EUROGRAPHICS 2020 / U. Assarsson and D. Panozzo
(Guest Editors)

Volume 39 (2020), Number 2

Local Bases for Model-reduced Smoke Simulations

Olivier Mercier[1] and Derek Nowrouzezahrai[2]

[1]Université de Montréal [2]McGill University

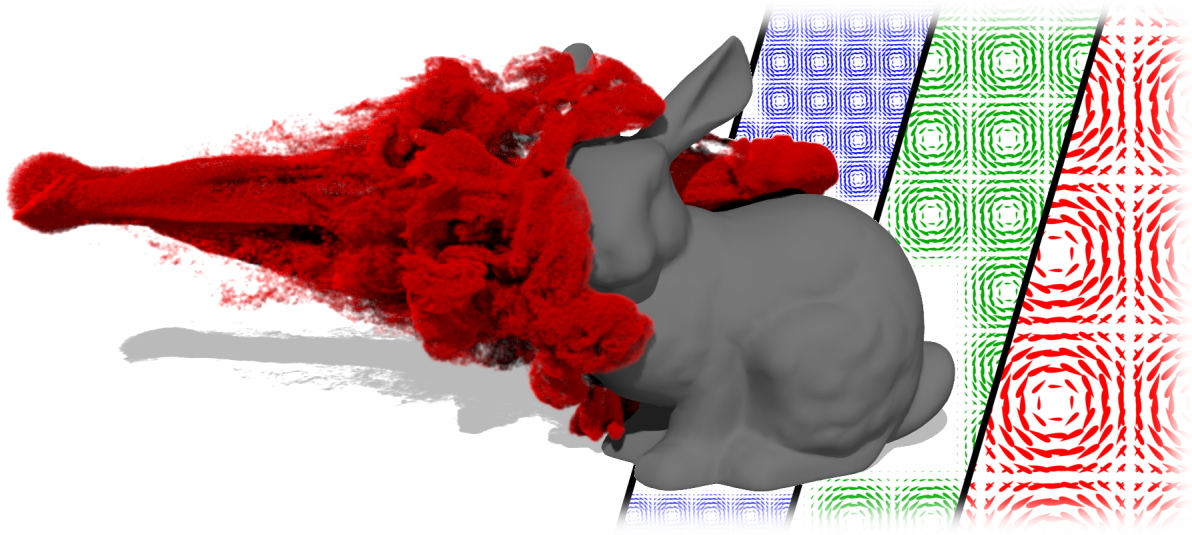

Figure 1: *We advect smoke particles using our model-reduced, multiresolution representation of the underlying fluid dynamics. Each basis flow (visualized in simplified form, on the right) has local support, which permits an adaptive representation of the fluid dynamics across scales over the simulation domain. Our basis is efficient to construct, apply and evaluate, it handles dynamic obstacles and curved boundaries, and it allows flexible user control.*

Abstract

We present a flexible model reduction method for simulating incompressible fluids. We derive a novel vector field basis composed of localized basis flows which have simple analytic forms and can be tiled on regular lattices, avoiding the use of complicated data structures or neighborhood queries. Local basis flow interactions can be precomputed and reused to simulate fluid dynamics on any simulation domain without additional overhead. We introduce heuristic simulation dynamics tailored to our basis and derived from a projection of the Navier-Stokes equations to produce physically plausible motion, exposing intuitive parameters to control energy distribution across scales. Our basis can adapt to curved simulation boundaries, can be coupled with dynamic obstacles, and offers simple adjustable trade-offs between speed and visual resolution.

CCS Concepts

• ***Computing methodologies*** → ***Physical simulation;***

1. Introduction

Realistic fluid simulation remains a fundamental challenge in computer graphics. Complex and intricate fluid features appear across spatial scales, and reproducing these detailed dynamics on uniformly discretized domains requires prohibitively large resolutions. Multiresolution methods can reduce this limitation by adapting the spatial and temporal simulation resolution, focusing computation time on appropriate regions of interest, e.g., regions where a fluid evolves into finer-scale structures or regions where a viewer's attention is more likely to be drawn.

Fluid simulation on non-uniform grids, such as octrees, is one common multiresolution approach used in graphics: given a uniform simulation grid, each cell is repeatedly subdivided until the required simulation resolution is reached. Implementing these ap-

proaches in a robust manner requires care, as interactions between cells across simulation scales can become non-trivial. Wavelet-based methods are another alternative, relying on localized multi-scale representations of the underlying dynamics. While these methods result in more compact representations of the simulation quantities, converting between wavelet and primal-domain representations often creates a computational bottleneck.

Our work is based, instead, on a *model reduction* methodology: simulated quantities are represented as weighted combinations of basis vector fields and the underlying dynamics are reformulated as operators that act on this *reduced* representation. The utility of these methods relies heavily on basis properties, and optimal basis design for model-reduced dynamics remains an open problem.

We present a novel basis suitable for multi-resolution fluid simulations. Our contributions include:

- a simple method to construct anisotropic vector field basis functions with local support and important orthogonality properties,
- flexible tiling strategies to cover arbitrary simulation domains without any basis recomputation, and
- an efficient, stable simulation algorithm that uses localized basis interactions and provides control over turbulent energy cascades.

2. Previous Work

Foster and Metaxas [FM96] and Stam [Sta99] pioneered efficient fluid simulation in computer graphics by solving a discretization of the dynamics on uniform grids. However, these uniform discretizations do not scale well to the requirements of modern high-fidelity fluid simulations. We outline below the various strategies used to accelerate fluid simulation in this context.

