fields (Mb)	Simulation Step (ms)	Advect Particles (ms)
Bust	10k	16	7.5	2.9	2.1	11.5
Bust	10k	32	31	5.83	4.25	11.6
Bust	10k	64	126	11.7	8.5	11.5
Bust	10k	128	508	23.3	17	11.6
Head	15k	16	7.5	3.94	2.6	12
Head	15k	32	31	7.88	5.1	12
Head	15k	64	126	15.7	9.9	12.5
Head	15k	128	508	31.5	19.8	12.5
Horse	20k	16	7.5	5.4	3.1	13.3
Horse	20k	32	31	10.8	6.1	13.3
Horse	20k	64	126	21.6	12	13.6
Horse	20k	128	508	43.2	23	13.9

Figure 6.23: Memory usage and run time performance for tetrahedral meshes.

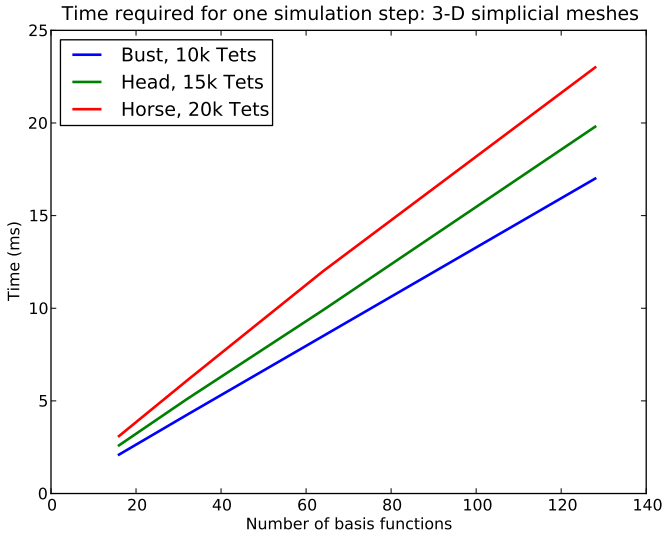


Figure 6.24: Simulation time for tetrehedral meshes. Although the theoretical performance is $O(n^2)$, it is nearly $O(n)$ for typical ranges of n.

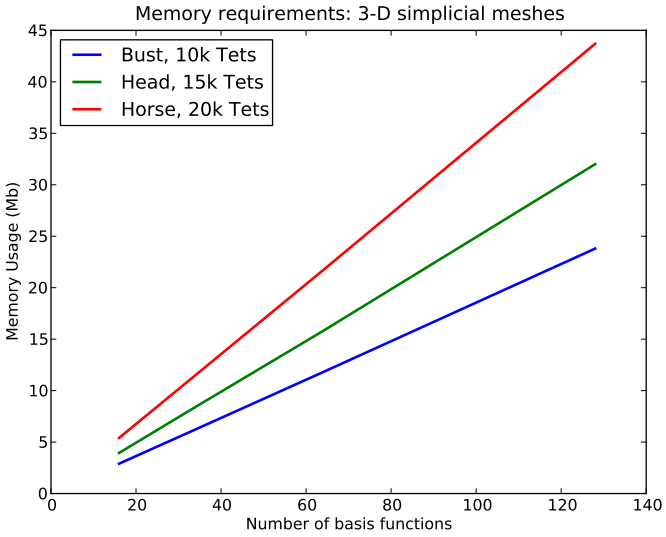


Figure 6.25: Memory usage for tetrehedral meshes for n basis functions and t tetrehedra. The theoretical storage requirements for $\mathbf{C}_k$ matrices is $O(n^2)$, but is far outweighed by the $O(tn)$ storage of the basis fields.

Chapter 7

Future Work

In this section, we propose some directions for future research related to artistic control, as well as improvements and limitations to our core method.

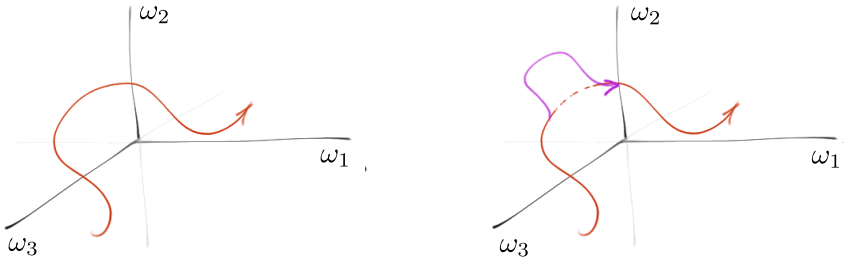
7.1 Artistic Control and Creative Use

As explained in the introduction, our motivation for this work was to design techniques amenable to user control. This thesis has focused mainly on a method of fluid simulation, but we believe many of the properties of our technique can be exploited for artistic purposes.

