Spiral-Spectral Fluid Simulation

QIAODONG CUI, Yale University, U.S.A
TIMOTHY LANGLOIS, Adobe Research, U.S.A
PRADEEP SEN, University of California, Santa Barbara, U.S.A
THEODORE KIM, Yale University, U.S.A

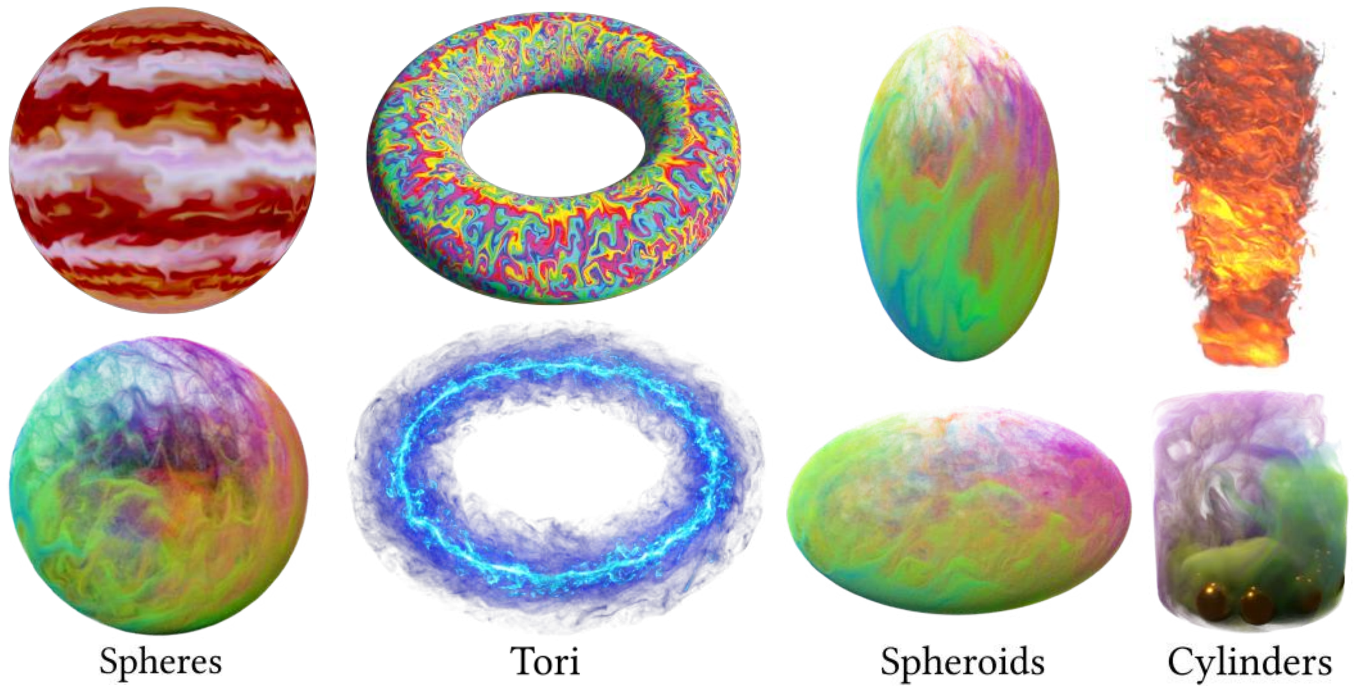


Fig. 1. Our *spiral-spectral* approach enables fluid simulations over variety of radial domains, and including both surfaces and volumes.

We introduce a fast, expressive method for simulating fluids over radial domains, including discs, spheres, cylinders, ellipses, spheroids, and tori. We do this by generalizing the spectral approach of Laplacian Eigenfunctions, resulting in what we call *spiral-spectral* fluid simulations. Starting with a set of divergence-free analytical bases for polar and spherical coordinates, we show that their singularities can be removed by introducing a set of carefully selected enrichment functions. Orthogonality is established at minimal cost, viscosity is supported analytically, and we specifically design basis functions that support scalable FFT-based reconstructions. Additionally, we present an efficient way of computing all the necessary advection tensors. Our approach applies to both three-dimensional flows as well as their surface-based, codimensional variants. We establish the completeness of our basis representation, and compare against a variety of existing solvers.

CCS Concepts: • **Computing methodologies** → **Physical simulation**.

Additional Key Words and Phrases: fluid simulation, physically based animation.

ACM Reference Format:
Qiaodong Cui, Timothy Langlois, Pradeep Sen, and Theodore Kim. 2018. Spiral-Spectral Fluid Simulation. *ACM Trans. Graph.* 37, 4, Article 202 (August 2018), 17 pages. https://doi.org/10.1145/1122445.1122456

Authors' addresses: Qiaodong Cui, Yale University, , New Haven, U.S.A, qiaodong.cui@yale.edu; Timothy Langlois, Adobe Research, , Seattle, U.S.A, tlangloi@adobe.com; Pradeep Sen, University of California, Santa Barbara, , Santa Barbara, U.S.A, psen@ece.ucsb.edu; Theodore Kim, Yale University, , New Haven, U.S.A, theodore.kim@yale.edu.

1 INTRODUCTION

Many interesting fluid phenomena exist on radial domains. Soap bubbles [Hill and Henderson 2016; Huang et al. 2020; Yang et al. 2019] and planetary flows [Stam 2003; Yaeger et al. 1986] are often approximated as two-dimensional flows on the surface of a sphere. Cylindrical twisters [Fangmeier 1996], tornadoes [Hall 2004; Yuksel et al. 2012], and mushroom clouds [Rasmussen et al. 2003] are commonplace in Hollywood disaster movies. Spherical vortex particles

[Selle et al. 2005], turbulence particles [Narain et al. 2008], and vorticles [Angelidis 2017; Park and Kim 2005] are used to both simulate new flows and enrich existing ones.

We present a fluid-simulation approach that applies to a wide variety of such domains and conforms exactly to their curved boundaries. We demonstrate its efficacy on discs, spheres, ellipses, spheroids, cylinders, and tori. Our approach is an analytic generalization of the method of Laplacian Eigenfunctions [De Witt et al. 2012] (a.k.a. the *Eigenfluids* method), which uses the eigenfunctions of the vector Laplacian as its basis. The Eigenfluids method can simulate fully inviscid flows because it formulates advection entirely within the spectral domain. We correspondingly refer to our new simulations as *spiral-spectral* fluids, due to their fully spectral and radial nature.

The original Eigenfluids approach [De Witt et al. 2012] only scaled to several hundred basis functions, resulting in relatively coarse flows. Cui et al. [2018] showed that by leveraging analytic basis functions and the Fast Fourier Transform (FFT), scalability could be asymptotically improved, which allowed tens of thousands of basis functions to be used, and yielded turbulent, highly detailed flows. Following this approach, we start from the divergence operator and design a set of vector basis functions over radial domains that support FFT-based reconstructions by design. Thus, our approach is also able to efficiently achieve similarly detailed simulations that use over ten thousand basis functions.

However, radial functions are known to contain troublesome *pole* singularities. For example, with surface-based flows on a sphere, point singularities appear at the north and south poles. In a fully volumetric sphere, these expand into a line connecting the poles. Consequently, previous work [Hill and Henderson 2016; Huang et al. 2020; Yang et al. 2019] required specialized advection schemes to address this issue.

We instead leverage the spectral nature of our approach and introduce a set of enrichment functions that are capable of handling both point and line singularities. While the spiral basis functions support fast transformations, they do not qualify as orthogonal eigenfunctions. Thus, we present a simple method for re-establishing orthogonality, and an efficient transform that allows us to still use the advection tensor of the original coordinate system. Additionally, we show that the diffusion operator, which had an analytic form in the original Eigenfluids formulation, retains this appealing form over radial domains. Our orthogonalization approach applies equally well to this operation.

Finally, we conclude by establishing the completeness of our basis, and demonstrating the efficacy of our approach across a variety of examples. In particular, we compare against modern spherical spectral methods [Burns et al. 2020; Lecoanet et al. 2019], as well as standard fluid solvers in graphics [Thuerey and Pfaff 2018]. In summary, our contributions are:

- A method for generating divergence-free *principal* basis functions over radial domains that intrinsically support FFT-based reconstructions
- *Enrichment* functions that remove the singularities that arise in the principal functions, while still supporting the FFT
- A spectral advection formulation for radial domains, as well as a fast method for applying it to orthogonalized principal and enrichment functions
- Analytic reduced-order diffusion over radial domains
- Demonstrated efficacy over a variety of geometries, including surface-based flows on spheres and tori, and volumetric flows over spheres, spheroids, tori, and cylinders

2 RELATED WORK

The most common representations for fluid simulations in computer graphics are finite differences over an Eulerian grid [Stam 1999], basis functions attached to Lagrangian particles [Müller et al. 2003], or a combination of both, such as in a Fluid Implicit Particle (FLIP) [Zhu and Bridson 2005] or Material Point Method (MPM) [Stomakhin et al. 2013] scheme. Excellent texts are available that summarize both approaches [Bridson 2015; Kim 2017].

In contrast, *radial* coordinate systems can be the most natural coordinate system for many applications, and various methods have been developed for these domains over the last few decades. Early on, Yaeger et al. [1986] used a surface-based spherical domain to simulate the surface of the planet Jupiter in the 1984 film *2010: The Year We Make Contact.* More recently, spherical domains have been used to model the features of soap bubbles [Hill and Henderson 2016; Yang et al. 2019] and subsequently improved by introducing chemomechanical factors [Huang et al. 2020]. Volumetric spheres of fluid are common in graphics, as they can be used to enrich existing simulations in the form of vortex [Pfaff et al. 2009; Selle et al. 2005] or turbulence [Narain et al. 2008; Pfaff et al. 2010] particles, or serve as the fundamental simulation primitive in the form of vorticles [Angelidis 2017; Park and Kim 2005].

Outside of graphics, spherical domains are the preferred representation for climate modeling, where early work focused on representing surface-based scalar stream functions using spherical harmonic coefficients [Silberman 1954], integrating forward in time [Baer and Platzman 1961], and different choices for basis functions [Boyd 1978; Orszag 1974]. An excellent overview of modern issues is available in Drake [2014]. Fast transforms have been developed for this case [Wedi et al. 2013], but they only apply to surface-based spherical simulations, not the general radial domains we investigate.

Hollywood disaster movies often contain phenomena with a naturally cylindrical geometry, such as tornadoes [Hall 2004; Yuksel et al. 2012] and twisters [Fangmeier 1996]. Mushroom clouds exhibit this symmetry, and so Rasmussen et al. [2003] developed a cylindrical interpolation method for efficient simulation. Outside of graphics, the study of flows through cylinders (pipe flow) goes back to Reynolds [1883], and understanding its characteristics is still an active topic in both experimental [Kühnen et al. 2018] and computational [Morinishi et al. 2004] physics. The scenario remains one of the canonical introductory examples in computational fluid dynamics (CFD) texts [Griebel et al. 1998].

Toroidal geometries have appeared in graphics in the form of smoke [Angelidis and Neyret 2005; Weissmann et al. 2014] and bubble [Padilla et al. 2019] rings, and are used more broadly in plasma physics for simulating the internal dynamics of tokamaks [Tamain et al. 2016; Zhu et al. 2018].

Our work builds on the Eigenfluids method of De Witt et al. [2012], which was further stabilized using the variational methods of Liu et al. [2015], and made asymptotically more scalable on rectangular domains by Cui et al. [2018]. More localized, wavelet-based methods have been developed for Cartesian [Mercier and Nowrouzezahrai 2020] and polar [Lessig 2019] domains, but we build on the global approach of the original method. The existence of suitable radial functions is not always clear, as scalar Laplacian eigenfunctions in polar coordinates [Grebenkov and Nguyen 2013] involve Bessel functions, which complicates the use of the FFT. Thus, we will describe several principles for selecting suitable functions in §4.

Our approach is a spectral method [Boyd 2001; Burns et al. 2020] which are known to be well-suited to cylindrical [Karpfinger et al. 2008] and spherical [Slevinsky et al. 2018] domains. Many basis functions have been proposed for curved domains [Boyd and Yu 2011; Vasil et al. 2019], but scalar basis functions (e.g., Zernike polynomials, Logan-Shepp polynomials, or spherical harmonics) must again be chosen to smooth over coordinate singularities. Novel fast-transform methods must be developed for each of these functions, and high-rank simulations are otherwise impractical. We instead focus on trigonometric functions that already support such transforms, which maintains the scalability of previous Eigenfluids approaches, while also generalizing to curved domains. Other graphics works have also leveraged the fact that projection onto a divergence-free spectral basis can be faster than a Chorin projection [Henderson 2012; Long and Reinhard 2009], and that diffusion becomes a point-wise exponential in the spectral domain [Stam 2002].

Finally, many subspace methods can be viewed as data-driven spectral methods [Kim and Delaney 2013; Stanton et al. 2013; Treuille et al. 2006]. These methods encounter the same memory storage problems as De Witt et al. [2012], which limits their basis rank. Compression [Jones et al. 2016] can ameliorate this somewhat, but the problem remains significant. Other analytic bases, such as Legendre polynomials [Gupta and Narasimhan 2007] are also possible, but the absence of a fast transform again limits the basis rank. Similar to previous methods [Barbič and James 2005; Gerszewski et al. 2015] we find it useful to enrich the basis with rigid-body-like modes. Unlike these works, our basis functions are analytic, which allows us to avoid a numerical SVD, and enables a variety of other efficiencies.

3 LAPLACIAN EIGENFLUIDS IN CARTESIAN COORDINATES

Notation: We will use unbolded lower case to denote scalars, k, unbolded upper case to denote scalar functions, $R(r)$, bold lower case to denote vectors, $\mathbf{u}$, and bold upper case to denote matrices, $\mathbf{C}$. An over-dot denotes the time derivative, i.e., $\dot{\mathbf{w}} = \frac{d\mathbf{w}}{dt}$. We use the Einstein summation convention, where a repeated index denotes summation over that index. The scalar s is used to denote the basis rank, and angled brackets are used to denote dot products, i.e., $\langle \mathbf{u}_i, \mathbf{u}_j \rangle = \mathbf{u}_i \cdot \mathbf{u}_j$. The 3rd order tensor $\mathfrak{C} \in \mathbb{R}^{s \times s \times s}$ (referred to as the advection tensor) is used to encode advection.