Model Reduction. The infinite-dimensional space of all possible vector fields can be reduced to linear combinations of specially-chosen basis fields. This high-level *model reduction* principle has been applied to many problems in computer graphics, including character animation [PW89, KRFC09], cloth simulation [HTC*14], deformation [BJ05] and global illumination [SSW*13].

Treuille et al. [TLP06] introduced model reduction for fluids to computer graphics using vector field basis functions constructed from SVD decompositions of full-space simulation data. Their divergence-free basis satisfies boundary conditions, but the need for full-space simulation constrains its use to re-simulations in a single, fixed domain. Improvements allow for fluid parameter variations [KD13] and limited domain deformations [SSW*13], but the precomputed full-space simulation constraint remains.

Wicke et al. [WST09] improve basis reusability by precomputing modular, reconfigurable flows. Gerszewski et al. [GKSB15] instead enrich a set of existing basis functions for task adaptation. In both cases, however, the initial basis and any interactions with additional basis elements must still be precomputed from a costly full-space simulation.

Instead of being precomputed, basis functions can be generated analytically. De Witt et al. [DWLF12] and Liu et al. [LMH*15] derive a basis using the eigenfunctions of the Laplacian operator, a construction akin to the scalar Fourier basis. This method applies to any fixed domain shape and produces an arbitrary number of divergence-free basis functions. It leads to an intuitive basis where every basis coefficient only influences a single flow frequency. Cui et al. [CSK18] also show the eigenfunctions can be extended to other boundary conditions, and they propose improvements to make them more computationally efficient. However, these basis functions must be completely recomputed if the simulation domain changes, and their global support precludes any local control over the final simulation.

We propose a multiresolution analytic basis that provides local control and that can be tiled and warped onto arbitrary domains, including dynamic obstacles and curved boundaries.

Vorticity Methods. Several methods use the vorticity formulation of the Navier-Stokes equations to simulate fluid motion. Vorticity representations include point primitives [Ang17], filaments [ANSN06, WP10], and sheets [PTG12]. These elements are advected by the flow and fully represent the fluid motion. Other methods use vorticity to add turbulent details atop coarser simulations [GNS*12, NSCL08]. In either case, fluid dynamics are transposed onto the vorticity elements, where they advect, rotate, and deform into elements of varying scale. Inspired by these approaches, we design basis functions that can adapt to multiscale dynamics.

Since vorticity elements move freely in the simulation domain, fluid flow reconstruction requires neighborhood searches and complex data structures. While we do not rely explicitly on a vorticity formulation, our basis functions are similar to localized vortices, but defined entirely in the spatial domain and statically positioned on regular patterns. This allows us to avoid costly frequency conversions and use simple data structures to accelerate computation.

Wavelet Methods. Our basis functions are inspired by wavelets, which have a rich history in the fluid dynamics literature (e.g., Schneider and Vasilyev [SV10]). Wavelets are used to study the statistical properties of turbulent flows [FR88, BHL93] and enable efficient simulations in vorticity formulation [CP96, FSK99], albeit mostly limited to 2D domains. More general approaches, such as adaptive wavelet collocation methods (AWCM) [KV05, NVBDV15], are also popular in engineering applications since they provide a general and physically-accurate framework applicable to many classes of differential equations. The wavelets used in AWCM, however, are not divergence-free and require frequent conversions to-and-from the frequency domain.

Divergence-free wavelets (DFWs) are also interesting, as they avoid the pressure and incompressibility computations of standard solvers. Lemarié-Rieusset [LR92] proposed DFWs with compact support and Deriaz et al. [DP06] applied them to simulation. These bases are not orthogonal and their construction mostly applies to periodic domains with grid discretizations, which hinders their application to more complex, dynamic domains encountered in graphics. More recent extensions of DFWs onto square domains [Ste11, KHP15] suffer from similar restrictions.

Our work is similar in principle to AWCM and divergence-free wavelets, but our basis functions are better adapted to fluid simulations on complex, potentially dynamic domains. While motivated by classical wavelet theory, we do not rely on it when constructing our vector field basis.

3. Notation and Model Reduction

We introduce our notation and model reduction in 2D, both of which extend naturally to 3D. We use lowercase script for scalars (a), bold lowercase for vectors and vector-valued functions ($\mathbf{a}$), and bold uppercase for matrices and tensors ($\mathbf{A}$). To simplify notation, we sometimes omit function parameters and integral differentials.

We rely on *quasimatrix* notation [Ste98] to express linear combinations of functions, where "matrices" have one dimension of infinite size, i.e., columns are $\in \mathbb{R}^{\infty}$. In our case, quasimatrix columns represent vector-valued functions on the simulation domain $\Omega \subset \mathbb{R}^2$. Let $\odot_i$ denote a tensor-vector product, with the sum taken over the i^{th} collapsed tensor dimension. For instance, given a matrix $\mathbf{A}$ with columns $\mathbf{a}_j$, and a vector $\mathbf{v}$ with elements v_j, the product $\mathbf{A}\mathbf{v}$ can be written as $\mathbf{A} \odot_j \mathbf{v} = \sum_j v_j \mathbf{a}_j$.