Perturbation of Physical Dynamics We described how geometrically, the dynamics of a fluid are a one dimensional path through the configuration space of linear superpositions of divergence free fields. In Chapter 5 we showed how to find a path corresponding to a physical flow. However, note that *any* smooth curve through this configuration space, physical or not, is in some sense 'fluid-like' since it always represents a continuously changing divergence free field that respects all boundary conditions. It would hence be interesting to explore how users could construct arbitrary continuous curves, or to perturb existing ones.

For example, starting from a physical path that has been precomputed using our method, a user could perturb or manipulate the path to achieve an artistic effect or fix some small undesired motion. Note that a fluid's dynamics depend only on the systems current state – an incompressible fluid has no memory. Hence, if a user manipulates a portion of a path but then returns to a point on the original, the subsequent dynamics would continue as previously calculated.

Spectral Energy Control We have explained that due to orthogonality of the employed basis and its correspondence with vorticity of varying scales, we have a mechanism for controlling energy independently at different scales. It is conceivable to create an interface showing a bar graph of the energy in the spectrum, much like the graphic equalizer on a stereo system. Users could draw envelopes or 'filters' describing which parts of the spectrum should be amplified, attenuated or ignored. These filters could be time varying, achieving crescendos of turbulence or gradual calming. Essentially, instead of just offering perfect energy preservation, or decay corresponding to physical viscosity, we have a means for a user to exert more expressive control or to meet specific measurements.



inin

Figure 7.1: Left: A physical path through the space of Laplacian eigenfunctions. Right: A portion of the path could be perturbed by a user for artistic effect.

Space-time Control Space-time control for fluids has been attempted previously by McNamara et al. [21]. These methods are quite slow because the optimization scales sharply with the size of the grid, making them impractical. A low dimensional basis offers a good setting to implement control policies that would be intractable in higher dimensions. Using an Eulerian method, making the grid extremely coarse to speed up optimization is not really an option – a 16 grid cell simulation will not produce convincing fluid motion. However, our method will produce natural fluid motion with only a small number basis functions. Additionally, we have analytic expressions available for time derivatives which are in many cases very useful for optimization algorithms.

We would like to attempt to use our method to make space time control of fluids practical, in order to meet user specified constraints such as as avoidance areas (so as to prevent smoke from drifting in front of a character's face), keyframes or any general space time objective.

Time Reversibility for Control Time reversible integrators have been described for both the dynamics of the velocity field and its advected quantities. Time reversibility for artistic control of rigid bodies has been explored previously by Twigg [34]. An animator may specify the *final* state of an animation, and then run a simulation in reverse. The resulting forward animation will proceed naturally to the intended final state. We would like to explore similar techniques applied to fluids. In addition to specifying a final state, time reversibility could be applied directly in a user's workflow. An animator could have the ability to scrub backwards and forwards in time while making parameter adjustments to adjust the behavior of a fluid simulation, without the computer having to store each keyframe in memory.

Texture Synthesis Although not presented in this thesis, we believe it will be straightforward to extend our method to operate on triangular meshes embedded in three dimensions. This would allow efficient calculation of fluid flow on the *surfaces* of models. By advecting colors, this could be used as a means of texture synthesis to produce patterns not easily obtained with existing surface noise methods.

Fluid Simulations Exhibiting Spatial Symmetry Some basis fields exhibit symmetry in one or more spatial directions. The dynamics amongst basis fields with the same symmetry are closed within their symmetry group. As shown in the results section, this property can be used to initiate flows with a few carefully selected basis functions so that the resulting flow changes constantly, but always retains symmetric behaviour. Perturbation in real systems

make such flows highly unlikely to occur in nature, but they are interesting artistically. They could also be useful in texture synthesis applications when a symmetric pattern is desired.

7.2 Improvements to Fluid Simulator

Moving Obstacles Our method precomputes basis fields for a static geometry. Moving obstacles necessarily require the basis functions to be recomputed at each time step which is an expensive prospect. We require fast numerical methods of precomputing bases for arbitrary geometries or for particular constrained motions of given obstacles. The work by Treuille et al. [33] may be useful in this respect.