In order to position our algorithm, we first summarize the scalable Eigenfluids algorithm of Cui et al. [2018]. Constant density fluid flow is governed by the incompressible Navier–Stokes equations:

$$\dot{\mathbf{u}} = -\mathbf{u} \cdot \nabla \mathbf{u} + \nu \nabla^2 \mathbf{u} - \nabla p + \mathbf{f} \tag{1}$$

$$\nabla \cdot \mathbf{u} = 0. \tag{2}$$

The vector $\mathbf{f}$ denotes external forces, and ν denotes viscosity. In Laplacian Eigenfluids, the fluid velocity is represented using the following linear combination:

$$\mathbf{u} = \sum_{i=1}^{s} \mathbf{\Phi}_i \mathbf{w}_i. \tag{3}$$

The $\mathbf{\Phi}_i$ denote divergence-free eigenfunctions ($\nabla \cdot \mathbf{\Phi}_i = 0$) of the vector Laplacian operator ($\nabla^2 \mathbf{\Phi}_i = \lambda_i \mathbf{\Phi}_i$). Previous works [De Witt et al. 2012; Liu et al. 2015] sample each $\mathbf{\Phi}_i$ over a uniform grid or tetrahedral mesh and store them in memory as a basis matrix $\mathbf{U} = \begin{bmatrix} \mathbf{\Phi}_1 & \mathbf{\Phi}_2 & \dots \end{bmatrix}$. When s basis functions are sampled on a $n \times n \times n$ grid, the memory complexity is $O(sn^3)$, which can quickly exceed available memory as s grows. Reconstructing the velocity field in Eqn. 3 also takes an unwieldy $O(sn^3)$ time.

Cui et al. [2018] observed that in a rectangular Cartesian domain, the basis functions admit closed-form expressions that support fast transformations. For example, in a box domain $[0, \pi]^2$, one possible set of basis functions is

$$\mathbf{\Phi}_x = -\frac{1}{\eta_i} i_2 \sin(i_1 x) \cos(i_2 y) \tag{4}$$

$$\mathbf{\Phi}_y = \frac{1}{\eta_i} i_1 \cos(i_1 x) \sin(i_2 y), \tag{5}$$

where $i_1, i_2 \in \mathbb{Z}^+$, and $\eta_i = \sqrt{i_1^2 + i_2^2}$. Then, the discrete cosine (DCT) and sine (DST) transforms are used to represent the basis functions. For example, $\mathbf{\Phi}_x$ can be represented by DST coefficients in the x direction and DCT coefficients along y. This way, no $n \times n \times n$ grid is needed to store $\mathbf{\Phi}_x$, and memory complexity asymptotically improves from $O(sn^3)$ to $O(s)$. Velocity reconstruction (Eqn. 3) is also accelerated by the DCT and DST, improving time complexity from $O(sn^3)$ to $O(n^3 \log(n))$. However, these improvements only apply when the fluid is constrained to a closed Cartesian box.

Advection is computed using a 3rd order tensor:

$$\dot{\mathbf{w}}_g = \mathfrak{C}_{ghi} \mathbf{w}_h \mathbf{w}_i \tag{6}$$

$$\mathfrak{C}_{ghi} = \int_{\Omega} \nabla \times \mathbf{\Phi}_g \cdot (\mathbf{\Phi}_h \times \mathbf{\Phi}_i) d\Omega. \tag{7}$$

The tensor entries are computed using the variational rule of Liu et al. [2015], and solved with an implicit trapezoidal rule that guarantees conservation of energy and enables inviscid simulations. Because the basis functions are eigenfunctions of the Laplacian operator, diffusion is trivially solved with exponential decay: $\mathbf{w}_k^{t+1} = \mathbf{w}_k^t e^{\nu \Delta t \lambda_k}$. External forces are projected onto the basis, $\hat{\mathbf{f}} = \mathbf{U}^T \mathbf{f}$, and then added to coefficients: $\mathbf{w}^{t+1} = \mathbf{w}^t + \Delta t \hat{\mathbf{f}}$.

Discussion: The scalability of this approach stems from the use of trigonometric eigenfunctions that are amenable to the FFT, as well as analytic differentiation and integration. While Laplacian eigenfunctions in non-Cartesian coordinates may admit fast transformations (e.g., spherical harmonics [Mohlenkamp 1999]), they require the construction of new codes. In this work, we use basis functions that are already supported by highly-optimized and well-established FFT libraries, such as FFTW [Frigo and Johnson 2005]. The differentiation and integration over other special (e.g., Bessel) functions are also more involved, which complicates basis function design and advection tensor computation. For these multiple reasons, we specifically target trigonometric functions.

4 GENERATING BASIS FUNCTIONS

In this section, we present a method for generating basis functions for radial domains. We start with the divergence-free constraint, which results in an under-constrained system. Next, we describe a set of principles for selecting additional constraints, resulting in a system that generates functions with a range of desirable properties. In subsequent sections, we use these principles to generate basis functions for a variety of radial domains.

The radially symmetric coordinate systems we use in this paper are instances of orthogonal coordinates, where all coordinate iso-contours meet at right angles. For a vector field $\mathbf{s}$ parameterized by orthogonal coordinate variables q_i, where $i \in \{1, 2, 3\}$, the divergence operator and divergence-free condition can be expressed as [Korn and Korn 2000]:

$$\nabla \cdot \mathbf{s} = \frac{1}{g}\left[h_2 h_3 \frac{\partial \mathbf{s}_1}{\partial q_1} + h_1 h_3 \frac{\partial \mathbf{s}_2}{\partial q_2} + h_1 h_2 \frac{\partial \mathbf{s}_3}{\partial q_3}\right] = 0 \tag{8}$$

The h_i are scale factors along each dimension, and defined as the norms of the basis vectors, $h_i = |\mathbf{e}_i|$. The basis vectors are derivatives about the parameterized points $\mathbf{p}$ as follows: $\mathbf{e}_i = \frac{\partial \mathbf{p}}{\partial q_i}$. The Jacobian determinant of the coordinates is then $g = h_1 h_2 h_3$.

The divergence-free condition is a first-order differential equation where the components of $\mathbf{s}$ are the unknowns. In 2D, a divergence-free vector field can be obtained by specifying one component and solving for the other. However, two components must be specified in 3D, so the system is under-constrained. Therefore, we establish four principles for selecting constraints that yield basis functions with desirable properties.

Principle 1. The basis functions should support efficient FFT-based operations. The components of basis functions should be trigonometric (sin, cos). Since the solved component arises from derivatives and integrals, it also becomes constrained to be trigonometric. FFT-based support for the basis functions is then guaranteed.

Principle 2. The basis functions should be separable along each coordinate. For example, in spherical coordinates, we assume each component is the summed product of 1D scalar functions: $R(r)T(\theta)P(\phi)$. This reduces Eqn. 8 to an ordinary differential equation along each direction, and the integral of the basis functions becomes three separate 1D integrals. This will later allow the entries of the advection tensor to be computed analytically.

Principle 3. The basis functions must be smooth and continuous at radial and angular boundaries. The first two principles already allow a generic set of basis functions to be obtained, but the radial direction can contain singularities that must be smoothed over with appropriate boundary conditions [Hill and Henderson 2016]. In the angular direction, a periodic boundary condition is needed to maintain smoothness, e.g., across the jump between 0 and 2π. The appropriate boundary conditions should be derived for each individual coordinate and used to select the basis functions once a generic set is obtained.

Principle 4. The basis functions should also be able to represent all divergence-free flows in the domain. After applying principle 3, the basis functions are often incomplete because they lack translation and rotation modes at coordinate singularities. We complete the bases by introducing enrichment functions to capture these modes.

5 POLAR BASIS FUNCTIONS

5.1 Generating Polar Basis Functions

We will now apply the principles from §4 to polar coordinates and generate a core set that we refer to as the *principal* basis functions. We will later see that they are necessary, but not sufficient, for representing arbitrary flows. A point $\mathbf{p}$ is parameterized in polar coordinates as

$$\mathbf{p}_x = r\cos(\theta) \qquad \mathbf{p}_y = r\sin(\theta), \tag{9}$$

where $r \in [0, \infty)$, $\theta \in [0, 2\pi)$, and we limit our domain to a closed circle, $r \le 1$. The divergence-free condition (Eqn. 8) becomes:

$$\nabla \cdot \mathbf{s} = \frac{1}{r}\left(\mathbf{s}_r + r\frac{\partial \mathbf{s}_r}{\partial r} + \frac{\partial \mathbf{s}_\theta}{\partial \theta}\right) = 0. \tag{10}$$

Following the principle 2, we limit ourselves to separable solutions of the form $\mathbf{s}_r = R_r(i_1\pi r)T_r(i_2\theta)$ and $\mathbf{s}_\theta = R_\theta(r)T_\theta(\theta)$, where i_1 and i_2 are wavenumbers. Following principle 1, we also limit R_r and T_r to sine or cosine functions. The unknown functions are now $R_\theta(r)$ and $T_\theta(\theta)$, and Eqn. 10 simplifies to:

$$R_r(i_1\pi r)T_r(i_2\theta) + r\frac{\partial R_r(i_1\pi r)}{\partial r}T_r(i_2\theta) + R_\theta(r)\frac{\partial T_\theta(\theta)}{\partial \theta} = 0. \tag{11}$$

Principle 2 guarantees we can use separation of variables to solve this equation as:

$$R_\theta(r) = -R_r(i_1\pi r) - r\frac{\partial R_r(i_1\pi r)}{\partial r} \tag{12}$$

$$T_\theta(\theta) = \int T_r(i_2\theta)d\theta + L(r). \tag{13}$$

If we assign $L(r) = 0$, then R_θ and T_θ become fully specified by R_r and T_r, and we obtain two vector basis functions. (Non-zero $L(r)$ will be revisited in §5.3.) For example, the functions arising from $R_r = \sin(i_1\pi r)$ are:

$$\begin{cases}\Phi^0_r(r,\theta) &= \sin(i_1\pi r)\sin(i_2\theta)\\ \Phi^0_\theta(r,\theta) &= \frac{1}{i_2}\left(\sin(i_1\pi r) + i_1\pi r\cos(i_1\pi r)\right)\cos(i_2\theta)\end{cases} \tag{14}$$

$$\begin{cases}\Phi^1_r(r,\theta) &= \sin(i_1\pi r)\cos(i_2\theta)\\ \Phi^1_\theta(r,\theta) &= -\frac{1}{i_2}\left(\sin(i_1\pi r) + i_1\pi r\cos(i_1\pi r)\right)\sin(i_2\theta),\end{cases} \tag{15}$$

where the superscript i in Φ^i_* indexes the vector basis functions. Using the first two principles, we have generated two vector basis functions, Φ^0_* and Φ^1_*, that are both divergence-free and composed entirely of FFT-friendly trigonometric functions.

5.2 Setting Boundary Conditions

Next, we use principle 3 to establish appropriate boundary conditions. Smoothness demands a periodic boundary condition in the θ direction, $\mathbf{\Phi}(r,\theta) = \mathbf{\Phi}(r,\theta+2\pi)$, that guarantees a smooth transition between $\theta = 0$ and $\theta = 2\pi$. This boundary condition is satisfied by constraining i_2 in Eqns. 14 and 15 to a positive integer: $i_2 \in \mathbb{Z}^+$.

In the radial direction, a Dirichlet boundary can be established at the border, $\mathbf{u}_r(r = 1) = 0$, by also constraining i_1 in Eqns. 14 and 15 to $i_1 \in \mathbb{Z}^+$. Alternatively, a Neumann open-boundary condition of $\left.\frac{\partial \mathbf{u}_r}{\partial r}\right|_{r=1} = 0$ can instead be achieved [Cui et al. 2018] by offsetting i_1 by $1/2$ to obtain $i_1 \in \left(\mathbb{Z}^+ - 1/2\right)$.

Finally, polar coordinates are singular at $r = 0$, which maps the entire parameter range $\theta \in [0, 2\pi)$ to a single spatial point. Thus, consistency can only be maintained if the polar velocities at $r = 0$ all map to a single, unique velocity. Hill and Henderson [2016] express this boundary condition using a transformation between polar and Cartesian coordinates:

$$\begin{bmatrix}\mathbf{u}_x\\ \mathbf{u}_y\end{bmatrix} = \begin{bmatrix}\cos(\theta) & -\sin(\theta)\\ \sin(\theta) & \cos(\theta)\end{bmatrix}\begin{bmatrix}\mathbf{u}_r\\ \mathbf{u}_\theta\end{bmatrix}. \tag{16}$$

The Cartesian velocity $\mathbf{u}_x$ and $\mathbf{u}_y$ at $r = 0$ should be constant with respect to θ, which leads to the following boundary condition:

$$\left.\frac{\partial \mathbf{u}_r}{\partial \theta}\right|_{r=0} = \mathbf{u}_\theta \qquad \left.\frac{\partial \mathbf{u}_\theta}{\partial \theta}\right|_{r=0} = -\mathbf{u}_r. \tag{17}$$

The polar basis functions $\mathbf{\Phi}^0_*$, $\mathbf{\Phi}^1_*$ from Eqns. 14 and 15 are both zero at $r = 0$, which satisfy these conditions.

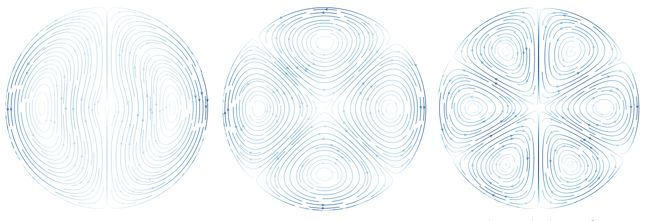
Therefore, as long as $i_1 \in \mathbb{Z}^+$ and $i_2 \in \mathbb{Z}^+$ for Dirichlet boundaries, or $i_1 \in \left(\mathbb{Z}^+ - 1/2\right)$ and $i_2 \in \mathbb{Z}^+$ for Neumann boundaries, the boundaries satisfy our smoothness conditions. These basis functions are visualized in Fig. 2 to illustrate the intuitive relationship between their wavenumbers and the resulting spatial frequencies. The two basis functions complement each other, are orthogonal ($\langle \mathbf{\Phi}^0_*, \mathbf{\Phi}^1_* \rangle = 0$) because integrating along θ always yields zero, and support fast transformations (see §9.1). This concludes our treatment of the *principal* polar basis functions. Next, we will see that these functions form a necessary, but not sufficient, set.

5.3 Enrichment Basis Functions

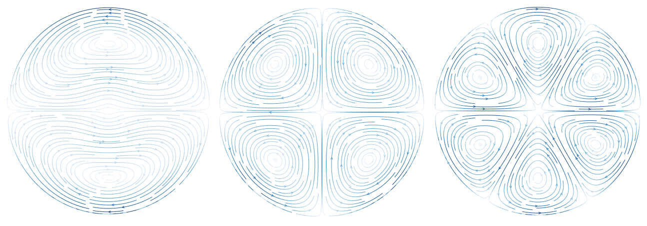
The principal basis functions are not sufficient for representing general flows inside a disc. Critically, they cannot represent non-zero velocities at $r = 0$. As shown on the left in Fig. 3, this causes all flows through the center of the disc to develop non-physical kinks.

More generally, the basis suffers from a well-known lack of rigid translation and rotation modes [Terzopoulos and Witkin 1988], as these often fall within the nullspace of the generating equations. Thus, we now revisit Eqn. 13 to derive three *enrichment* basis functions that correspond to these modes. As seen on the right of Fig. 3, their addition successfully removes the artifacts.