We refer to the set of all continuous vector fields defined on Ω as the *full space* $\mathcal{F}$, and refer to its elements as *flows*. The idea of model reduction is to operate on a *reduced* subspace $\mathcal{R} \subset \mathcal{F}$ of flows composed only of linear combinations of a given finite set of linearly-independent *basis flows* $\mathbf{b}_i \in \mathcal{F}, i \in \{1,\dots,r\}$. Let $\mathbf{B} \in \mathbb{R}^{\infty \times r}$ be the quasimatrix whose columns are the r basis flows $\mathbf{b}_i$. Any flow $\mathbf{u} \in \mathcal{R}$ is represented by a set of coefficients $\widetilde{\mathbf{u}} \in \mathbb{R}^r$ as $\mathbf{u} = \mathbf{B}\widetilde{\mathbf{u}}$. Conversely, any flow $\mathbf{u} \in \mathcal{F}$ can be projected to the closest (in the least-squares sense) element of $\mathcal{R}$ using the *normal equation* $\widetilde{\mathbf{u}} = \mathbf{B}^{+}\mathbf{u}$, where $\mathbf{B}^{+}$ is the *pseudoinverse* of $\mathbf{B}$, defined as

$$\mathbf{B}^{+} = \mathbf{B}^{-}\mathbf{B}^{T} \qquad \text{where} \quad \mathbf{B}^{-} = \left(\mathbf{B}^{T}\mathbf{B}\right)^{-1} \tag{1}$$

and where matrix $(\mathbf{B}^T\mathbf{B}) \in \mathbb{R}^{r\times r}$ has entries $(\mathbf{B}^T\mathbf{B})_{ij} = \int_\Omega \mathbf{b}_i \cdot \mathbf{b}_j$.

Our goal is to approximate the behavior of the Navier-Stokes equations on Ω:

$$\partial\mathbf{u}/\partial t = -(\nabla\mathbf{u})\,\mathbf{u} + \nabla p + \nu\nabla^2\mathbf{u} + \mathbf{f} \quad \text{and} \quad \nabla\cdot\mathbf{u} = 0\,, \tag{2}$$

where $\mathbf{u}$, p and $\mathbf{f}$ are functions of space with an implicit dependence on time t. Specifically, $\mathbf{u}$ is the fluid velocity flow, p is the scalar pressure, ν is the viscosity, $\mathbf{f}$ is the external forces flow, ∇ is the gradient operator for scalars or the Jacobian operator for vectors, and ∇^2 is the Laplace operator. We use no-slip boundary conditions, i.e., the flow must match boundary velocities.

Model-reduced simulations commonly impose that all basis flows be divergence-free, making any combined flow $\mathbf{u} \in \mathcal{R}$ divergence-free by linearity. As noted by Treuille et al. [TLP06], this allows us to avoid the costly pressure solve of more traditional solvers [Sta99], and we replace Equations 2 by the simpler system

$$\partial\mathbf{u}/\partial t = -(\nabla\mathbf{u})\,\mathbf{u} + \nu\nabla^2\mathbf{u} + \mathbf{f} \quad . \tag{3}$$

Projecting both sides of Equations 3 onto $\mathcal{R}$ yields

$$\partial\widetilde{\mathbf{u}}/\partial t = -\mathbf{B}^{-}\left((\mathbf{A}\odot_i\widetilde{\mathbf{u}})\odot_j\widetilde{\mathbf{u}}\right) + \nu\mathbf{B}^{-}\left(\mathbf{D}\odot_i\widetilde{\mathbf{u}}\right) + \widetilde{\mathbf{f}}\,, \tag{4}$$

where the *advection tensor* $\mathbf{A} \in \mathbb{R}^{r\times r\times r}$ and the *diffusion matrix* $\mathbf{D} \in \mathbb{R}^{r\times r}$ have elements

$$\mathbf{A}_{ijl} = \int_\Omega \mathbf{b}_l \cdot ((\nabla\mathbf{b}_i)\,\mathbf{b}_j) \quad \text{and} \quad \mathbf{D}_{il} = \int_\Omega \mathbf{b}_l \cdot \nabla^2\mathbf{b}_i\,. \tag{5}$$

Equation 4 thus expresses the fluid dynamics in terms of only the reduced coefficients. This simplifies computational complexity at the cost of restricting the set of possible generated flows.

4. Basis Construction

Equation 4 suggests desirable properties for the basis flows $\mathbf{b}_i$:

1. **divergence-free** basis flows are essential in order to apply the simplified Navier-Stokes formulation in Equation 3,
2. an **orthogonal** basis, i.e., $\int_\Omega \mathbf{b}_i \cdot \mathbf{b}_j = 0 \forall i \neq j$, implies that $\mathbf{B}^{-}$ is the identity, avoiding costly matrix inversions in Equation 4,
3. **local support** sparsifies the $\mathbf{A}$ and $\mathbf{D}$ tensors as elements corresponding to the combination of basis flows with non-overlapping support are zero; this facilitates basis manipulation, since modifying the coefficient of one basis flow only affects the total flow locally in the domain,
4. and basis **completeness**, i.e., the ability to represent *any* flow in $\mathcal{F}$; while this is not generally achievable with a finite number of basis flows, it is desirable to have as large a set of linearly independent basis flows (and at as many *scales*) as possible.

As discussed in Section 2, methods based on full-space simulation snapshots generally only satisfy the divergence-free property, and previous work on Laplacian eigenvectors do not yield locally supported basis functions. Wavelet bases satisfy the last three properties, but it is mathematically impossible to create a basis that satisfies all four properties using traditional wavelet theory [LR94].

We construct a multiscale basis that carefully compromises between these four properties. Each of our basis flows is divergence-free and has bounded support. Our basis can be made increasingly complete, in the sense that we can create as many basis flows with arbitrary frequency as desired, but without necessarily being able to mathematically represent *any* flow in $\mathcal{F}$. We devise a *relaxed* orthogonality property, where only basis flows of the same frequency are orthogonal. We show in Section 5 how this last property can still be exploited to reduce computation cost.