Tiling Fine scale turbulence is not always needed everywhere in a fluid simulation domain. Our basis functions have an ordering in the scale of their vortices, but because they have global support, cannot be used to selectively increase detail in one area. We would like to explore a mechanism that employs bases at different localized 'tiles' in the domain, essentially allowing multiple independent fluid simulations at different scales to interact. This could be accomplished both analytically for a restricted set of overlaps, or by numerical projection for irregular overlapping domains.

Performance Engineering Interactivity is a very useful property, but requires lean performance. There exist many opportunities to optimize the algorithms and implementation of our technique, including parallel GPU implementations of dynamics advection and further consideration of fast eigendecomposition methods.

Additional Analytic Bases We have derived analytic dynamics for rectangular 2-D and 3-D domains. Bases and bracket evaluation for additional topologies remain open problems. Interesting analytic geometries include a 2-D disk (harmonic bases of Bessel functions), a spherical surface or a spherical cavity. Also, different boundary conditions (for example, a wrap around boundary condition) remain to be considered.

7.3 Other Applications

Divergence free fields have many potential uses besides simulating water or smoke. For example, Treuille applied continuous fluid equations to the movement of people in crowds[33]. Divergence free or *volume preserving* transformations find uses in image analysis and shape deformation. It would be interesting to consider how properties of our method could be exploited in these fields.

Biomedical Imaging One particular application worth discussing is that of template matching in the fields of computer vision and of biomedical imaging. Fluid motion describes the optimal transport in an incompressible medium, and can be used to quantify volume preserving deformations. This has been applied to develop useful metrics and procedures for comparison of biomedical images. Existing methods are computationally expensive. Our model-reduced multiscale approach could be exploited for new heuristic methods to computing optimal deformations. Our method uses analytic formulations for rectangular domains making it very well suited for images.

Chapter 8

Conclusion

We have presented a method of fluid simulation using Laplacian eigenfunctions. Compared to existing techniques, this method is not merely an incremental improvement but represents a completely novel approach. We have described many of its unique properties and their 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 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 great computational efficiency, allowing interactive performance. The existence of closed form solutions for simple domains allows precise evaluation of velocities thereby improving accuracy and visual quality.

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 believe there is great potential for our method to be exploited in other diverse research areas such as optimal transport and biomedical imaging. In addition to these, we sincerely hope that others will seek to extend and apply it in both creative and practical contexts.

Appendix A

Bracket Evaluation for Analytic Domains

In this appendix we outline the evaluation of the Lie bracket for additional analytic domains.

A.1 3-D Bounded Rectangular Cavity

In 3-D, vorticity can no longer be described as a scalar $\mathbf{a_z}$ component. Instead, vorticity is a vector and can be viewed as an oriented axis around which the fluid velocity tends to rotate at a given point. Hence, the resulting basis functions are more complex. In addition to being parameterized by a vector wave number $k \in \mathbb{Z}^3$, three distinct forms must be considered. These forms are similar, but are rotated to align along the three axes. An additional index is appended to k to identify them uniquely, taking values between 1 and 3. Our motivation to choose this basis stems from electromagnetics, where these fields describe the electric field of the transverse electric (TE) modes of a rectangular cavity in each of three reference directions [7]. The vorticity basis functions $\{\phi_{k,i}\}$ are

$$\begin{aligned}
\phi_{k,1} &= 0\mathbf{a_x} + k_3 \sin(k_1 x)\cos(k_2 y)\sin(k_3 z)\mathbf{a_y} - k_2 \sin(k_1 x)\sin(k_2 y)\cos(k_3 z)\mathbf{a_z} \\
\phi_{k,2} &= k_3 \cos(k_1 x)\sin(k_2 y)\sin(k_3 z)\mathbf{a_x} + 0\mathbf{a_y} - k_1 \sin(k_1 x)\sin(k_2 y)\cos(k_3 z)\mathbf{a_z} \\
\phi_{k,3} &= k_2 \cos(k_1 x)\sin(k_2 y)\sin(k_3 z)\mathbf{a_x} - k_1 \sin(k_1 x)\cos(k_2 y)\sin(k_3 z)\mathbf{a_y} + 0\mathbf{a_z}.
\end{aligned}$$