To derive our enrichment basis functions, we again apply principles 1 and 2 and limit ourselves to separable, trigonometric functions. To obtain a non-zero value at the center, we first select $R_r = \cos(i_1\pi r)$. The first translation mode $\mathbf{\Phi}^2_*$ can then be generated

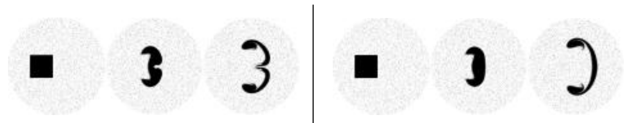


(a) $\mathbf{\Phi}^0_*$ for different wavenumbers i_1 and i_2.



(b) $\mathbf{\Phi}^1_*$ for different wavenumbers i_1 and i_2.

Fig. 2. Visualizations of polar basis functions $\mathbf{\Phi}^0_*$ and $\mathbf{\Phi}^1_*$ with Dirichlet boundaries. Wavenumber i_2 directly controls the number of periods in the θ direction.



(a) Without enrichment (b) With enrichment

Fig. 3. Left: A polar simulation with only principal functions develops a non-physical kink due to the singularity at the pole. Right: After enrichment, the same simulation flows naturally through the pole.

by inserting $T_r = \sin(i_2\theta)$ into Eqn. 13:

$$\begin{cases}\mathbf{\Phi}^2_r &= \cos(i_1\pi r)\sin(i_2\theta)\\ \mathbf{\Phi}^2_\theta &= \frac{1}{i_2}\left(\cos(i_1\pi r) - i_1\pi r\sin(i_1\pi r)\right)\cos(i_2\theta).\end{cases} \tag{18}$$

The periodic boundary condition is satisfied by constraining $i_2 \in \mathbb{Z}^+$. The Dirichlet border is satisfied by $i_1 \in \left(\mathbb{Z}^+ - 1/2\right)$, while the Neumann border is satisfied by $i_1 \in \mathbb{Z}$. As desired, the basis becomes now non-zero at $r = 0$:

$$\mathbf{\Phi}^2_r(r=0) = \sin(i_2\theta) \qquad \mathbf{\Phi}^2_\theta(r=0) = \frac{1}{i_2}\cos(i_2\theta). \tag{19}$$

The smoothness boundary condition at the pole from Eqn. 17 yields $i_2 = 1$, which further simplifies $\mathbf{\Phi}^2_*$ to:

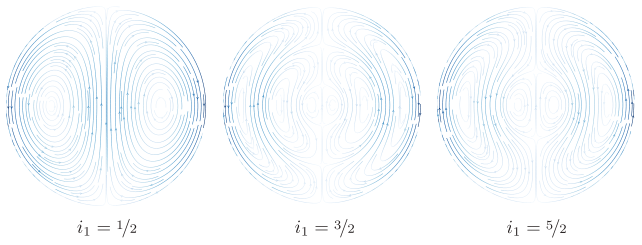
$$\begin{cases}\mathbf{\Phi}^2_r &= \cos(i_1\pi r)\sin(\theta)\\ \mathbf{\Phi}^2_\theta &= \left(\cos(i_1\pi r) - i_1\pi r\sin(i_1\pi r)\right)\cos(\theta).\end{cases} \tag{20}$$

The center now evaluates to $\mathbf{\Phi}^2_r = \sin(\theta)$ and $\mathbf{\Phi}^2_\theta = \cos(\theta)$, and corresponds to a translation in the y direction: $\mathbf{u}_x = 0, \mathbf{u}_y = 1$. These basis functions are visualized in Fig. 4(a).

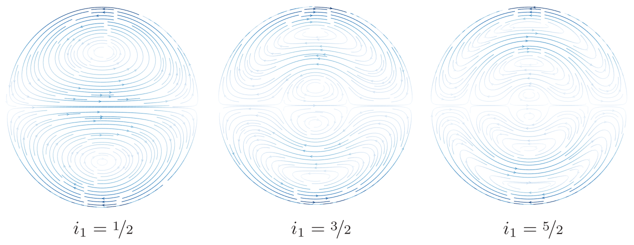
The corresponding x-translation is obtained by inserting $R_r = \cos(i_1\pi r)$ and $T_r = \cos(\theta)$ into Eqn. 13:

$$\begin{cases} \mathbf{\Phi}^3_r &= \cos(i_1\pi r)\cos(\theta) \\ \mathbf{\Phi}^3_\theta &= (-\cos(i_1\pi r) + i_1\pi r\sin(i_1\pi r))\sin(\theta). \end{cases} \tag{21}$$

This evaluates to $\mathbf{\Phi}^3_r = \cos(\theta)$ and $\mathbf{\Phi}^3_\theta = -\sin(\theta)$ at the center, yielding the desired x-translation of: $\mathbf{u}_x = 1, \mathbf{u}_y = 0$. These are further visualized in Fig. 4(b).



(a) Enrichment function $\mathbf{\Phi}^2_*$, which forms a y-translation at the center.



(b) Enrichment function $\mathbf{\Phi}^3_*$, which forms a x-translation at the center.

Fig. 4. Visualizations of enrichment basis functions with Dirichlet boundary conditions.

The last enrichment basis function captures global rotation. We revisit Eqn. 13 for the case of non-zero $L(r)$ which we now discretely sample to obtain:

$$\begin{cases} \mathbf{\Phi}^4_r &= 0, \\ \mathbf{\Phi}^4_\theta &= \sin(i_1\pi r). \end{cases} \tag{22}$$

Our basis set is now complete. The *principal* basis functions $\{\mathbf{\Phi}^0_*, \mathbf{\Phi}^1_*\}$ capture the fine details, while the *enrichment* functions $\{\mathbf{\Phi}^2_*, \mathbf{\Phi}^3_*, \mathbf{\Phi}^4_*\}$ capture the bulk rigid motions. Our final basis is their union, $\{\mathbf{\Phi}^0_*, \mathbf{\Phi}^1_*, \mathbf{\Phi}^2_*, \mathbf{\Phi}^3_*, \mathbf{\Phi}^4_*\}$. The same set generalizes to elliptical coordinates, which we show in Appendix B. We numerically verify the completeness of this basis in §10. Compared to Laplacian eigenfunctions from Cui et al. [2018], the basis is not orthogonal, so we use Gram-Schmidt to impose orthogonality, during which we filter basis functions that are sufficiently represented by existing components to improve numerical conditioning. Details are provided in Appendix A.

6 VOLUMETRIC SPHERICAL BASIS

The approach from §4 generalizes almost directly to codimensional (surface-based) flow on a sphere, so we defer that case to the supplement. However, a new *double singularity* appears at $r = 0$ in the volumetric case, so we cover that case here.

6.1 Spherical Divergence Operator

A Cartesian point $\mathbf{p}$ is parameterized in spherical coordinates as

$$\mathbf{p}_x = r\sin(\theta)\cos(\phi) \quad \mathbf{p}_y = r\sin(\theta)\sin(\phi) \quad \mathbf{p}_z = r\cos(\theta), \tag{23}$$

where $r \in [0, \infty)$, $\theta \in [0, \pi]$, $\phi \in [0, 2\pi)$. The divergence operator in spherical coordinates is:

$$\nabla\cdot\mathbf{s} = \frac{1}{r^2\sin(\theta)}\left[\sin(\theta)\frac{\partial}{\partial r}(r^2\mathbf{s}_r) + r\frac{\partial}{\partial\theta}(\mathbf{s}_\theta\sin(\theta)) + r\frac{\partial\mathbf{s}_\phi}{\partial\phi}\right]. \tag{24}$$

First, we solve the function along ϕ direction. Using principle 2, we assume the separate forms,

$$\mathbf{s}_r = A_r(r)A_\theta(\theta)B(\phi) \quad \mathbf{s}_\theta = C_r(r)C_\theta(\theta)D(\phi) \quad \mathbf{s}_\phi = E_r(r)E_\theta(\theta)F(\phi),$$

which in turn yields

$$\frac{1}{r\sin(\theta)}\left[\frac{\partial\sin(\theta)C_r(r)C_\theta(\theta)}{\partial\theta}D(\phi) + E_r(r)E_\theta(\theta)\frac{\partial F(\phi)}{\partial\phi}\right] + \frac{1}{r^2}\frac{\partial r^2A_r(r)A_\theta(\theta)}{\partial r}B(\phi) = 0. \tag{25}$$

Using separation of variables, trigonometric functions along ϕ can be solved by setting $B(\phi) = D(\phi) = \frac{\partial F(\phi)}{\partial\phi}$. The divergence-free condition then simplifies to:

$$\sin(\theta)\left[r\frac{\partial A_r}{\partial r}A_\theta + 2A_rA_\theta + \frac{\partial C_\theta}{\partial\theta}C_r\right] + \cos(\theta)C_rC_\theta + E_rE_\theta = 0. \tag{26}$$

This equation is under-constrained, and different term choices yield different basis functions. We will now derive two versions, where the first results in compact solutions that are unsuitable as a principal basis they do not vanish at the poles. The second will contain zeros that make it suitable as a principal basis.

6.2 Generating Spherical Basis Functions

6.2.1 Version 1. We group Eqn. 26 as follows:

$$\left[\sin(\theta)\left(r\frac{\partial A_r}{\partial r}A_\theta + 2A_rA_\theta + \frac{\partial C_\theta}{\partial\theta}C_r\right)\right] + \left[\cos(\theta)C_rC_\theta + E_rE_\theta\right] = 0. \tag{27}$$

The most compact solution can be obtained by assuming that each addend separately sums to zero,

$$\begin{aligned} r\frac{\partial A_r}{\partial r}A_\theta + 2A_rA_\theta + \frac{\partial C_\theta}{\partial\theta}C_r &= 0 \\ \cos(\theta)C_rC_\theta + E_rE_\theta &= 0, \end{aligned} \tag{28}$$

yielding the terms

$$A_\theta = \frac{\partial C_\theta}{\partial\theta} \quad C_r = -r\frac{\partial A_r}{\partial r} - 2A_r \quad E_rE_\theta = -\cos(\theta)C_rC_\theta. \tag{29}$$

The general solution follows:

$$\begin{cases} \mathbf{s}_r &= A_r\frac{\partial C_\theta}{\partial\theta}\frac{\partial F(\phi)}{\partial\phi} \\ \mathbf{s}_\theta &= -\left(r\frac{\partial A_r}{\partial r} + 2A_r\right)C_\theta\frac{\partial F(\phi)}{\partial\phi} \\ \mathbf{s}_\phi &= \left(r\frac{\partial A_r}{\partial r} + 2A_r\right)\cos(\theta)C_\theta F(\phi). \end{cases} \tag{30}$$

Once the boundary condition at the poles ($\theta = 0, \theta = \pi$) is established, it will become obvious that this version is unsuitable for principal bases, as it does not evaluate to zero at the poles. Therefore we will derive a second version to be used as principal bases.

6.2.2 Version 2. We obtain our second version by setting $A_\theta = A^*_\theta \sin(\theta)$ and $C_\theta = C^*_\theta \sin(\theta)$, where A^*_θ and C^*_θ are as-yet unspecified scalar functions. The $\sin(\theta)$ term pins velocity to zero at $\theta = 0$ and $\theta = \pi$, which we will need later as a boundary condition. Eqn. 26 then simplifies to:

$$\left[\sin^2(\theta)\left(r\frac{\partial A_r}{\partial r}A^*_\theta + 2A_r A^*_\theta + \frac{\partial C^*_\theta}{\partial \theta}C_r\right)\right] + \left[\sin(2\theta)C_r C^*_\theta + E_r E_\theta\right] = 0. \tag{31}$$

Again assuming each addend sums to zero yields

$$A^*_\theta = \frac{\partial C^*_\theta}{\partial \theta} \quad C_r = -r\frac{\partial A_r}{\partial r} - 2A_r \quad E_r E_\theta = -\sin(2\theta)C_r C^*_\theta, \tag{32}$$

which leads to the general basis functions:

$$\begin{cases} \mathbf{s}_r &= A_r \frac{\partial C_\theta}{\partial \theta} \sin(\theta) \frac{\partial F(\phi)}{\partial \phi} \\ \mathbf{s}_\theta &= -\left(r\frac{\partial A_r}{\partial r} + 2A_r\right) C_\theta \sin(\theta) \frac{\partial F(\phi)}{\partial \phi} \\ \mathbf{s}_\phi &= \left(r\frac{\partial A_r}{\partial r} + 2A_r\right) \sin(2\theta) C_\theta F(\phi). \end{cases} \tag{33}$$

With these two generating functions in hand, we can now specify boundary conditions.

6.3 Setting Boundary Conditions

Similar to the polar case, a periodic boundary condition is needed: $\mathbf{\Phi}(\theta,\phi) = \mathbf{\Phi}(\theta, \phi + 2\pi)$. Next, a consistency condition applies to the two poles, as well as the central axis connecting them:

$$\frac{\partial \mathbf{u}_r}{\partial \phi} = 0 \quad \frac{\partial \mathbf{u}_\phi}{\partial \phi} = -\mathbf{u}_\theta \quad \frac{\partial \mathbf{u}_\theta}{\partial \phi} = \mathbf{u}_\phi \quad \text{at } \theta = 0, \tag{34}$$

$$\frac{\partial \mathbf{u}_r}{\partial \phi} = 0 \quad \frac{\partial \mathbf{u}_\phi}{\partial \phi} = \mathbf{u}_\theta \quad \frac{\partial \mathbf{u}_\theta}{\partial \phi} = -\mathbf{u}_\phi \quad \text{at } \theta = \pi. \tag{35}$$

Finally, the center $r = 0$, contains a *double* singularity, as the parameter range $[\theta, \phi] \in [0, \pi] \times [0, 2\pi)$ maps to a single point, and necessitates simultaneous consistency requirements in both the θ and ϕ directions. These can be expressed by transforming the velocity between Cartesian and spherical coordinates:

$$\begin{bmatrix} \mathbf{u}_x \\ \mathbf{u}_y \\ \mathbf{u}_z \end{bmatrix} = \begin{bmatrix} \sin(\theta)\cos(\phi) & \cos(\theta)\cos(\phi) & -\sin(\phi) \\ \sin(\theta)\sin(\phi) & \cos(\theta)\sin(\phi) & \cos(\phi) \\ \cos(\theta) & -\sin(\theta) & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u}_r \\ \mathbf{u}_\theta \\ \mathbf{u}_\phi \end{bmatrix}. \tag{36}$$

The Cartesian velocity $\mathbf{u}_x$, $\mathbf{u}_y$ and $\mathbf{u}_z$ must be constant at $r = 0$ with respect to both θ and ϕ, which yields the boundary condition:

$$\frac{\partial \mathbf{u}_x}{\partial \phi} = \frac{\partial \mathbf{u}_x}{\partial \theta} = \frac{\partial \mathbf{u}_y}{\partial \phi} = \frac{\partial \mathbf{u}_y}{\partial \theta} = \frac{\partial \mathbf{u}_z}{\partial \phi} = \frac{\partial \mathbf{u}_z}{\partial \theta} = 0 \quad \text{at } r = 0. \tag{37}$$

This completes the boundary conditions, and we will now generate basis functions that satisfy them using the two functions from §6.2.