We detail the construction of our basis in the remainder of this section. Our exposition focuses on the 2D case, with particularities of extensions to 3D highlighted in Section 4.6.

4.1. Basis Scheme

Similarly to wavelet approaches, our basis is built atop exemplar basis flows which are tiled over Ω at various scales. We first construct a *basis template* $\mathbf{b}^{\mathbf{k}}(\mathbf{x}) := \mathbf{b}^{(k_x,k_y)}(x,y)$ for each *frequency* $\mathbf{k}$ in a predetermined integer set $\mathcal{K} \subset \left(\mathbb{N}_{\geq 1}\right)^2$. Each basis template $\mathbf{b}^{\mathbf{k}}$ is centered at $(0,0)$ and has a finite rectangular support $\mathcal{S}^{\mathbf{k}} = [-1/(2k_x), 1/(2k_x)] \times [-1/(2k_y), 1/(2k_y)]$, outside of which it has value $(0,0)$. We tile Ω with copies of the basis templates and denote these translated basis flows $\mathbf{b}^{\mathbf{k}}_{\mathbf{c}}(\mathbf{x}) := \mathbf{b}^{(k_x,k_y)}_{(c_x,c_y)}(x,y) := \mathbf{b}^{\mathbf{k}}(x-c_x, y-c_y)$, where subscript $\mathbf{c} \in \mathcal{C}^{\mathbf{k}} \subset \mathbb{R}^2$ indexes the basis flow centers.

The choice of $\mathcal{K}$ and $\mathcal{C}^{\mathbf{k}}$ defines the overall coverage of the simulation domain, with larger sets improving coverage while also increasing computational cost. Throughout this section, we use $\mathcal{K} = \left\{(2^\alpha, 2^\beta) \,|\, \alpha,\beta \in \mathbb{N}_{\geq 0}\right\}$, which corresponds to a power-of-two basis refinement. Note that the refinement in each axis is independent, allowing for anisotropic basis flows. We tile the basis flows on regular lattices using $\mathcal{C}^{\mathbf{k}} = ((\phi/k_x)\mathbb{Z}) \times ((\phi/k_y)\mathbb{Z})$, where ϕ is the *tilling density*. We use $\phi = 1/2$ throughout this section,

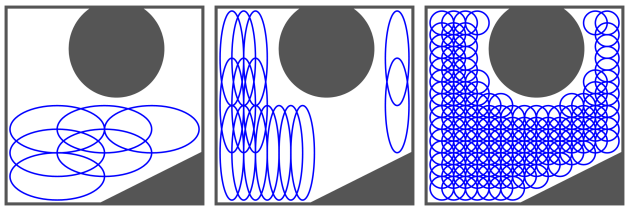

Figure 2: *Visualization of our coverage of a simulation domain, where each ellipsoid represents a single basis flow. For each frequency layer, basis flows are aligned on a regular lattice, and cover as much of the simulation domain as possible. Basis flows with higher anisotropy ratios or smaller scale naturally reach narrower regions, and better wrap around curved boundaries.*

which means basis flows overlap by half of their support size. Figure 2 illustrates the basis coverage obtained with these parameter settings, visualizing only the basis flows with support inside the simulation domain. These $\mathcal{K}$ and $\mathcal{C}^{\mathbf{k}}$ choices adequately cover the simulation domain, but we detail other coverage options in Section 6.

4.2. Divergence-Free, Continuity, and Locality Constraints

Inspired by De Witt et al. [DWLF12], our basis templates are constructed from the divergence-free vector-valued eigenfunctions of the Laplacian on the unit square, with free-slip boundary conditions. Note that even though we use the free-slip eigenfunctions in our construction, our final basis templates will have a no-slip boundary condition. The eigenfunctions are more naturally described on the square $\mathcal{D}^{\mathbf{1}} = [0,1]^2$, so we will define the basis templates on $\mathcal{D}^{\mathbf{k}} = [0,1/k_x] \times [0,1/k_y]$ and shift them back to $\mathcal{S}^{\mathbf{k}}$ in Section 4.5.

Vector eigenfunctions on $\mathcal{D}^{\mathbf{1}}$, which we call the *eigenflows*, are

$$\mathbf{e}^{\mathbf{k}}(\mathbf{x}) = \begin{pmatrix} k_y \sin(k_x \pi x) \cos(k_y \pi y) \\ -k_x \cos(k_x \pi x) \sin(k_y \pi y) \end{pmatrix} \in \mathbb{R}^2, \tag{6}$$

where $\mathbf{k} \in (\mathbb{N}_{\geq 1})^2$ is the eigenflow's *frequency*. We consider a periodic extension of these eigenflows to $\mathbb{R}^2$, with period $2k_x$ in x and $2k_y$ in y (Figure 3). This basis is akin to the Fourier basis, since eigenflows represent a single frequency and have infinite support.

We use linear combinations of eigenflows to construct our basis, aiming for the simplest combinations that satisfy our constraints. Ideally, we would only need to use the single eigenflow $\mathbf{e}^{\mathbf{k}}$ to define a basis template $\mathbf{b}^{\mathbf{k}}$, since it isolates the coarsest frequency matching the size of $\mathcal{D}^{\mathbf{k}}$; however, we cannot simply clamp $\mathbf{e}^{\mathbf{k}}$ to zero outside $\mathcal{D}^{\mathbf{k}}$, as it would create a discontinuous flow.