The associated velocity basis fields $\{\Phi_k\}$ are

$$
\begin{aligned}
\Phi_{k,1} = \frac{1}{\lambda_k} \big(& (k_2^2 + k_3^2)\cos(k_1 x)\sin(k_2 y)\sin(k_3 z)\mathbf{a_x} \\
& - k_1 k_2 \sin(k_1 x)\cos(k_2 y)\sin(k_3 z)\mathbf{a_y} \\
& - k_1 k_3 \sin(k_1 x)\sin(k_2 y)\cos(k_3 z)\mathbf{a_z}) \\
\Phi_{k,2} = \frac{1}{\lambda_k} \big(& (k_2 k_1 \cos(k_1 x)\sin(k_2 y)\sin(k_3 z)\mathbf{a_x} \\
& - (k_1^2 + k_3^2)\sin(k_1 x)\cos(k_2 y)\sin(k_3 z)\mathbf{a_y} \\
& + k_2 k_3 \sin(k_1 x)\sin(k_2 y)\cos(k_3 z)\mathbf{a_z}) \\
\Phi_{k,2} = \frac{1}{\lambda_k} \big(& (-k_3 k_1 \cos(k_1 x)\sin(k_2 y)\sin(k_3 z)\mathbf{a_x} \\
& - k_3 k_2 \sin(k_1 x)\cos(k_2 y)\sin(k_3 z)\mathbf{a_y} \\
& + (k_1^2 + k_2^2)\sin(k_1 x)\sin(k_2 y)\cos(k_3 z)\mathbf{a_z}\big)
\end{aligned}
$$

where $\lambda_k = k_1^2 + k_2^2 + k_3^2$. Examples of the velocity basis fields are shown in Fig A.1

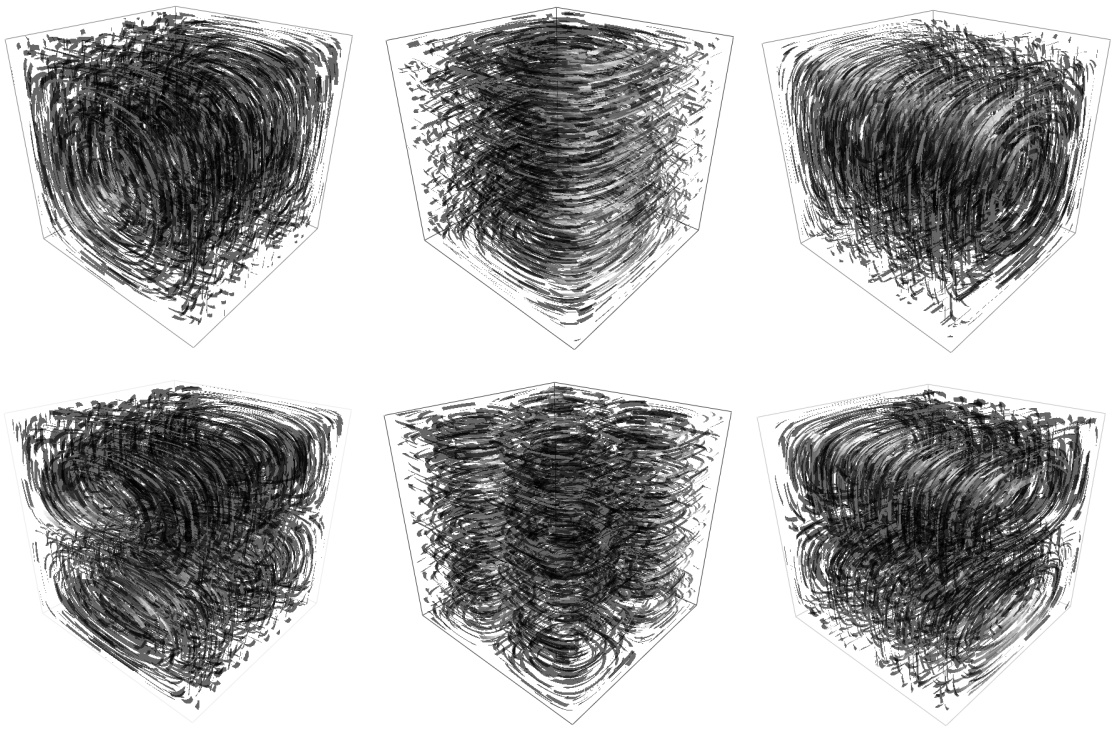


Figure A.1: Examples of $\Phi_{k,i}$ for analytic 3D rectangular domain.