6.4 Principal and Enrichment Functions

6.4.1 Principal Basis Functions. We will use Version 2 as our generating function (Eqn. 33). By design, this function was weighted by $\sin(\theta)$ in all three components to create zeros at $\theta = 0$ and $\theta = \pi$ that satisfy the boundary conditions in Eqns. 34 and 35. The boundary condition in Eqn. 37 is satisfied by choosing $A(i_1\pi r) = \sin(\frac{\pi}{2}r)\sin(i_1\pi r)$, as we need a radial function where $A_r(r = 0) = 0$. The $\sin(\frac{\pi}{2}r)$ term attenuates velocities at the center, as we found that high frequency, anisotropic artifacts can otherwise appear as the viscosity approaches zero. The enrichment functions will later reintroduce velocities to this region.

Four principal basis functions can now be generated by following principle 1 and assigning C_θ and $F(\phi)$ to trigonometric functions. The first set is obtained with $C_\theta = \sin(i_2\theta)$ and $F(\phi) = \cos(i_3\phi)$:

$$\begin{cases} \mathbf{\Phi}^0_r &= i_2 i_3 \sin(\frac{\pi r}{2})\sin(i_1\pi r)\sin(\theta)\cos(i_2\theta)\sin(i_3\phi) \\ \mathbf{\Phi}^0_\theta &= -i_3 L(r)\sin(\theta)\sin(i_2\theta)\sin(i_3\phi) \\ \mathbf{\Phi}^0_\phi &= -L(r)\sin(2\theta)\sin(i_2\theta)\cos(i_3\phi). \end{cases} \tag{38}$$

The scalar function $L(r)$ denotes:

$$L(r) = i_1\pi r \sin(\omega)\cos(i_1\pi r) + (\omega\cos(\omega) + 2\sin(\omega))\sin(i_1\pi r)$$

where $\omega = \frac{\pi r}{2}$. The next three sets are:

$$\begin{cases} \mathbf{\Phi}^1_r &= i_2 i_3 \sin(\frac{\pi}{2}r)\sin(i_1\pi r)\sin(\theta)\cos(i_2\theta)\cos(i_3\phi) \\ \mathbf{\Phi}^1_\theta &= -i_3 L(r)\sin(\theta)\sin(i_2\theta)\cos(i_3\phi) \\ \mathbf{\Phi}^1_\phi &= L(r)\sin(2\theta)\sin(i_2\theta)\sin(i_3\phi), \end{cases} \tag{39}$$

$$\begin{cases} \mathbf{\Phi}^2_r &= i_2 i_3 \sin(\frac{\pi r}{2})\sin(i_1\pi r)\sin(\theta)\sin(i_2\theta)\sin(i_3\phi) \\ \mathbf{\Phi}^2_\theta &= i_3 L(r)\sin(\theta)\cos(i_2\theta)\sin(i_3\phi) \\ \mathbf{\Phi}^2_\phi &= L(r)\sin(2\theta)\cos(i_2\theta)\cos(i_3\phi), \end{cases} \tag{40}$$

$$\begin{cases} \mathbf{\Phi}^3_r &= i_2 i_3 \sin(\frac{\pi r}{2})\sin(i_1\pi r)\sin(\theta)\sin(i_2\theta)\cos(i_3\phi) \\ \mathbf{\Phi}^3_\theta &= i_3 L(r)\sin(\theta)\cos(i_2\theta)\cos(i_3\phi) \\ \mathbf{\Phi}^3_\phi &= -L(r)\sin(2\theta)\cos(i_2\theta)\sin(i_3\phi). \end{cases} \tag{41}$$

6.4.2 Enrichment Functions for $\theta = 0, \pi$. We first enrich the basis along the central axis of $\theta = 0, \pi$, excluding the double-singularity at the center, using Version 1 from Eqn. 30.

We choose $A(i_1\pi r) = \sin(\frac{\pi}{2}r)\sin(i_1\pi r)$ to satisfy the condition from Eqn. 37. We then choose $C_\theta = \cos(i_2\theta)$, so that $\mathbf{s}_t$ and $\mathbf{s}_\theta$ are non-zero at the poles. This results in two enrichment basis functions, the first of which is generated with $F(\phi) = \cos(\phi)$:

$$\begin{cases} \mathbf{\Phi}^4_r &= i_2 \sin(\frac{\pi r}{2})\sin(i_1\pi r)\sin(i_2\theta)\sin(\phi) \\ \mathbf{\Phi}^4_\theta &= L(r)\cos(i_2\theta)\sin(\phi) \\ \mathbf{\Phi}^4_\phi &= L(r)(\cos(\theta)\cos(i_2\theta))\cos(\phi). \end{cases} \tag{42}$$

At the north pole of $\theta = 0$, the basis function evaluates to,

$$\begin{cases} \mathbf{\Phi}^4_r(\theta = 0) &= 0, \\ \mathbf{\Phi}^4_\theta(\theta = 0) &= L(r)\sin(\phi), \\ \mathbf{\Phi}^2_\phi(\theta = 0) &= L(r)\cos(\phi), \end{cases} \tag{43}$$

which satisfies Eqn. 34. At the south pole of $\theta = \pi$, we obtain

$$\begin{cases} \mathbf{\Phi}^4_r(\theta = \pi) &= 0, \\ \mathbf{\Phi}^4_\theta(\theta = \pi) &= L(r)\cos(i_2\pi)\sin(\phi), \\ \mathbf{\Phi}^2_\phi(\theta = \pi) &= -L(r)\cos(i_2\pi)\cos(\phi), \end{cases} \tag{44}$$

which satisfies the boundary of Eqn. 35. A second, similar set is generated using $F(\phi) = \sin(\phi)$:

$$\begin{cases} \mathbf{\Phi}^5_r & = i_2 \sin(\frac{\pi r}{2}) \sin(i_1 \pi r) \sin(i_2 \theta) \cos(\phi) \\ \mathbf{\Phi}^5_\theta & = L(r) \cos(i_2 \theta) \cos(\phi) \\ \mathbf{\Phi}^5_\phi & = -L(r) \cos(\theta) \cos(i_2 \theta) \sin(\phi). \end{cases} \tag{45}$$

In all previous cases, $i_2 \in \mathbb{Z}^+$, and the periodic boundary is satisfied by setting $i_3 \in \mathbb{Z}^+$. Dirichlet boundaries can be attained by setting $i_1 \in \mathbb{Z}^+$, and Neumann can be attained with $i_1 \in (\mathbb{Z}^+ - 1/2)$.

Next, we must enrich the double-singularity at the center, $r = 0$.

6.4.3 Enrichment Functions for $r = 0$. We will start with translation modes, and observe that each row of the velocity transform in Eqn. 36 can be viewed as a constant flow in each Cartesian direction. For example, the third row $\mathbf{u}_r = \cos(\theta)$, $\mathbf{u}_\theta = -\sin(\theta)$, $\mathbf{u}_\phi = 0$ is a constant flow along the z-axis, $\mathbf{u}_x = 0, \mathbf{u}_y = 0, \mathbf{u}_z = 1$. Thus, if we can find basis functions that correspond to these rows, we can satisfy the boundary conditions in Eqn. 37.

First, we focus on the last row, $\begin{bmatrix}\cos(\theta) & -\sin(\theta) & 0\end{bmatrix}$, which implies $B(\phi) = 1, D(\phi) = 1$ and $F(\phi) = 0$ for Eqn. 25, and simplifies the divergence-free constraint to:

$$\left(r\frac{\partial A_r}{\partial r} + 2A_r\right) \sin(\theta) A_\theta + C_r \left(\sin(\theta)\frac{\partial C_\theta}{\partial \theta} + \cos(\theta) C_\theta\right) = 0 \tag{46}$$

The last row also implies $A_\theta = \cos(\theta)$, and $A_r = \cos(i_1 r)$ is chosen similar to the polar case. Solving this equation, we obtain the enrichment basis for the z-axis,

$$\begin{cases} \mathbf{\Phi}^6_r & = \cos(i_1 \pi r) \cos(\theta) \\ \mathbf{\Phi}^6_\theta & = \left(-\cos(i_1 \pi r) + \frac{1}{2} i_1 \pi r \sin(i_1 \pi r)\right) \sin(\theta) \\ \mathbf{\Phi}^6_\phi & = 0. \end{cases} \tag{47}$$

As a consistency check, the center evaluates to:

$$\mathbf{\Phi}^6_*(r = 0) = \begin{bmatrix}\cos(\theta) & -\sin(\theta) & 0\end{bmatrix}^T, \tag{48}$$

which corresponds to $\mathbf{u} = \begin{bmatrix}0 & 0 & 1\end{bmatrix}^T$, and so the boundary condition in Eqn. 37 is satisfied.

Next, we enrich along the y-axis using the second row of Eqn. 36, $\begin{bmatrix}\sin(\theta)\sin(\phi) & \cos(\theta)\sin(\phi) & \cos(\phi)\end{bmatrix}$, which implies $B(\phi) = D(\phi) = \sin(\phi)$, $F(\phi) = \cos(\phi)$, $A_\theta = \sin(\theta)$ and $C_\theta = \cos(\theta)$. Substituting into Eqn. 25, we obtain:

$$\sin^2(\theta) \left[\left(r\frac{\partial A_r}{\partial r} + A_r\right) + A_r - C_r\right] + \cos^2(\theta) C_r - E_r E_\theta = 0. \tag{49}$$

This can be solved by setting

$$C_r = r\frac{\partial A_r}{\partial r} + A_r \qquad E_r E_\theta = A_r + r\frac{\partial A_r}{\partial r}\cos^2(\theta). \tag{50}$$

By assigning $A_r = \cos(i_1 r)$, we obtain the enrichment function:

$$\begin{cases} \mathbf{\Phi}^7_r & = \cos(i_1 \pi r) \sin(\theta) \sin(\phi) \\ \mathbf{\Phi}^7_\theta & = (-i_1 \pi r \sin(i_1 \pi r) + \cos(i_1 \pi r)) \cos(\theta) \sin(\phi) \\ \mathbf{\Phi}^7_\phi & = \left(-i_1 \pi r \sin(i_1 \pi r) \cos^2(\theta) + \cos(i_1 \pi r)\right) \cos(\phi). \end{cases} \tag{51}$$

Checking the center again,

$$\mathbf{\Phi}^7_*(r = 0) = \begin{bmatrix}\sin(\theta)\sin(\phi) & \cos(\theta)\sin(\phi) & \cos(\phi)\end{bmatrix}^T, \tag{52}$$

we obtain the expected $\mathbf{u} = \begin{bmatrix}0 & 1 & 0\end{bmatrix}^T$.

Swapping the ϕ functions in Eqn. 51 yields the x-axis basis:

$$\begin{cases} \mathbf{\Phi}^8_r & = \cos(i_1 \pi r) \sin(\theta) \cos(\phi) \\ \mathbf{\Phi}^8_\theta & = (-i_1 \pi r \sin(i_1 \pi r) + \cos(i_1 \pi r)) \cos(\theta) \cos(\phi) \\ \mathbf{\Phi}^8_\phi & = -\left(-i_1 \pi r \sin(i_1 \pi r) \cos^2(\theta) + \cos(i_1 \pi r)\right) \sin(\phi). \end{cases} \tag{53}$$

The center velocity is:

$$\mathbf{\Phi}^8_*(r = 0) = \begin{bmatrix}\sin(\theta)\cos(\phi) & \cos(\theta)\cos(\phi) & -\sin(\phi)\end{bmatrix}^T, \tag{54}$$

which corresponds to the desired $\mathbf{u} = \begin{bmatrix}1 & 0 & 0\end{bmatrix}^T$. For these three functions, Neumann boundaries can be attained by setting $i_1 \in \mathbb{Z}$, and Dirichlet with $i_1 \in (\mathbb{Z}^+ - 1/2)$.

Finally, to capture circular flow around the center axis, we add the last enrichment basis:

$$\begin{cases} \mathbf{\Phi}^9_r & = \mathbf{\Phi}^9_\theta = 0 \\ \mathbf{\Phi}^9_\phi & = \sin(\frac{\pi r}{2}) \sin(i_1 \pi r) \sin(i_2 \theta). \end{cases} \tag{55}$$

Circular flows along the other axes are already captured by the principal functions, so no further enrichment is necessary. The final volumetric spherical basis is the union $\{\mathbf{\Phi}^0_*, \mathbf{\Phi}^1_*, \ldots \mathbf{\Phi}^9_*\}$.

Generalization to cylinders and tori proceeds similarly, and spheroids are direct extensions of the spherical case. Thus, all are shown in the supplementary material.

7 SPIRAL-SPECTRAL FLUID DYNAMICS

With the basis functions in place, we now address the dynamics of a spiral-spectral fluid. We first examine advection, as we would like to apply the variational approach of Liu et al. [2015], which both guarantees stability and enables inviscid simulations. However, the approach assumes the basis functions are orthogonal, which is no longer true in our case. Thus, we present an orthogonalization approach that allows variational advection while maintaining the analytic nature of our basis functions.

Next, we address the diffusion, which has a trivial exponential decay solution when the Laplacians of the basis functions project onto themselves. This no longer holds for our new spiral functions, so we present an analytic projection of the diffusion operator onto our basis. The resulting system can be solved at a negligible cost relative to the advection solver.

7.1 Advection With Orthogonalization

In this section, for the sake of brevity, we will refer to explicit instantiations of our basis functions using a single subscript, $\mathbf{\Phi}_i$. For example, $\mathbf{\Phi}_6$ could refer to $\mathbf{\Phi}^0_*$ with explicit wavenumbers $i_1 = 2$ and $i_2 = 3$.

Similar to when rigid modes or modal derivatives are added to a subspace basis [Barbič and James 2005], we must first orthogonalize our basis functions. We use a Gram-Schmidt process (see Appendix A) to obtain a set of orthogonal basis functions $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_s\}$. The Gram-Schmidt process yields a transfer matrix $\mathbf{A}$ that transforms our spiral basis functions into an orthogonal basis:

$$\mathbf{V} = \mathbf{AP}. \tag{56}$$

The matrix $\mathbf{V}$ is a concatenation of all of the orthogonal basis functions, i.e., $\mathbf{V}^T = \begin{bmatrix}\mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_s\end{bmatrix}$, and matrix $\mathbf{P}$ concatenates our spiral basis functions $\mathbf{\Phi}_i$, i.e., $\mathbf{P}^T = \begin{bmatrix}\mathbf{\Phi}_1 & \mathbf{\Phi}_2 & \cdots & \mathbf{\Phi}_s\end{bmatrix}$. We index into these basis functions with a subscript, i.e., $\mathbf{v}_i$ and $\mathbf{\Phi}_i$ where $1 \leq i \leq s$.