To define basis template $\mathbf{b}^{\mathbf{k}}$, we therefore need to add some eigenflow *octaves* with higher frequencies to the *fundamental* eigenflow $\mathbf{e}^{\mathbf{k}}$, where the frequencies of the octave eigenflows are integer multiples of $\mathbf{k}$. We denote them $\mathbf{e}^{\mathbf{ak}} = \mathbf{e}^{(a_x k_x, a_y k_y)}$ with $\mathbf{a} = (a_x, a_y) \in (\mathbb{N}_{\geq 1})^2$. In particular, octave $\mathbf{a} = (1,1)$ is the fundamental frequency. We consider a linear combination $\mathbf{h}^{\mathbf{k}}$ of octaves, artificially restricting their support to $\mathcal{D}^{\mathbf{k}}$, which yields

$$\mathbf{h}^{\mathbf{k}}(\mathbf{x}) = \begin{cases} \sum_{\mathbf{a} \in \mathcal{A}} w^{\mathbf{k}}_{\mathbf{a}} e^{\mathbf{ak}}(\mathbf{x}) & \text{if } \mathbf{x} \in \mathcal{D}^{\mathbf{k}} \\ 0 & \text{otherwise} \end{cases}, \tag{7}$$

where $\mathcal{A}$ is a given finite octave multiplier subset of $(\mathbb{N}_{\geq 1})^2$ containing at least $(1,1)$, and $w^{\mathbf{k}}_{\mathbf{a}} \in \mathbb{R}$ are the scalar weights of the linear combination. Since we do not want basis templates to favor either spatial axis, we impose $\mathcal{A}$ to be a Cartesian product of the same set of multipliers $\mathcal{A}_\star$ in x and y, i.e., $\mathcal{A} = \mathcal{A}_\star \times \mathcal{A}_\star$. We aim for $\mathcal{A}$ to be as small as possible, so that the linear combination is dominated by the fundamental eigenflow, and the structure of the basis is as simple as possible. For the same reasons, we impose

$$w^{\mathbf{k}}_{\mathbf{1}} = 1 \tag{8}$$

and aim for $|w^{\mathbf{k}}_{\mathbf{a}}|$ to be as small as possible $\forall \mathbf{a} \neq (1,1)$. To enforce continuity, we constrain $\mathbf{h}^{\mathbf{k}}$ to be zero on the boundary of its support, noted $\partial\mathcal{D}^{\mathbf{k}}$. Evaluating the eigenflows $\mathbf{e}^{\mathbf{ak}}$ on the four sides of $\partial\mathcal{D}^{\mathbf{k}}$ gives

$$\begin{aligned} e^{\mathbf{ak}}(\mathbf{x})\big|_{x=0} &= (0, -a_x k_x \sin(a_y k_y \pi y)) \\ e^{\mathbf{ak}}(\mathbf{x})\big|_{x=1/k_x} &= (0, -(-1)^{a_x} a_x k_x \sin(a_y k_y \pi y)) \\ e^{\mathbf{ak}}(\mathbf{x})\big|_{y=0} &= (a_y k_y \sin(a_x k_x \pi x), 0) \\ e^{\mathbf{ak}}(\mathbf{x})\big|_{y=1/k_y} &= ((-1)^{a_y} a_y k_y \sin(a_x k_x \pi x), 0) \end{aligned}, \tag{9}$$

which results in the constraints

$$\mathbf{h}^{\mathbf{k}}\big|_{\partial\mathcal{D}^{\mathbf{k}}} = 0 \tag{10}$$

$$\Rightarrow \begin{cases} \sum_{a_y \in \mathcal{A}_\star} \left(\sum_{a_x \in \mathcal{A}_\star} - w^{\mathbf{k}}_{\mathbf{a}} a_x k_x \right) \sin(a_y k_y \pi y) & = 0 \\ \sum_{a_y \in \mathcal{A}_\star} \left(\sum_{a_x \in \mathcal{A}_\star} - w^{\mathbf{k}}_{\mathbf{a}} (-1)^{a_x} a_x k_x \right) \sin(a_y k_y \pi y) & = 0 \\ \sum_{a_x \in \mathcal{A}_\star} \left(\sum_{a_y \in \mathcal{A}_\star} w^{\mathbf{k}}_{\mathbf{a}} a_y k_y \right) \sin(a_x k_x \pi x) & = 0 \\ \sum_{a_x \in \mathcal{A}_\star} \left(\sum_{a_y \in \mathcal{A}_\star} w^{\mathbf{k}}_{\mathbf{a}} (-1)^{a_y} a_y k_y \right) \sin(a_x k_x \pi x) & = 0 \end{cases}.$$

Since the functions $\sin(\beta \pi x)$ are linearly independent for different β, we can equate their coefficients in the sums to zero, leading to

$$\Rightarrow \begin{cases} \sum_{a_x \in \mathcal{A}_\star} w^{\mathbf{k}}_{\mathbf{a}} a_x & = 0 \quad \forall a_y \in \mathcal{A}_\star \\ \sum_{a_x \in \mathcal{A}_\star} w^{\mathbf{k}}_{\mathbf{a}} (-1)^{a_x} a_x & = 0 \quad \forall a_y \in \mathcal{A}_\star \\ \sum_{a_y \in \mathcal{A}_\star} w^{\mathbf{k}}_{\mathbf{a}} a_y & = 0 \quad \forall a_x \in \mathcal{A}_\star \\ \sum_{a_y \in \mathcal{A}_\star} w^{\mathbf{k}}_{\mathbf{a}} (-1)^{a_y} a_y & = 0 \quad \forall a_x \in \mathcal{A}_\star \end{cases}, \tag{11}$$

which is a system of linear constraints for variables $w^{\mathbf{k}}_{\mathbf{a}}$. By choosing a sufficiently large $\mathcal{A}$, we can have enough variables to solve this system and obtain weights that zero $\mathbf{h}^{\mathbf{k}}$ on $\partial\mathcal{D}^{\mathbf{k}}$. Figure 3 provides some visual intuition on simple solutions to this system. With weights satisfying Equations 11, $\mathbf{h}^{\mathbf{k}}$ is continuous and locally supported on $\mathcal{D}^{\mathbf{k}}$, as desired.