Because these fields contain non-zero components in each component, the resulting expansion of $[\Phi_i, \phi_j]$ results in an explosion of trigonometric terms and is very tedious to evaluate. A symbolic math program was used to evaluate this expansion. Invariably, the result simplifies to a form

$$\begin{aligned}[\Phi_{a,i}, \phi_{b,i}] = D\cos((a_1 \pm b_1)x)\sin((a_2 \pm b_2)y)\sin((a_3 \pm b_3)z)\mathbf{a_x} \\ +E\sin((a_1 \pm b_1)x)\cos((a_2 \pm b_2)y)\sin((a_3 \pm b_3)z)\mathbf{a_y} \\ +F\sin((a_1 \pm b_1)x)\sin((a_2 \pm b_2)y)\cos((a_3 \pm b_3)z)\mathbf{a_z}. \end{aligned} \tag{A.1}$$

where D, E, F are polynomials of $a_1, a_2, a_3, b_1, b_2, b_3$. This form resembles the expressions for $\phi_{k,i}$, which is desirable since we expect the result to factor into a linear combination of these vorticity basis functions.

Factoring these expressions is difficult to do symbolically. Instead, it is possible to evaluate their projection onto $\phi_{k,i}$ by transforming to a linear space and performing the projection numerically through least squares. The nonlinear trigonometric terms in the factored result and $\phi_{k,i}$ resemble each other, so they can be considered as a basis for a linear space.

Additionally, we need only consider a finite range of k_1, k_2, k_3, since from the expressions in Eq. A.1 we see for example that $-(|a_1| + |b_1|) \leq k_1 \leq |a_1| + |b_1|$. This can also be understood from the fact that multiplication of two functions with bandlimits BL_1, BL_2 will result in a function that is bandlimited by $BL_1 + BL_2$.

The result is that the system is finite dimensional, permitting a solution method through least squares. The expression does in fact project perfectly with zero remainder on to $\phi_{k,i}$, producing sparse $\mathbf{C}_k$ matrices.

In summary, for a given pair $(a, b) = ((a_1, a_2, a_3), (b_1, b_2, b_3))$ we can evaluate D, E, F analytically, and then project the result onto the linear space spanned by the trigonometric terms of Equation A.1 in a finite range of k_1, k_2, k_3. Although we do not obtain a closed form expression for the projection coefficients in terms of $a_1, a_2, a_3, b_1, b_2, b_3$, we are still able to evaluate them *exactly* through this procedure.

Bibliography

[1] Andrey A. Agrachev and Andrey V. Sarychev. Navier-Stokes Equations: Controllability by Means of Low Modes Forcing. *Journal of Mathematical Fluid Mechanics*, 7(1):108–152, March 2005.

[2] Alexis Angelidis, Fabrice Neyret, Karan Singh, and Derek Nowrouzezahrai. 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, pages 25–32, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.

[3] Vladimir I. Arnold. On an apriori estimate in the theory of hydrodynamical stability. *Amer. Math. Soc. Transl.*, 19, 1969.

[4] Vladimir I. Arnold and Boris A. Khesin. *Topological Methods in Hydrodynamics*. Springer-Verlag New York, Inc. New York, NY, USA, 1998.

[5] Jernej Barbič and Doug L. James. Real-time subspace integration for st. venant-kirchhoff deformable models. In *ACM SIGGRAPH 2005 Papers*, SIGGRAPH '05, pages 982–990, New York, NY, USA, 2005. ACM.

[6] Nicolas Bonneel, George Drettakis, Nicolas Tsingos, Isabelle Viaud-Delmon, and Doug James. Fast modal sounds with scalable frequency-domain synthesis. In *ACM SIGGRAPH 2008 papers*, SIGGRAPH '08, pages 24:1–24:9, New York, NY, USA, 2008. ACM.

[7] David K. Cheng. Field and Wave Electromagnetics. *Addison-Wesley*, 1999.

[8] O. Christensen. *An Introduction to Frames and Riesz Bases*. Birkauser, 2003.

[9] Mathieu Desbrun, Eva Kanso, and Yiying Tong. Discrete differential forms for computational modeling. In *ACM SIGGRAPH 2005 Courses*, SIGGRAPH '05, New York, NY, USA, 2005. ACM.

[10] Mathieu Desbrun and Marie paule Gascuel. 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*, pages 61–76. Springer-Verlag, 1996.

[11] Ronald Fedkiw, Jos Stam, and Henrik Wann Jensen. Visual simulation of smoke. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '01, pages 15–22, New York, NY, USA, 2001. ACM.

[12] Nick Foster and Dimitri Metaxas. Realistic animation of liquids. *Graph. Models Image Process.*, 58:471–483, September 1996.