The basis coefficients $\mathbf{w}_i$ in Eqns. 3 and 7 now correspond to the coefficients of $\mathbf{v}_i$. With this orthogonal basis, we can compute the advection tensor of Liu et al. [2015]:

$$\mathfrak{C}_{ghi} = \int (\nabla \times \mathbf{v}_i) \cdot (\mathbf{v}_g \times \mathbf{v}_h) d\Omega. \tag{57}$$

In general, we do not have compact expressions for $\mathbf{v}_i$, so we expand $\mathfrak{C}$ in terms of our spiral basis functions $\mathbf{\Phi}_i$:

$$\begin{aligned} \mathfrak{C}_{ghi} &= \mathbf{A}_{il}\mathbf{A}_{gm}\mathbf{A}_{hn} \int (\nabla \times \mathbf{\Phi}_l) \cdot (\mathbf{\Phi}_m \times \mathbf{\Phi}_n) d\Omega \\ &= \mathbf{A}_{gm}\mathbf{A}_{hn}\mathbf{A}_{il}\mathfrak{T}_{mnl}. \end{aligned} \tag{58}$$

We use $\mathfrak{T}$ to denote the advection tensor computed using the original $\mathbf{\Phi}_i$. This tensor can be computed analytically, as $\mathbf{\Phi}_i$ are analytic.

A straightforward approach would be to compute $\mathfrak{T}$ and then orthogonalize according to Eqn. 58. However, this approach is $O(s^4)$ for s basis functions, because computing one entry of $\mathbf{A}_{il}\mathfrak{T}_{mnl}$ requires iterating over $O(s)$ entries in $\mathfrak{T}$. Precomputing the tensor is already an $O(s^3)$ operation that can take hours for large $s \approx 10000$.

We instead avoid this undesirable $O(s^4)$ bottleneck by orthogonalizing the advection tensor *on-the-fly* inside the implicit advection solver. The orthogonalization can be achieved with only a few extra matrix-vector products during each iteration of our conjugate gradients (CG) solver. The trapezoidal update rule is used to solve for the advection:

$$\left(\mathbf{I} - \frac{\Delta t}{2}\mathbf{C}\right)\mathbf{w}^{t+1} = \frac{\Delta t}{2}\mathbf{C}\mathbf{w}^t + \mathbf{w}^t + \mathbf{f} \tag{59}$$

where $\mathbf{C}_{ij} = \mathfrak{C}_{ijk}\mathbf{w}^t_k$. We use Eqn. 58 to expand the matrix $\mathbf{C}$ in terms of the original tensor $\mathfrak{T}$:

$$\begin{aligned} \mathbf{C}_{gh} &= \mathbf{A}_{gm}\mathbf{A}_{hn}\mathfrak{T}_{mnl}\mathbf{A}_{il}\mathbf{w}_i \\ &= \mathbf{A}_{gm}\mathbf{A}_{hn}\mathfrak{T}_{mnl}\mathbf{y}_l \\ &= \mathbf{A}_{gm}\mathbf{T}_{mn}\mathbf{A}_{hn} \\ &= (\mathbf{A}\mathbf{T}\mathbf{A}^T)_{gh}. \end{aligned} \tag{60}$$

The vector $\mathbf{y} = \mathbf{A}^T\mathbf{w}$ is the coefficients of the original spiral basis functions, and contracts with the tensor $\mathfrak{T}$ to yield the matrix $\mathbf{T}$. Thus, the new system matrix becomes $\mathbf{A}\mathbf{T}\mathbf{A}^T$.

Previous work [Cui et al. 2018] solved Eqn. 59 using conjugate gradients on the normal equations (CGNR) [Saad 2003]. Our new system matrix requires six matrix-vector products at each CGNR iteration: three for $\mathbf{A}\mathbf{T}\mathbf{A}^T$, and three for its transpose. A detailed pseudocode of the CGNR algorithm is shown in §8 of our supplementary material. Compared to Cui et al. [2018], four extra matrix-vector products are needed. However, $\mathbf{A}$ is very sparse, because most spiral basis functions were already orthogonal (see Appendix A). These sparse matrix-vector products with $\mathbf{A}$ are relatively negligible compared to the dense matrix-vector multiplies elsewhere in the solver, so we found the run-time performance of CGNR to be similar to previous work.

7.2 Analytically Reduced Diffusion

Next, we address diffusion in spiral-spectral fluid simulations. An implicit diffusion step [Stam 1999] can be written in Cartesian coordinates as:

$$(\mathbf{I} - \nu\Delta t\nabla^2)\mathbf{u}^{t+1} = \mathbf{u}^t \tag{61}$$

Expanding $\mathbf{u}$ in terms of our orthogonal basis functions yields:

$$(\mathbf{I} - \nu\Delta t\nabla^2)\mathbf{V}^T\mathbf{w}^{t+1} = \mathbf{V}^T\mathbf{w}^t. \tag{62}$$

Left-multiplying this equation with $\mathbf{V}$, and expanding in terms of the transfer matrix $\mathbf{V} = \mathbf{A}\mathbf{\Phi}$, obtains an implicit diffusion equation:

$$(\mathbf{I} - \nu\Delta t\mathbf{A}\mathbf{D}\mathbf{A}^T)\mathbf{w}^{t+1} = \mathbf{w}^t. \tag{63}$$

The matrix $\mathbf{D} \in \mathbb{R}^{s\times s}$ is equal to $\mathbf{P}\nabla^2\mathbf{P}^T$. The entries of $\mathbf{D}$ can then be computed as:

$$\mathbf{D}_{ij} = \int_\Omega \mathbf{\Phi}_i \cdot (\nabla^2\mathbf{\Phi}_j) d\Omega. \tag{64}$$

Compared to previous work which explicitly integrates against a subspace basis matrix [Kim and Delaney 2013; Stanton et al. 2013; Treuille et al. 2006], this integration can be computed analytically, because both the basis functions $\mathbf{\Phi}_i$ and their Laplacians $\nabla^2\mathbf{\Phi}_j$ are analytic. The complexity of precomputing $\mathbf{D}$ is thus $O(s^2)$.

While Eqn. 63 can be pre-factored, we use a CG solver because $\mathbf{D}$ is very sparse in practice, and each iteration only involves three sparse matrix-vector products. We found that Eqn. 63 converges in a similar number of iterations as advection, but requires no dense matrix-vector product. Thus, the diffusion solve is negligible relative to the advection solve.

8 COMPARISONS TO OTHER SPECTRAL METHODS

In this section, we discuss the completeness of our chosen spectral functions, and compare against other spectral methods.

8.1 Completeness of Basis Functions

One important question is whether our chosen spectral basis is complete, i.e., that it can represent any arbitrary vector field over its domain. To establish this, we compare it to an established set of spherical harmonic functions that are known to be divergence-free, vector valued, and complete [Barrera et al. 1985; Hill 1954]. In §7 of our supplementary material, we show that for any basis function in that set, there exists a non-zero projection of our own bases onto that function. Therefore, our functions do not omit any spectral modes, and are complete. Further details are in the supplemental material, where we also show that several low-order vector spherical harmonics on the sphere are in fact identical to our basis functions.

8.2 Comparison with Existing Spectral Methods

We compared our Spiral-Spectral method to the spherical-spectral method of Lecoanet et al. [2019], which has an open-source implementation in the Dedalus [Burns et al. 2020] framework. The results are shown in Fig. 5. In this scene, we applied an impulse that pushed two blocks of smoke into each other near the center of a sphere. No other external forces were applied, and we ran the experiment under different viscosities. We used a similar number of basis functions in both simulations. Advection in Dedalus is computed using an explicit scheme on a spatial quadrature grid, which

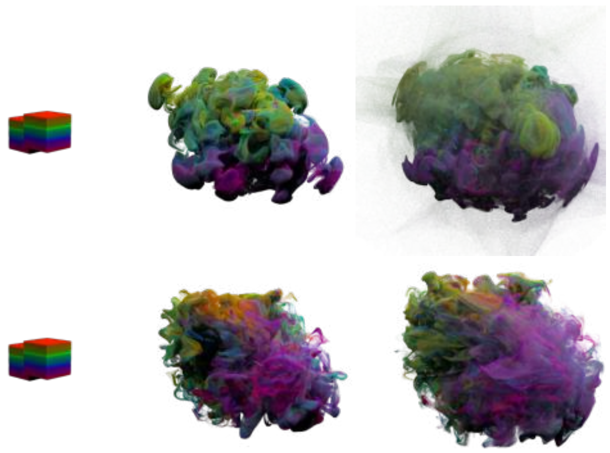


Fig. 5. Two blocks of smoke with viscosity is 10^{-5} are pushed toward each other. **Top:** The explicit spectral advection scheme from Lecoanet et al. [2019] blows up at frame 93, despite substepping with a timestep an order of magnitude smaller than ours. **Bottom:** Our semi-implicit advection achieves better stability, while maintaining a similar appearance.

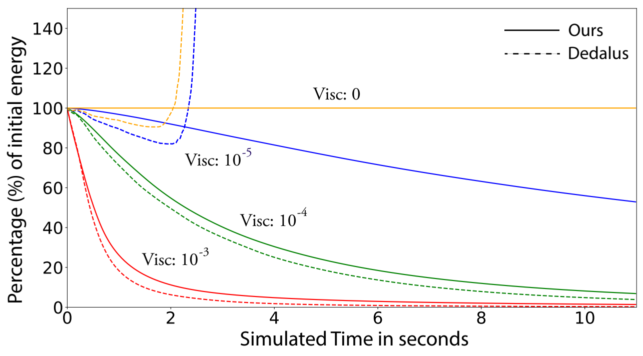


Fig. 6. Energy behavior of Fig. 5. The spectral approach of Lecoanet et al. [2019] becomes unstable at viscosity 10^{-5}, while our method maintains stability. In general, our method is also more energy-preserving.

is faster but more unstable than our semi-implicit approach. To establish a baseline, each simulation was given an equal amount of wall-clock time for computation. This resulted in a timestep size of 0.03s for our simulations, and a smaller 0.001s is used for the Dedalus simulations.

At larger viscosities, e.g., 10^{-3}, both yield plausible flows, but our method is consistently more energy-preserving (Fig. 6). As the viscosity decreases, the explicit advection scheme becomes increasingly unstable, and the Dedalus simulation diverges. At zero viscosity, we even tried decreasing the Dedalus timestep to 10^{-5}, but the simulation still diverged.

9 OPTIMIZATION AND IMPLEMENTATION

In this section, we describe several algorithmic optimizations and implementation details for spiral-spectral fluid simulation.

9.1 Efficient Transformations

Transforming between our basis functions and spatial velocity fields is the main computational cost of the simulation. A naïve approach would transform each basis individually and sum the results, but the functions have a shared structure that can be leveraged.

For simplicity, we discuss transforming the $\mathbf{u}_r$ component in polar coordinates (Fig. 7). The r component of $\mathbf{\Phi}_r^0 = \sin(i_1\pi r)\sin(i_2\theta)$ and $\mathbf{\Phi}_r^2 = \cos(i_1\pi r)\sin(\theta)$ in Eqn. 14 and Eqn. 20 share a DST in the θ direction. An efficient implementation applies a DST to $\mathbf{\Phi}_r^0$ and DCT to $\mathbf{\Phi}_r^2$ along the r direction. Then, a shared DST can be applied to their sum in the θ direction. This reconstructs $\mathbf{\Phi}_r^0$ and $\mathbf{\Phi}_r^2$ with three transforms instead of four. The basis functions $\mathbf{\Phi}_r^1$ in Eqn. 15 and $\mathbf{\Phi}_r^3$ in Eqn. 21 can be transformed in a similar manner. Finally, the full rank velocity $\mathbf{u}_r$ is obtained by adding the results together.

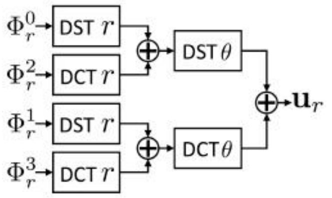


Fig. 7. Efficient polar velocity reconstruction first applies r-transforms to each basis function, then a shared θ transform to their sums.

This process can be combined with the *pruned* transforms from Cui et al. [2018]. In 2D, an $n \times n$ grid with s basis functions can be reconstructed with n transformations along the r direction. However, only $\sqrt{s}$ transforms are needed because the coefficients are located in a $\sqrt{s} \times \sqrt{s}$ square about the DC component. A rank-n transform is then applied in the θ direction. Four r-transforms and the first two summations from Fig. 7 are reduced to rank-$\sqrt{s}$, which can be computed on a $\sqrt{s} \times n$ grid.

The naïve approach needs eight rank-n transforms in Fig. 7, while our optimized version only needs two, reducing the cost by roughly 75% for polar functions. The improvements carry over to volumetric spherical basis, where both the r and θ transforms can be pruned. The r components from Eqn. 38 to Eqn. 41 only need two rank-n transformations, which saves 83% of the transforms. Force projections can be computed by reversing Fig. 7.

9.2 Efficient Advection Tensor Computation

Our basis functions were designed according to principle 2 from §4, specifically so that Eqn. 58 simplifies to a series of 1D integrals. Principle 1 further guarantees that the integrand only contains products of trigonometric functions (sometimes weighted by an integer-powered r) so each 1D integral can be computed analytically. Elliptical and spheroid basis functions are exceptions, because additional scale factors preclude an analytical solution. We instead efficiently compute the advection tensor semi-analytically by precomputing an intermediate lookup table of numerical integrals. Further details on this case are in the supplementary materials.

The basis functions $\mathbf{\Phi}_l$, $\mathbf{\Phi}_m$ and $\mathbf{\Phi}_n$ in Eqn. 58 pose a combinatoric challenge. The five different functions for polar coordinates results in 125 different combinations. We automate this process by first computing the different tensor expression combinations in Mathematica. The expressions are then separated along each direction, and the products of trigonometric functions are reduced to summations using TrigReduce. The expressions are then parsed into C++, and built-in functions evaluate the integrals of the trigonometric functions. With the exception of the elliptical and spheroid cases, we found the run-time cost of advection tensor computation to be similar to those reported in Cui et al. [2018].

9.3 Implementation Details

We implemented our algorithm in C++ and used FFTW3 [Frigo and Johnson 2005] for DCTs and DSTs. We multi-threaded with OpenMP whenever possible, including during particle advection, tensor contraction and basis transforms. Density was passively advected using Lagrangian particles integrated with RK4. The density particles were advected in Cartesian coordinates to avoid the need for a specialized density scheme [Hill and Henderson 2016; Huang et al. 2020; Yang et al. 2019]. This was achieved by interpolating the velocities from a parametric grid into Cartesian coordinates. In all our examples, the basis functions are first uniformly sampled along each wavenumber. During orthogonalization, we then reject basis functions already well-represented by existing components in order to ensure a well-conditioned transfer matrix **A**. This process is described further in Appendix A.

10 RESULTS

We now present a variety of results obtained using spiral-spectral fluid simulation over various geometries. Per-frame timing breakdowns of these examples are in Table 1, and were recorded on a server with 10 physical cores running at 2.6GHz. The timing breakdowns are comparable to those in Cui et al. [2018]. Similarly, the basis ranks are limited by the size of the advection tensors, shown in Table 2. While previous works [Cui et al. 2018; Hasan et al. 2008; Jones et al. 2016] suggest that significant compression of this tensor is possible, we left this to future work.