Interestingly, applying the same method on basis flow derivatives

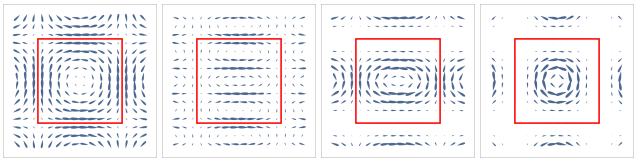

Figure 3: *We combine eigenflows to zero-out flow along the boundary of $\mathcal{D}^{\mathbf{k}}$, shown in red. $\boldsymbol{e}^{(1,1)}$ exhibits the structure we desire for our basis flows (left), but has infinite support and is non-zero on the desired support boundary (and so clamping it would introduce discontinuities). $1/3\boldsymbol{e}^{(3,1)}$ has the same value along on the top and bottom boundaries (middle left), so subtracting it from $\boldsymbol{e}^{(1,1)}$ zeroes the top and bottom velocities (middle right). We repeat this step to zero-out flow along the entire boundary (right). This process is extended with more eigenflows to construct our basis templates.*

to impose $\partial \mathbf{h}^{\mathbf{k}}/\partial x = 0$, $\partial \mathbf{h}^{\mathbf{k}}/\partial y = 0$ or $\partial^2 \mathbf{h}^{\mathbf{k}}/(\partial x \partial y) = 0$ on $\partial \mathcal{D}^{\mathbf{k}}$ results in the exact same constraints. Therefore, by simply imposing continuity on the border, we also obtain continuous first-order derivatives. More importantly, it implies a well-defined divergence (of zero) along the border.

Equations 11 do not imply continuity in the higher-order derivatives. However, these properties could be directly added to the constraint set, if desired. Since $\mathbf{h}^{\mathbf{k}}$ is composed exclusively of sinusoids, its derivatives are easily computed, leading to expressions similar to those in Equations 9. Any degree of smoothness, or more generally any homogeneous linear boundary constraint on the derivatives of the basis flows, could thus be imposed on $\mathbf{h}^{\mathbf{k}}$. Doing so, however, would require a larger $\mathcal{A}$ to satisfy the added constraints, so we choose not to impose any additional smoothness constraints for our application.

4.3. Orthogonality per Frequency

Since $\mathcal{A}$ can be as large as necessary, Equations 11 can be solved with an arbitrarily large number of free coefficients. We exploit these extra coefficients to impose additional orthogonality properties on our basis.

If the location and frequency of every basis flow used in a given domain were known in advance, and did not change during simulation, we could enforce full orthogonality between all basis flow pairs. This would result in a prohibitively large set of quadratic constraints for the coefficients $w_{\mathbf{a}}^{\mathbf{k}}$, mandating in turn the use of a very large $\mathcal{A}$: this is computationally impractical, creates basis flows with complex structures, and requires the computation of a new basis each time the simulation domain changes.

To reduce the number of constraints, we compromise and only impose orthogonality between basis flows of the same frequency. Since two basis flows whose support do not intersect are trivially orthogonal, and our basis flows are tiled regularly with $\phi = 1/2$, we only need to locally impose orthogonality between any given basis flow and its eight neighbors (in 2D) of that same frequency. Due to symmetries, this reduces to only the three constraints:

$$\begin{cases} \int_{\cap_1} \mathbf{h}^{\mathbf{k}}(x,y) \cdot \mathbf{h}^{\mathbf{k}}(x, y - 1/(2k_y)) & = 0 \\ \int_{\cap_2} \mathbf{h}^{\mathbf{k}}(x,y) \cdot \mathbf{h}^{\mathbf{k}}(x - 1/(2k_x), y - 1/(2k_y)) & = 0 \\ \int_{\cap_3} \mathbf{h}^{\mathbf{k}}(x,y) \cdot \mathbf{h}^{\mathbf{k}}(x - 1/(2k_x), y) & = 0 \end{cases}, \tag{12}$$

where $\cap_1$, $\cap_2$ and $\cap_3$ are the support intersections of the two flows in each integrand. This system results in the quadratic constraints:

$$\begin{cases} \sum\limits_{\substack{\mathbf{a}_1 \in \mathcal{A} \\ \mathbf{a}_2 \in \mathcal{A}}} w_{\mathbf{a}_1}^{\mathbf{k}} w_{\mathbf{a}_2}^{\mathbf{k}} \int_{\cap_1} \mathbf{e}^{\mathbf{a}_1 \mathbf{k}}(x,y) \cdot \mathbf{e}^{\mathbf{a}_2 \mathbf{k}}\left(x, y - \frac{1}{2k_y}\right) & = 0 \\ \sum\limits_{\substack{\mathbf{a}_1 \in \mathcal{A} \\ \mathbf{a}_2 \in \mathcal{A}}} w_{\mathbf{a}_1}^{\mathbf{k}} w_{\mathbf{a}_2}^{\mathbf{k}} \int_{\cap_2} \mathbf{e}^{\mathbf{a}_1 \mathbf{k}}(x,y) \cdot \mathbf{e}^{\mathbf{a}_2 \mathbf{k}}\left(x - \frac{1}{2k_x}, y - \frac{1}{2k_y}\right) & = 0 \\ \sum\limits_{\substack{\mathbf{a}_1 \in \mathcal{A} \\ \mathbf{a}_2 \in \mathcal{A}}} w_{\mathbf{a}_1}^{\mathbf{k}} w_{\mathbf{a}_2}^{\mathbf{k}} \int_{\cap_3} \mathbf{e}^{\mathbf{a}_1 \mathbf{k}}(x,y) \cdot \mathbf{e}^{\mathbf{a}_2 \mathbf{k}}\left(x - \frac{1}{2k_x}, y\right) & = 0 \end{cases}. \tag{13}$$

We solve the integrals in Equation 13 analytically for each $\mathbf{k}$ to obtain a system of three quadratic equations for the coefficients $w_{\mathbf{a}}^{\mathbf{k}}$.

4.4. Solving the Constraints

From Sections 4.2 and 4.3, we search for the smallest set $\mathcal{A}$ that satisfies constraints 8, 11, and 13. By successively eliminating families of sets, we arrive at an optimal octave set with $\mathcal{A}_\star = \{1,3,5\}$.

To solve (numerically) for the nine coefficients $w_{\mathbf{a}}^{\mathbf{k}}$ corresponding to this choice of $\mathcal{A}_\star$, we first solve the linear system (Equations 11 and 8), which expresses six of the $w_{\mathbf{a}}^{\mathbf{k}}$ as linear combinations of the other weights. Substituting this into the quadratic Equations 13, we are left with three quadratic equations of three unknowns. This cannot be solved exactly, as it requires solving roots of polynomials of degree eight. We instead obtain a numerical solution using the `NSolve` method from Mathematica [Wol17] with the `"EndomorphismMatrix"` option.

From Bezout's theorem [Coo32], there exist eight solutions to this system, which gives us a way of confirming that the numerical procedure has found all solutions. We discard solutions with complex coefficients and retain the solution that minimizes $\|\mathbf{h}^{\mathbf{k}}\| := \sqrt{\int \mathbf{h}^{\mathbf{k}} \cdot \mathbf{h}^{\mathbf{k}}}$. This leads to a solution that minimizes the influence of higher octaves, since we set $w_{\mathbf{1}}^{\mathbf{k}} = 1$.

4.5. Final Basis Templates

Having defined basis templates on $\mathcal{D}^{\mathbf{k}}$, we translate them back to their more convenient support $\mathcal{S}^{\mathbf{k}}$. We also normalize the basis templates to have unit norm. The final basis template definition is

$$\mathbf{b}^{\mathbf{k}}(x,y) = \left(1/\|\mathbf{h}^{\mathbf{k}}\|\right) \mathbf{h}^{\mathbf{k}}(x - 1/(2k_x), y - 1/(2k_y)) \ . \tag{14}$$

A priori, it seems Equations 8, 11 and 13 must be solved for each fundamental frequency $\mathbf{k}$. However, if k_x and k_y have common factors, they can be factored out of the constraints, and the same solutions are obtained for fundamental frequencies (k_x, k_y) and $(\alpha k_x, \alpha k_y) \forall \alpha \in \mathbb{N}_{\geq 1}$. Therefore, we only need to compute the octave coefficients once per *anisotropy ratio*, denoted $(k_x : k_y)$, and we only compute the templates for frequencies where $\min(\mathbf{k}) := \min(k_x, k_y) = 1$. Also, due to symmetries, coefficients need only be

a	Anisotropy Ratio		
	(1 : 1)	(2 : 1)	(4 : 1)
(1,1)	1.0000000000	1.0000000000	1.0000000000
(1,3)	-0.1107231463	0.0277351959	0.0336558844
(1,5)	-0.1335661122	-0.2166411175	-0.2201935306
(3,1)	-0.1107231463	-0.4866818264	-0.5578126029
(3,3)	0.1262767635	0.0543786840	-0.0036701213
(3,5)	-0.0536214289	0.0647091549	0.1137645934
(5,1)	-0.1335661122	0.0920090959	0.1346875617
(5,3)	-0.0536214289	-0.0381742496	-0.0045291041
(5,5)	0.0588860798	0.0045027306	-0.0242200499
$\|\mathbf{b}^{\widehat{\mathbf{k}}}\|$	0.9783644776	1.3121697019	1.7797075185

Table 1: *2D octave weights $w_a^{\widehat{k}}$ and scaling coefficent $\|\mathbf{b}^{\widehat{\mathbf{k}}}\|$ for anisotropy ratios $(1:1)$, $(2:1)$ and $(4:1)$ solving the constraints of Section 4.*

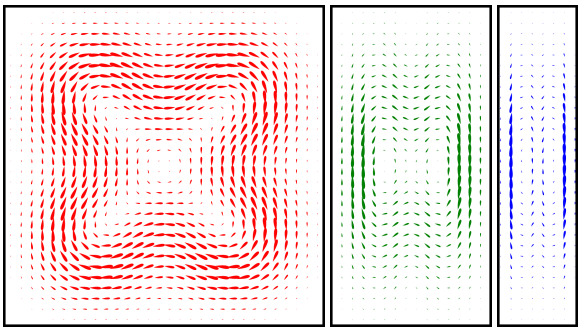

Figure 4: *Basis flow templates for anisotropy ratios $(1:1)$, $(2:1)$ and $(4:1)$.*

computed for anisotropy ratios $(\alpha:\beta)$ with $\alpha \geq \beta$, since other basis templates can be obtained by rotation.