[13] Alain Fournier and William T. Reeves. A simple model of ocean waves. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '86, pages 75–84, New York, NY, USA, 1986. ACM.

[14] Joselph Gilland. *Elemental Magic: The Classical Art of Special Effects Animation*. Elsevier, Inc., 2006.

[15] Ernst Hairer, Christian Lubich, and Gerhard Wanner. *Geometric Numerical Integration*. Springer-Verlag New York, Inc. New York, NY, USA, 2006.

[16] Anil Nirmal Hirani. *Discrete exterior calculus*. PhD thesis, California Institute of Technology, Pasadena, CA, USA, 2003. AAI3086864.

[17] Michael Kass and Gavin Miller. Rapid, stable fluid dynamics for computer graphics. In *Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '90, pages 49–57, New York, NY, USA, 1990. ACM.

[18] J. E. Marsden and M. West. Discrete mechanics and variational integrators. *Acta Numer.*, 10:357–514, 2001.

[19] Jerrold E. Marsden and Tudor S. Ratiu. *Introduction to Mechanics and Symmetry*. Springer-Verlag New York, Berlin, Heidelberg, 1999.

[20] Nelson L. Max. Vectorized procedural models for natural terrain: Waves and islands in the sunset. In *Proceedings of the 8th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '81, pages 317–324, New York, NY, USA, 1981. ACM.

[21] Antoine McNamara, Adrien Treuille, Zoran Popović, and Jos Stam. Fluid control using the adjoint method. In *ACM SIGGRAPH 2004 Papers*, SIGGRAPH '04, pages 449–456, New York, NY, USA, 2004. ACM.

[22] Patrick Mullen, Keenan Crane, Dmitry Pavlov, Yiying Tong, and Mathieu Desbrun. Energy-preserving integrators for fluid animation. In *ACM SIGGRAPH 2009 papers*, SIGGRAPH '09, pages 38:1–38:8, New York, NY, USA, 2009. ACM.

[23] Matthias Müller, David Charypar, and Markus Gross. Particle-based fluid simulation for interactive applications. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '03, pages 154–159, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[24] Sang Il Park and Myoung Jun Kim. Vortex fluid for gaseous phenomena. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '05, pages 261–270, New York, NY, USA, 2005. ACM.

[25] H. Poincaré. Sur une forme nouvelle des équations de la méchanique. *C.R. Acad. Sci.*, 132:369–371, 1901.

[26] William T. Reeves. Particle systems a technique for modeling a class of fuzzy objects. In *Proceedings of the 10th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '83, pages 359–375, New York, NY, USA, 1983. ACM.

[27] Mikio Shinya and Alain Fournier. Stochastic motion - Motion under the influence of wind. *Computer Graphics Forum*, 11, 1992.

[28] Peter-Pike Sloan, Ben Luna, and John Snyder. Local, deformable precomputed radiance transfer. In *ACM SIGGRAPH 2005 Papers*, SIGGRAPH '05, pages 1216–1224, New York, NY, USA, 2005. ACM.

[29] B. Solenthaler and R. Pajarola. Predictive-corrective incompressible SPH. In *ACM SIGGRAPH 2009 papers*, SIGGRAPH '09, pages 40:1–40:6, New York, NY, USA, 2009. ACM.

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

[31] Jos Stam. A simple fluid solver based on the FFT. *J. Graph. Tools*, 6:43–52, September 2002.

[32] Jos Stam and Eugene Fiume. Turbulent wind fields for gaseous phenomena. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '93, pages 369–376, New York, NY, USA, 1993. ACM.

[33] Adrien Treuille, Andrew Lewis, and Zoran Popović. Model reduction for real-time fluids. In *ACM SIGGRAPH 2006 Papers*, SIGGRAPH '06, pages 826–834, New York, NY, USA, 2006. ACM.

[34] Christopher D. Twigg and Doug L. James. Backward steps in rigid body simulation. In *ACM SIGGRAPH 2008 papers*, SIGGRAPH '08, pages 25:1–25:10, New York, NY, USA, 2008. ACM.

[35] Steffen Weißmann and Ulrich Pinkall. Filament-based smoke with vortex shedding and variational reconnection. In *ACM SIGGRAPH 2010 papers*, SIGGRAPH '10, pages 115:1–115:12, New York, NY, USA, 2010. ACM.

[36] V. I. Yudovich. Non-stationary flow of an ideal incompressible liquid. *USSR Computational Mathematics and Mathematical Physics*, 3(6):1407 – 1456, 1963.