Our work is is compatible with procedurally defined external forces, which can be efficiently projected onto our basis functions. The results are obtained by first iterating with a fast, low-rank simulation. The workflow is similar to rectilinear solvers, but we have the additional feature that the user can start from the (somewhat foreign) inviscid regime, and then gradually increase the viscosity.

Completeness Test. We first examine the completeness of basis functions by attempting to reconstruct a spatial velocity field from a conventional Eulerian (Stam [1999]-type) simulation using our polar functions. As shown in Fig. 8, we start with three spatial velocity fields from different Eulerian simulations inside a disc (leftmost column). We then project this field both into and out of our polar basis, and measure the error. As the basis rank progressively increases to $s = 10K$, Fig. 9 shows error becoming less than 1%. While some residual error lingers around the border, this is due to the inability of the *Eulerian* simulation to capture the curved boundary, not a limitation of our basis. As expected, low-rank approximations of the spatial velocity look like low-pass filtered versions of the field.

Planetary Flow. In Fig. 10, we use our codimensional basis to simulate planetary flow along the surface of a sphere. The simulation is computed with $s = 8000$ basis functions on a 1024×2048 grid. Visually turbulent patterns reminiscent of the surface of Jupiter emerge.

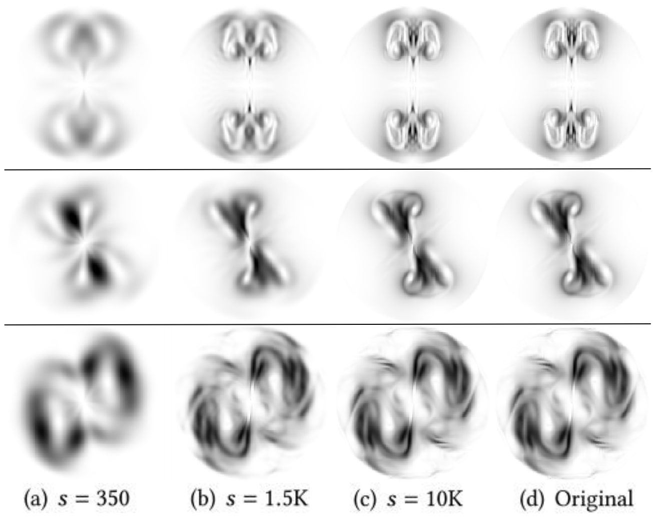


Fig. 8. Starting with a detailed flow from a Stam [1999]-type simulation (d), our basis reconstructs the flow as its rank increases. From top to bottom, the relative residual errors for the 10K examples (c) are: 0.6%, 0.1% and 0.1%.

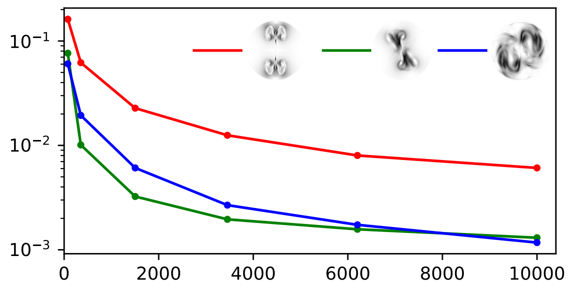
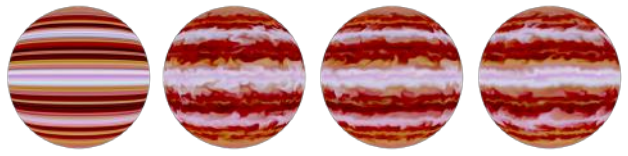
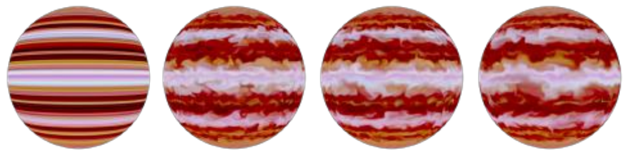


Fig. 9. Error convergence of Fig. 8 as more basis functions are added. Note the y-axis is a log scale.



(a) Planetary flow with horizontal forces.



(b) Planetary flow with Coriolis forces added as well.

Fig. 10. Planetary flow with $s = 8000$ basis functions. More turbulent appearances are obtained with Coriolis forces.

Atmospheric flows are usually driven by pressure gradients [Yaeger et al. 1986], but since our solve is pressure-free, we approximate its effects by injecting horizontal forces that align with the color bands. Coriolis forces are added in the bottom of Fig. 10, resulting in more turbulent flows. Each simulation frame is computed in 3.87 seconds.

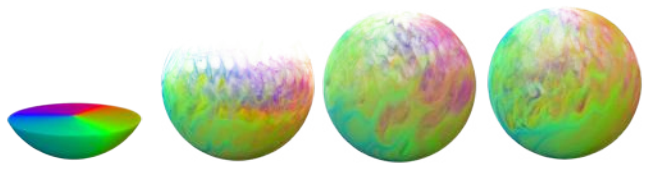


Fig. 11. Smoke Inside a Sphere with $s = 6386$ and Dirichlet boundaries.

Smoke Inside a Sphere. Next, we simulate smoke inside a sphere with Dirichlet boundaries in Fig. 11. The simulation is computed with $s = 6386$ on a $160 \times 320 \times 640$ parametric grid aligned in the r, θ, and ϕ directions. The motion is stimulated by injecting circular forces along the interior of the sphere. Turbulent details appear as smoke gradually fills the volume. Each frame takes 10.8 seconds.

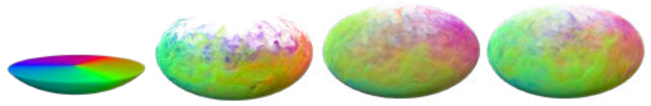


Fig. 12. Smoke Inside Spheroids Oblate spheroid with width 1.0, height 0.6, and $s = 6413$.

Smoke Inside Spheroids. The spherical basis functions extend to prolate and oblate spheroidal coordinates, enabling simulations inside spheroids. We show simulations (Fig. 12) with Dirichlet boundaries inside an $s = 6413$ oblate spheroid at 11.86s per frame. Additional spheroidal geometries are in the supplementary video.

Dropping Balls. We show that spiral-spectral fluids support obstacle interactions by dropping rigid balls into a $s = 11096$ fluid. The obstacle interacts with the fluid using penalty forces similar to Cui et al. [2018] and De Witt et al. [2012]. All boundaries are set to Dirichlet. As the highest-rank simulation we attempted, this example took 38.28 seconds per frame.

Cylindrical Tornado. We show a fire tornado in Fig. 19 swirling inside a cylindrical domain with $s = 6586$. Neumann boundaries are implemented along the top and bottom, allowing the fire to flow in and out, while the vertical wall is Dirichlet. We inject heat particles that linearly cool, and their temperatures are rendered as blackbody emitters. Buoyancy forces and a constant circulation force drives the simulation, which takes 11.98s per frame.

Plume in a Cylinder. In Fig. 15, we compare cylindrical flows using our methods to the conventional Eulerian implementation in Mantaflow [Thuerey and Pfaff 2018]. As in §8.2, both simulations were given similar wall-clock running times to facilitate the comparison. Both simulations produce similar overall motions, but the

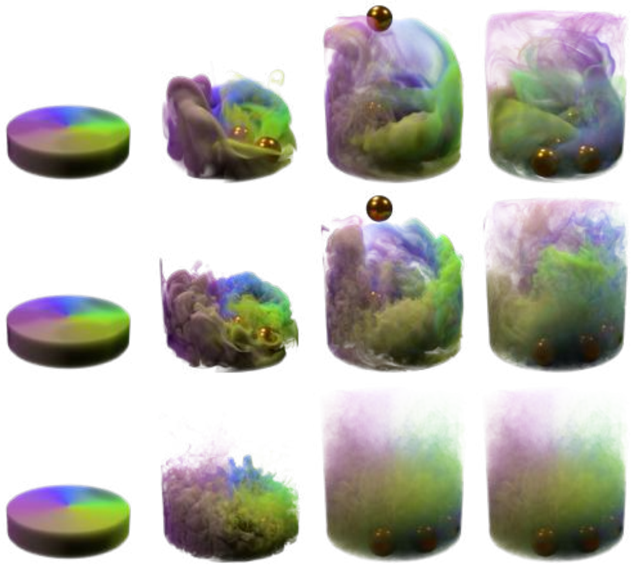


Fig. 13. Dropping Balls into $s = 11096$ cylindrical basis functions. From top row to bottom row, the viscosities are $\nu = 0.003$, $\nu = 0.001$, $\nu = 0.0001$. All boundaries are Dirichlet.

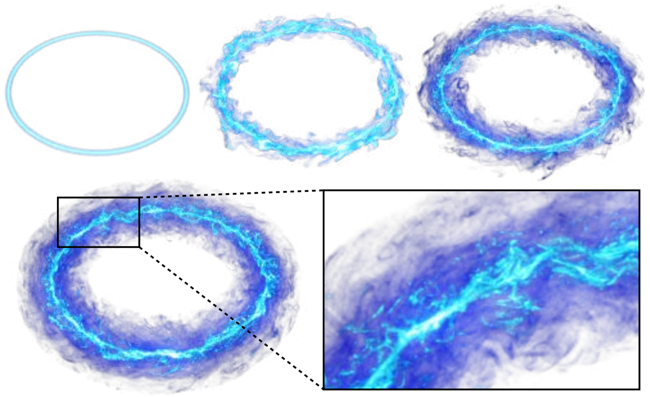


Fig. 14. Tokamak Flow with $s = 6269$ volumetric, toroidal basis functions and Dirichlet boundaries.

qualitative appearance of our approach is distinct, and provides a new "look" for artists to explore. In Fig. 16, more details appear in a higher-rank plume (s = 15381). In this case, the advection tensor hits the memory limit.

Tokamak Fluid. In Fig. 14, we approximate fluid traveling around a tokamak (fusion reactor) using a $s = 6269$ volumetric, toroidal basis. Heat particles are injected along the centerline and gradually cool down over time. To approximate a plasma-like visual effect (we do not solve the full magneto-hydro-dynamic equations), the heat particles are rendered as blackbodies with a bluish tint. A rotating force drives the simulation, and turbulence sheds off the centerline, creating the details highlighted in the inset. The simulation takes 6.74s per frame.

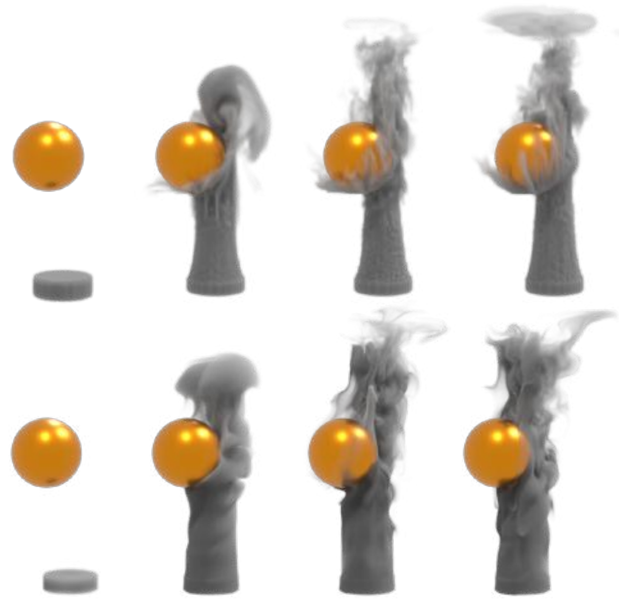


Fig. 15. Plume Cylinder. Top row: Smoke rising past a sphere using Mantaflow [Thuerey and Pfaff 2018]. Bottom: Same scene with our method, $s = 3397$, in cylindrical coordinates. The overall motions are similar, while ours achieves a novel look.

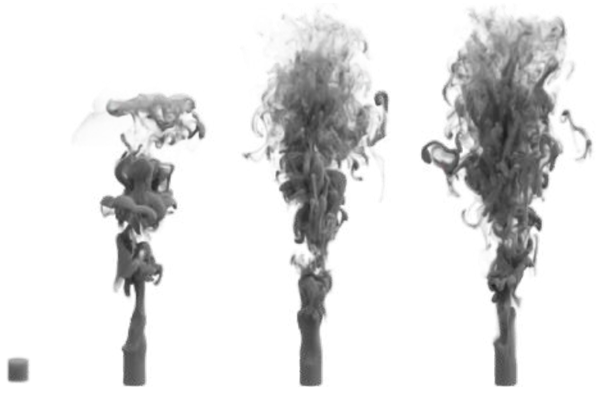
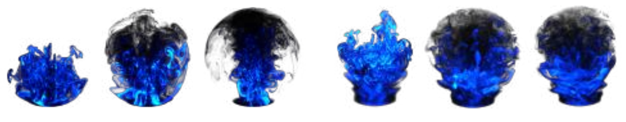


Fig. 16. A plume rising in cylindrical coordinates with $s = 15381$.



(a) Dirichlet Boundary (b) Neumann Boundary

Fig. 17. Sphere with $s = 6386$. Top row: With Dirichlet boundaries, smoke flows down the sides. Bottom row: In the same simulation with Neumann boundaries, it flows out of the domain.

Dirichlet/Neumann Spheres. In Fig. 17, we compare Neumann and Dirichlet boundary conditions, by simulating a flame rising from the bottom of a sphere with $s = 6386$. Heat is injected at the bottom of the sphere and rises from buoyancy. The fluid rebounds off the spherical ceiling with Dirichlet boundaries, and flows down the side. With Neumann boundaries, fluid can freely exit and enter. As a result, the fluid spreads more widely throughout the domain. The Dirichlet and Neumann simulations respectively take 9.7s and 10.2s per frame. We use homogeneous Neumann and Dirichlet boundary conditions in all our examples.

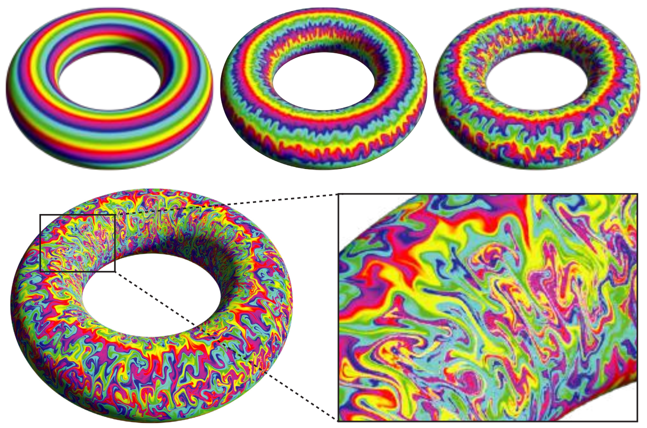


Fig. 18. Tie-Dye Torus with $s = 7175$ codimensional basis functions. Colored bands are initialized on the surface, and then pulled with gravity.

Tie-Dye Torus. In Fig. 18, we simulated $s = 7175$ codimensional, toroidal basis functions. Colored bands are initialized on the surface and then driven by gravity. Very small viscosity is used, which resulted in many small scale vortices, as shown in the inset.

11 CONCLUSION AND FUTURE WORK

We have presented the *spiral-spectral* method for simulating fluids over radial domains. Using our framework, we are able to generate basis functions, compute advection tensors, and simulate dynamics over these domains. We share the limitation from previous work, where advection tensor size limits the spectral resolution of the simulations. Other works have also shown that tensors [Cui et al. 2018; Hasan et al. 2008] and general fluid phenomena [Jones et al. 2016] can compress dramatically, so a detailed investigation of this could yield further scalability.

Another potential direction is the use of spiral-spectral spheres as enrichment particles for existing simulations. In contrast to existing noise-based approaches [Narain et al. 2008; Pfaff et al. 2009; Selle et al. 2005], a detailed physics model is built in. Further work would be needed to account for phenomena such as vortex stretching. Finally, the extension of this spectral approach to other parametric surfaces is an interesting future direction, but it remains to be seen if such approaches could remain analytical.

ACKNOWLEDGMENTS

This work was supported by a funding gift from Adobe Research. We thank the Yale Center for Research Computing for guidance and use of the research computing infrastructure, specifically Grace.

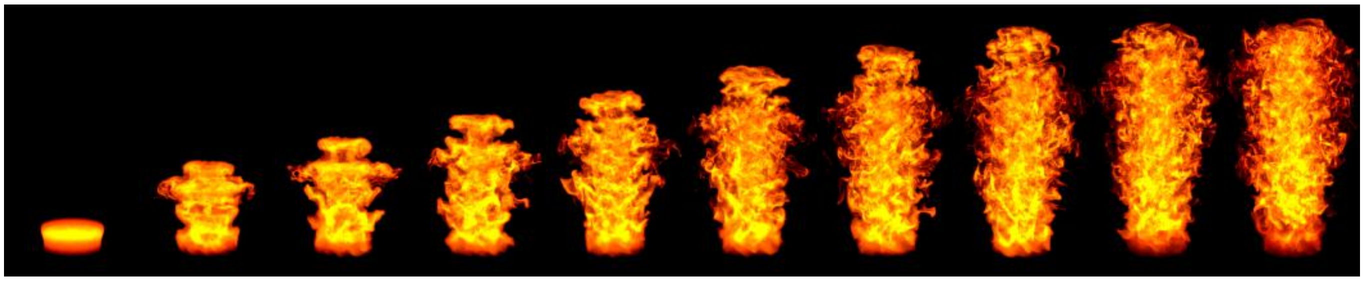


Fig. 19. CYLINDRICAL TORNADO example with $s = 6586$ basis functions. Neumann boundary conditions are enabled on the top and bottom. Heat particles are gradually injected at the bottom, and rendered with blackbody radiation.

Table 1. Per-frame timing breakdown of our algorithm across all the different examples. The parametric grid‡ is discretized along the r, θ and ϕ coordinates, and basis transforms occur on this grid. The swirling smoke examples are simulated inside a unit sphere*, a prolate spheroid†, and an oblate spheroid*.

Scene	Grid‡ Resolution	Basis Dimension	Tensor Contraction	Linear Solver	Diffusion Solver	DCT/DST	Density Advection	Total
PLANETARY FLOW	1024×2048	8000	2.08 secs	0.85 secs	0.04 secs	0.16 secs	0.74 secs	3.87 secs
SWIRLING SMOKE	$160 \times 320 \times 640$	6386*	2.20 secs	0.56 secs	0.05 secs	4.97 secs	3.02 secs	10.80 secs
		6384†	2.02 secs	0.70 secs	0.05 secs	4.28 secs	3.02 secs	10.07 secs
		6413*	2.53 secs	0.79 secs	0.05 secs	4.92 secs	3.57 secs	11.86 secs
CYLINDER TORNADO	$150 \times 300 \times 600$	6586	2.43 secs	1.33 secs	0.05 secs	4.06 secs	4.11 secs	11.98 secs
DROPPING BALLS	$160 \times 320 \times 640$	11096	3.85 secs	1.43 secs	0.15 secs	30.20 secs	2.65 secs	38.28 secs
TOKAMAK FLOW	$128 \times 256 \times 512$	6269	2.06 secs	0.44 secs	0.05 secs	1.87 secs	2.32 secs	6.74 secs
TIE-DYE TORUS	1024×2048	7175	1.68 secs	0.76 secs	0.06 secs	0.25 secs	1.52 secs	4.27 secs
PLUME IN A CYLINDER	$80 \times 160 \times 320$	4930	0.45 secs	0.20 secs	0.01 secs	0.80 secs	0.22 secs	1.68 secs
		15381	6.32 secs	3.31 secs	0.30 secs	4.28 secs	3.02 secs	17.23 secs

Table 2. The advection tensor size across different examples. The geometry marked by * is computed with Neumann boundaries on both endcaps, while the geometry marked by † is computed with all-Dirichlet boundaries.

Scene	Geometry	Basis Dimension	Tensor Size
PLANETARY FLOW	sphere surface	8000	34.7 GB
SWIRLING SMOKE	sphere	6386	29.8 GB
	prolate spheroid	6384	41.1 GB
	oblate spheroid	6413	39.7 GB
CYLINDER TORNADO	cylinder*	6586	73.2 GB
DROPPING BALLS	cylinder†	11096	30.7 GB
PLUME IN A CYLINDER	cylinder†	4930	5.9 GB
	cylinder†	15381	74.1 GB
TOKAMAK FLOW	torus	6269	43.7 GB
TIE-DYE TORUS	torus surface	7175	17.3 GB

REFERENCES

Alexis Angelidis. 2017. Multi-Scale Vorticle Fluids. *ACM Trans. Graph.* 36, 4, Article 104 (July 2017), 12 pages.

Alexis Angelidis and Fabrice Neyret. 2005. Simulation of Smoke Based on Vortex Filament Primitives. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation.* 87–96.

Ferdinand Baer and George W Platzman. 1961. A Procedure for Numerical Integration of the Spectral Vorticity Equation. *Journal of Meteorology* 18, 3 (Jun 1961), 393–401.

Jernej Barbič and Doug L. James. 2005. Real-Time Subspace Integration for St. Venant-Kirchhoff Deformable Models. *ACM Trans. Graph.* 24, 3 (July 2005), 982–990.

Rubén G Barrera, GA Estevez, and J Giraldo. 1985. Vector spherical harmonics and their application to magnetostatics. *European Journal of Physics* 6, 4 (1985), 287.

John P Boyd. 1978. The Choice of Spectral Functions on a Sphere for Boundary and Eigenvalue Problems: A Comparison of Chebyshev, Fourier and Associated Legendre Expansions. *Mon. Weather Rev.* 106, 8 (1978), 1184 – 1191.

John P Boyd. 2001. *Chebyshev and Fourier spectral methods.* Dover Publications.

John P Boyd and Fu Yu. 2011. Comparing seven spectral methods for interpolation and for solving the Poisson equation in a disk: Zernike polynomials, Logan–Shepp ridge polynomials, Chebyshev–Fourier series, cylindrical Robert functions, Bessel–Fourier expansions, square-to-disk conformal mapping and radial basis functions. *J. Comput. Phys.* 230, 4 (2011), 1408–1438.

Robert Bridson. 2015. *Fluid simulation for computer graphics.* CRC Press.

Keaton J Burns, Geoffrey M Vasil, Jeffrey S Oishi, Daniel Lecoanet, and Benjamin P Brown. 2020. Dedalus: A flexible framework for numerical simulations with spectral methods. *Physical Review Research* 2, 2 (2020), 023068.

Qiaodong Cui, Pradeep Sen, and Theodore Kim. 2018. Scalable Laplacian eigenfluids. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–12.

Tyler De Witt, Christian Lessig, and Eugene Fiume. 2012. Fluid Simulation Using Laplacian Eigenfunctions. *ACM Trans. Graph.* 31, 1, Article 10 (2012), 11 pages.

John B Drake. 2014. *Climate modeling for scientists and engineers.* SIAM.

Stefen Fangmeier. 1996. Designing digital tornados. *Comput. Graph. World* 19, 8 (1996), 58.

Matteo Frigo and Steven G. Johnson. 2005. The Design and Implementation of FFTW3. *Proc. IEEE* 93, 2 (2005), 216–231. Special issue on "Program Generation, Optimization, and Platform Adaptation".

Dan Gerszewski, Ladislav Kavan, Peter-Pike Sloan, and Adam W. Bargteil. 2015. Basis Enrichment and Solid-fluid Coupling for Model-reduced Fluid Simulation. *Computer Animation and Virtual Worlds* 26 (Mar/Apr 2015).

Gene H Golub and Charles F Van Loan. 2012. *Matrix computations.* Vol. 3. JHU Press.

Denis S Grebenkov and B-T Nguyen. 2013. Geometrical structure of Laplacian eigenfunctions. *siam REVIEW* 55, 4 (2013), 601–667.

Michael Griebel, Thomas Dornseifer, and Tilman Neunhoeffer. 1998. *Numerical simulation in fluid dynamics: a practical introduction.* SIAM.

Mohit Gupta and Srinivasa G. Narasimhan. 2007. Legendre Fluids: A Unified Framework for Analytic Reduced Space Modeling and Rendering of Participating Media. In *Symposium on Computer Animation.* 17–25.

Jonah Hall. 2004. The Day After Tomorrow Twister Sequence Toolkit: A Volumetric Tornado Pipeline. In *ACM SIGGRAPH Sketches* (Los Angeles, California). 143.

Milos Hasan, Edgar Velazquez-Armendariz, Fabio Pellacini, and Kavita Bala. 2008. Tensor Clustering for Rendering Many-Light Animations. *Computer Graphics Forum* 27, 4 (2008), 1105–1114.

Ronald D. Henderson. 2012. Scalable Fluid Simulation in Linear Time on Shared Memory Multiprocessors. In *Proceedings of the Digital Production Symposium.* 43–52.

David J Hill and Ronald D Henderson. 2016. Efficient fluid simulation on the surface of a sphere. *ACM Transactions on Graphics (TOG)* 35, 2 (2016), 1–9.

EL Hill. 1954. The theory of vector spherical harmonics. *American Journal of Physics* 22, 4 (1954), 211–214.

Weizhen Huang, Julian Iseringhausen, Tom Kneiphof, Ziyin Qu, Chenfanfu Jiang, and Matthias B Hullin. 2020. Chemomechanical simulation of soap film flow on spherical bubbles. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 41–1.

Aaron Demby Jones, Pradeep Sen, and Theodore Kim. 2016. Compressing Fluid Subspaces. In *Symposium on Computer Animation.* 77–84.

Florian Karpfinger, Boris Gurevich, and Andrey Bakulin. 2008. Modeling of wave dispersion along cylindrical structures using the spectral method. *The Journal of the Acoustical Society of America* 124, 2 (2008), 859–865.

Doyub Kim. 2017. *Fluid engine development.* CRC Press.

Theodore Kim and John Delaney. 2013. Subspace Fluid Re-simulation. *ACM Trans. Graph.* 32, 4, Article 62 (2013), 9 pages.

Granino Arthur Korn and Theresa M Korn. 2000. *Mathematical handbook for scientists and engineers: definitions, theorems, and formulas for reference and review.* Courier Corporation.

Jakob Kühnen, Baofang Song, Davide Scarselli, Nazmi Burak Budanur, Michael Riedl, Ashley P Willis, Marc Avila, and Björn Hof. 2018. Destabilizing turbulence in pipe flow. *Nature Physics* 14, 4 (2018), 386–390.

Daniel Lecoanet, Geoffrey M Vasil, Keaton J Burns, Benjamin P Brown, and Jeffrey S Oishi. 2019. Tensor calculus in spherical coordinates using Jacobi polynomials. Part-II: Implementation and examples. *J. Comput. Phys: X* 3 (2019), 100012.

Christian Lessig. 2019. Divergence Free Polar Wavelets for the Analysis and Representation of Fluid Flows. *J. Math. Fluid Mech.* 21, 18 (Feb 2019).

Beibei Liu, Gemma Mason, Julian Hodgson, Yiying Tong, and Mathieu Desbrun. 2015. Model-reduced Variational Fluid Simulation. *ACM Trans. Graph.* 34, 6, Article 244 (2015), 12 pages.

Benjamin Long and Erik Reinhard. 2009. Real-time Fluid Simulation Using Discrete Sine/Cosine Transforms. In *Symp. on Interactive 3D Graphics and Games.* 99–106.

Olivier Mercier and Derek Nowrouzezahrai. 2020. Local Bases for Model-reduced Smoke Simulations. In *Comput. Graph. Forum*, Vol. 39. Wiley Online Library, 9–22.

Martin J Mohlenkamp. 1999. A fast transform for spherical harmonics. *Journal of Fourier analysis and applications* 5, 2-3 (1999), 159–184.

Youhei Morinishi, Oleg V Vasilyev, and Takeshi Ogi. 2004. Fully conservative finite difference scheme in cylindrical coordinates for incompressible flow simulations. *J. Comput. Phys.* 197, 2 (2004), 686–710.

Matthias Müller, David Charypar, and Markus Gross. 2003. Particle-Based Fluid Simulation for Interactive Applications. In *Proc. of Symposium on Computer Animation*, D. Breen and M. Lin (Eds.).

Rahul Narain, Jason Sewall, Mark Carlson, and Ming C. Lin. 2008. Fast Animation of Turbulence Using Energy Transport and Procedural Synthesis. *ACM Trans. Graph.* 27, 5, Article 166 (Dec. 2008), 8 pages.

Steven A Orszag. 1974. Fourier Series on Spheres. *Mon. Weather Rev.* 102, 1 (1974), 56 – 75.

Marcel Padilla, Albert Chern, Felix Knöppel, Ulrich Pinkall, and Peter Schröder. 2019. On Bubble Rings and Ink Chandeliers. *ACM Trans. Graph.* 38, 4, Article 129 (July 2019), 14 pages.

Sang Il Park and Myoung Jun Kim. 2005. Vortex fluid for gaseous phenomena. In *Proc. of the ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.* 261–270.

Tobias Pfaff, Nils Thuerey, Jonathan Cohen, Sarah Tariq, and Markus Gross. 2010. Scalable Fluid Simulation Using Anisotropic Turbulence Particles. *ACM Trans. Graph.* 29, 6, Article 174 (Dec. 2010), 8 pages.

Tobias Pfaff, Nils Thuerey, Andrew Selle, and Markus Gross. 2009. Synthetic Turbulence Using Artificial Boundary Layers. *ACM Trans. Graph.* 28, 5 (Dec. 2009), 1–10.

Nick Rasmussen, Duc Quang Nguyen, Willi Geiger, and Ronald Fedkiw. 2003. Smoke Simulation for Large Scale Phenomena. *ACM Trans. Graph.* 22, 3 (July 2003), 703–707.

Osborne Reynolds. 1883. XXIX. An experimental investigation of the circumstances which determine whether the motion of water shall be direct or sinuous, and of the law of resistance in parallel channels. *Philosophical Transactions of the Royal society of London* 174 (1883), 935–982.

Yousef Saad. 2003. *Iterative methods for sparse linear systems.* SIAM.

Andrew Selle, Nick Rasmussen, and Ronald Fedkiw. 2005. A Vortex Particle Method for Smoke, Water and Explosions. *ACM Trans. Graph.* 24, 3 (July 2005), 910–914.

Isadore Silberman. 1954. Planetary Waves in the Atmosphere. *Journal of Meteorology* 1 (Feb 1954), 27–34.

Richard Mikael Slevinsky, Hadrien Montanelli, and Qiang Du. 2018. A spectral method for nonlocal diffusion operators on the sphere. *J. Comput. Phys.* 372 (2018), 893–911.

J. Stam. 1999. Stable Fluids. In *Proceedings of SIGGRAPH.* 121–128.

J. Stam. 2002. A Simple Fluid Solver Based on the FFT. *J. Graph. Tools* 6, 2 (Sept. 2002), 43–52.

J. Stam. 2003. Flows on surfaces of arbitrary topology. *ACM Trans. Graph.* 22, 3 (2003), 724–731.

Matt Stanton, Yu Sheng, Martin Wicke, Federico Perazzi, Amos Yuen, Srinivasa Narasimhan, and Adrien Treuille. 2013. Non-polynomial Galerkin Projection on Deforming Meshes. *ACM Trans. Graph.* 32, 4, Article 86 (July 2013), 14 pages.

Alexey Stomakhin, Craig Schroeder, Lawrence Chai, Joseph Teran, and Andrew Selle. 2013. A Material Point Method for Snow Simulation. *ACM Trans. Graph.* 32, 4, Article 102 (July 2013), 10 pages.

P. Tamain, H. Bufferand, G. Ciraolo, C. Colin, D. Galassi, Ph. Ghendrih, F. Schwander, and E. Serre. 2016. The TOKAM3X code for edge turbulence fluid simulations of tokamak plasmas in versatile magnetic geometries. *J. Comput. Phys.* 321 (2016), 606 – 623.

Demetri Terzopoulos and Andrew Witkin. 1988. Physically based models with rigid and deformable components. *IEEE CG&A* 8, 6 (1988), 41–51.

Nils Thuerey and Tobias Pfaff. 2018. MantaFlow. *http://mantaflow.com.*

Adrien Treuille, Andrew Lewis, and Zoran Popović. 2006. Model Reduction for Real-time Fluids. *ACM Trans. Graph.* 25, 3 (July 2006), 826–834.

Geoffrey M Vasil, Daniel Lecoanet, Keaton J Burns, Jeffrey S Oishi, and Benjamin P Brown. 2019. Tensor calculus in spherical coordinates using Jacobi polynomials. Part-I: Mathematical analysis and derivations. *J. Comput. Phys: X* 3 (2019), 100013.

N. P. Wedi, M. Hamrud, and G. Mozdzynski. 2013. A Fast Spherical Harmonics Transform for Global NWP and Climate Models. *Mon. Weather Rev.* 141, 10 (oct 2013), 3450–3461.

Steffen Weissmann, Ulrich Pinkall, and Peter Schröder. 2014. Smoke Rings from Smoke. 4, Article 140 (July 2014), 8 pages.

Larry Yaeger, Craig Upson, and Robert Myers. 1986. Combining Physical and Visual Simulation—Creation of the Planet Jupiter for the Film "2010". In *Proceedings of SIGGRAPH.* 85–93.

Bowen Yang, William Corse, Jiecong Lu, Joshuah Wolper, and Chen-Fanfu Jiang. 2019. Real-Time Fluid Simulation on the Surface of a Sphere. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 2, 1 (2019), 1–17.

Can Yuksel, Domin Lee, and Ronald Henderson. 2012. Vortex of Awesomeness. In *ACM SIGGRAPH Talks* (Los Angeles, California).

Ben Zhu, Manaure Francisquez, and Barrett N. Rogers. 2018. GDB: A global 3D two-fluid model of plasma turbulence and transport in the tokamak edge. *Computer Physics Communications* 232 (2018), 46 – 58.

Yongning Zhu and Robert Bridson. 2005. Animating Sand As a Fluid. *ACM Trans. Graph.* 24, 3 (July 2005), 965–972.

A GRAM-SCHMIDT PROCESS

We describe an analytic version of the modified Gram-Schmidt process [Golub and Van Loan 2012] that we use in §7.1. The process is as follows. Above, the dot products $\left\langle \boldsymbol{\varphi}_k^{j-1}, \mathbf{v}_j \right\rangle$ are computed analytically. These coefficient are stored in the triangular transfer matrix $\mathbf{A}$, analogous to the $\mathbf{R}$ matrix in a QR factorization, and transforms the input basis functions to orthogonality:

$$\mathbf{A} = \begin{bmatrix} \frac{1}{|\Phi_1|} & 0 & 0 & \dots & 0 \\ \mathbf{A}_{21} & \frac{1}{|\boldsymbol{\varphi}_2^1|} & 0 & \dots & 0 \\ \mathbf{A}_{31} & \mathbf{A}_{32} & \frac{1}{|\boldsymbol{\varphi}_3^2|} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}_{i,1} & \mathbf{A}_{i2} & \mathbf{A}_{i3} & \dots & \frac{1}{|\boldsymbol{\varphi}_k^i|} \end{bmatrix} \tag{65}$$

The norm inverses appear along the diagonal, and to keep $\mathbf{A}$ well-conditioned, we set a threshold τ to filter basis function with very small remainders. We experimentally found $\tau = 0.2$ worked well. With no filtering, i.e., $\tau = 0$ in the planetary flow example, 69 more basis functions are retained, resulting in $s = 8069$. However, the

Algorithm 1 Gram Schmidt Orthogonalization

```
Input: Set of analytical basis functions $\{\mathbf{\Phi}_1, \cdots, \mathbf{\Phi}_k\}$
Output: Set of orthogonal basis functions $\{\mathbf{v}_1, \cdots, \mathbf{v}_i\}$
1:  procedure ORTHOGONALIZATION
2:      $\mathbf{v}_1 \leftarrow \frac{\mathbf{\Phi}_1}{|\mathbf{\Phi}_1|}$
3:      $i \leftarrow 1$
4:      for $k$ from 2 to $n$ do
5:          $\boldsymbol{\varphi}_k^0 \leftarrow \mathbf{\Phi}_k$
6:          for $j$ from 1 to $i$ do
7:              $\boldsymbol{\varphi}_k^j \leftarrow \boldsymbol{\varphi}_k^{j-1} - \left\langle \boldsymbol{\varphi}_k^{j-1}, \mathbf{v}_j \right\rangle \mathbf{v}_j$
8:          end for
9:          if $|\boldsymbol{\varphi}_k^i| > \tau$ then
10:             $i \leftarrow i + 1$
11:             $\mathbf{v}_i \leftarrow \frac{\boldsymbol{\varphi}_k^i}{|\boldsymbol{\varphi}_k^i|}$
12:         end if
13:     end for
14: end procedure
```

linear solver becomes five times slower because the transfer matrix $\mathbf{A}$ becomes less well-conditioned. We verified that the algorithm stably produces a $\mathbf{A}$ that diagonalizes the inner product matrix to within near-floating-point precision (10^{-12}).

In practice, we found $\mathbf{A}$ to be block-sparse, because most basis functions are already orthogonal, e.g. the bases $\mathbf{\Phi}_*^0$ (Eqn. 14) and $\mathbf{\Phi}_*^1$ (Eqn. 15) are orthogonal. Within each set, basis functions are orthogonal if they have different wavenumber along the θ direction. This is because the following integrals are computed along θ:

$$\int_{\theta=0}^{2\pi} \sin(i_2\theta)\sin(j_2\theta) = \pi\delta_{i_2,j_2} \qquad \int_{\theta=0}^{2\pi} \cos(i_2\theta)\cos(j_2\theta) = \pi\delta_{i_2,j_2}.$$

Therefore, any basis functions with different wavenumbers along θ are orthogonal. In practice, we group basis functions by their wavenumbers along θ, and orthogonalize within each group. The transfer matrix $\mathbf{A}$ is then assembled from the sub-matrices from each group. This approach extends to volumetric bases as well, and we observed that only 0.1% to 0.5% entries of matrix $\mathbf{A}$ were non-zero.

B EXTENSION TO ELLIPTIC COORDINATES

We extend the polar basis functions from §5 to elliptical coordinates. First, we extend the polar coordinates in Eqn. 9 to elliptical,

$$\mathbf{p}_x = a\sqrt{1+c^2r^2}\cos(\theta) \qquad \mathbf{p}_y = acr\sin(\theta), \tag{66}$$

where $a = \sqrt{1-b^2}$, the length of major axis is 1, the minor axis is b, the scalar $c = b/a$, and the parameter ranges are $r \in [0,1]$, $\theta \in [0, 2\pi)$. This converges to polar coordinates as $b \longrightarrow 1$, and allows us to find elliptic basis functions that exactly converge to the polar basis functions. The divergence operator becomes

$$\nabla \cdot \mathbf{s} = \frac{1}{h_r h_\theta}\left[\frac{\partial}{\partial r}(\mathbf{s}_r h_\theta) + \frac{\partial}{\partial \theta}(\mathbf{s}_\theta h_r)\right], \tag{67}$$

where the scale factors are

$$h_r = ac\frac{h}{\sqrt{1+c^2r^2}} \qquad h_\theta = ah, \tag{68}$$

and $h = \sqrt{c^2r^2 + \sin^2(\theta)}$.

As $b \longrightarrow 1$, the h_θ becomes r and h_r becomes 1, which establishes that the elliptic divergence operator (Eqn. 67) converges to the polar case (Eqn. 10). We use $\mathbf{\Psi}$ to denote the elliptic basis functions. Given a divergence-free basis $\mathbf{\Phi}$ in polar coordinates, it can be converted to the elliptic functions $\mathbf{\Psi}$ as follows:

$$\begin{cases} \mathbf{\Psi}_r = \frac{cr}{h}\mathbf{\Phi}_r, \\ \mathbf{\Psi}_\theta = \frac{\sqrt{1+c^2r^2}}{h}\mathbf{\Phi}_\theta. \end{cases} \tag{69}$$

By inserting $\mathbf{\Psi}_r$ and $\mathbf{\Psi}_\theta$ as $\mathbf{s}_r$ and $\mathbf{s}_\theta$ into Eqn. 67, the elliptic scale factors h_r and h_θ canceled out, and leave the polar divergence operator with $\mathbf{\Phi}_r$ and $\mathbf{\Phi}_\theta$. Thus, it should be zero because $\mathbf{\Phi}_r$ and $\mathbf{\Phi}_\theta$ are divergence-free in polar coordinates. The only exception is the enrichment function $\mathbf{\Psi}_*^2$, where the following basis is preferred:

$$\begin{cases} \mathbf{\Psi}_r^2 &= \frac{1}{h}\sqrt{1+c^2r^2}\cos(i_1\pi r)\sin(\theta), \\ \mathbf{\Psi}_\theta^2 &= \frac{1}{h}\left(cr\cos(i_1\pi r) - \frac{i_1}{c}\pi(1+c^2r^2)\sin(i_1\pi r)\right)\cos(\theta). \end{cases} \tag{70}$$

This is because the velocity transform in elliptical coordinates is

$$\begin{bmatrix} \mathbf{u}_x \\ \mathbf{u}_y \end{bmatrix} = \begin{bmatrix} \frac{cr\cos(\theta)}{h} & -\frac{\sqrt{1+c^2r^2}\sin(\theta)}{h} \\ \frac{\sqrt{1+c^2r^2}\sin(\theta)}{h} & \frac{cr\cos(\theta)}{h} \end{bmatrix} \begin{bmatrix} \mathbf{u}_r \\ \mathbf{u}_\theta \end{bmatrix}, \tag{71}$$

and when $i_1 = 0$, Eqn. 70 evaluates to $\mathbf{\Psi}_r^2 = \frac{1}{h}\sqrt{1+c^2r^2}\sin(\theta)$ and $\mathbf{\Psi}_\theta^2 = \frac{1}{h}cr\cos(\theta)$. By transforming the velocity to Cartesian coordinates with Eqn. 71, we obtain a y-translation: $\mathbf{u}_x = 0, \mathbf{u}_y = 1$. Therefore, it is able to represent the same translation motion as $\mathbf{\Phi}_*^2$ in Eqn. 20, which is not possible by directly applying Eqn. 69 to $\mathbf{\Phi}_*^2$ in polar coordinates.

Limitations. Due to the scalar c appearing in h, the basis functions depend on the shape of the ellipse. Thus, the dot products in the advection tensor must be recomputed if c changes. Another limitation is that velocity derivative conditions at the border become approximate. A Neumann boundary can be achieved in polar coordinates by using an integer wavenumber offset of $1/2$ in $\mathbf{\Phi}_*^0$ from Eqn. 14, but this is no longer holds in elliptical coordinates, because the derivative of $\mathbf{\Psi}_r^0$ from Eqn. 69 becomes:

$$\mathbf{\Psi}_r^0 = \frac{cr\sin(i_1\pi r)\sin(i_2\theta)}{h}, \tag{72}$$

$$\frac{\partial \mathbf{\Psi}_r^0}{\partial r} = \left[-\frac{c^3r^2\sin(i_1\pi r)}{h^3} + \frac{ci_1\pi r\cos(i_1\pi r)}{h} + \frac{c\sin(i_1\pi r)}{h}\right]\sin(i_2\theta). \tag{73}$$

Both $\sin(i_1\pi r)$ and $\cos(i_1\pi r)$ appear in the derivative, so it cannot be constrained to zero exactly, but remains bounded. If we assign then $i_1 \in \mathbb{Z}^+ - 1/2$, then $\cos(i_1\pi r)$ vanishes at $r = 1$:

$$\frac{\partial \mathbf{\Psi}_r^0}{\partial r}(r=1) = \frac{c\sin^2(\theta)}{\sqrt[3]{c^2+\sin^2(\theta)}}\sin(i_1\pi)\sin(i_2\theta). \tag{74}$$

The absolute value of the derivative is bounded by:

$$\left|\frac{\partial \mathbf{\Psi}_r^0}{\partial r}(r=1)\right| \le \frac{c\sin^2(\theta)}{\sqrt[3]{c^2+\sin^2(\theta)}}. \tag{75}$$

The maximum value is obtained at $\sin^2(\theta) = 2c^2$ when $2c^2 \le 1$, where above equation evaluates to $2/\sqrt[3]{1+2} \approx 0.3849$. When $2c^2 >$

1, the maximum is obtained at $\sin^2(\theta) = 1$, where the above equation evaluates to $c/\sqrt[3]{c^2+1}$. Thus, the velocity derivative is bounded between ± 0.3849. This approximation only applies to Neumann conditions; Dirichlet are satisfied exactly.