Finer basis templates with $\min(\mathbf{k}) > 1$ can be obtained by scaling. From the eigenflow definition in Equation 6, we have $\mathbf{e}^{\beta\mathbf{k}}(\mathbf{x}) = \beta\mathbf{e}^{\mathbf{k}}(\beta\mathbf{x})$ for any scalar β, which leads to the scaling relation

$$\mathbf{b}^{\mathbf{k}}(\mathbf{x}) = \min(\mathbf{k})\,\mathbf{b}^{\widehat{\mathbf{k}}}(\min(\mathbf{k})\,\mathbf{x}) \ , \tag{15}$$

where $\widehat{\mathbf{k}} = \mathbf{k}/\min(\mathbf{k})$. The few remaining basis templates that need to be explicitly evaluated are stored on a fine grid, and we evaluate these basis templates with tabulation (i.e., GPU texture lookups).

This ability to reuse basis templates is a major advantage of our method. As noted by Jones et al. [JSK16], model reduction methods based on simulation snapshots usually store basis functions at the same resolution as the simulation grid, incurring prohibitive memory costs. Our method only stores one basis template per anisotropy ratio, and all translated basis flows can be represented by only storing their center and frequency.

We provide the octave coefficients and normalization factors for frequency ratios $(1:1)$, $(2:1)$ and $(4:1)$ in Table 1, and illustrate the corresponding basis flows in Figure 4. As desired, these basis flows are divergence-free, have local support, and can be tiled at any scale while remaining orthogonal within each frequency layer.

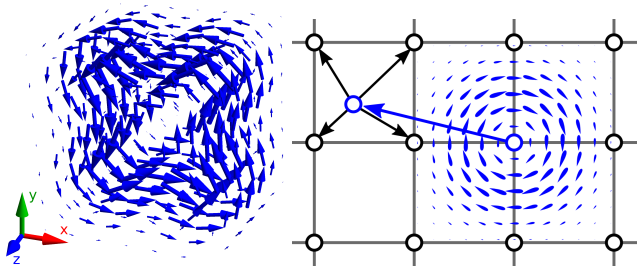

Figure 5: *Left: Z-aligned 3D basis flow of frequency $(1,1,1)$, constructed from the extrusion of the 2D basis flows. Right: Transport method of Section 5.2. A basis flow is advected to a location where no basis flow of the same frequency exists. Since basis flows cannot actually move, its weight is distributed to its four (eight in 3D) neighbors, following the interpolation formulation of Equation 20.*

4.6. Basis Flows in 3D

We extend our construction method to 3D. The main difference is that 3D eigenflows are defined in three separate groups, aligned along each spatial axis. This relates to the fact that vorticity is a scalar value in 2D, but requires three components in 3D. We therefore construct a basis template $\mathbf{b}_z^{\mathbf{k}}$ aligned along the z-axis, and rotate it to create basis templates $\mathbf{b}_x^{\mathbf{k}}$ and $\mathbf{b}_y^{\mathbf{k}}$ aligned along the x- and y-axes. We tile the simulation domain as before, but now each $\mathbf{c} \in \mathcal{C}^{\mathbf{k}}$ is the center of three collocated basis flows (one per axis).

Basis construction is more involved in 3D since not all the eigenflows are linearly independent. Fortunately, we arrive at a very simple solution, which we discuss on an intuitive level, below; we provide full mathematical derivations in a supplemental document.

To construct $\mathbf{b}_z^{\mathbf{k}}$ with frequency $\mathbf{k} = (k_x, k_y, k_z)$, we reuse the 2D basis template definition in (x,y) and extrude it along z. We define

$$\mathbf{b}_z^{\mathbf{k}}(x,y,z) = 2k_z \cos(k_z \pi z)^2 \begin{pmatrix} \left(\mathbf{b}^{(k_x,k_y)}(x,y)\right)\cdot(1,0) \\ \left(\mathbf{b}^{(k_x,k_y)}(x,y)\right)\cdot(0,1) \\ 0 \end{pmatrix} , \tag{16}$$

which is divergence-free by construction. The weighting in z is necessary for the basis flow to be zero along the z boundaries of its support. This particular choice of weighting function is obtained from the general 3D basis template construction method in our supplemental material, and the $2k_z$ factor normalizes the basis template.

We illustrate this basis template in Figure 5-left for frequency $(1,1,1)$. We still only need to compute basis flows where $\min(\mathbf{k}) = 1$, as detailed in Section 4.5. The scaling relation of Equation 15 is however replaced in 3D by

$$\mathbf{b}_z^{\mathbf{k}}(\mathbf{x}) = (\min(\mathbf{k}))^{3/2}\, \mathbf{b}_z^{\widehat{\mathbf{k}}}(\min(\mathbf{k})\,\mathbf{x}) \ . \tag{17}$$

5. Model-Reduced Fluid Dynamics

We now detail an efficient model-reduced fluid solver using our basis. In most model reduction methods, the dynamics are computed directly with Equation 4, which presents many challenges.

First, the advection tensor $\mathbf{A}$ can be very large since it deals with

triplets of basis flows. With our basis, however, the local basis flow supports introduce significant sparsity into $\mathbf{A}$ since any element $\mathbf{A}_{ijl}$ is zero if the supports of any two of the involved basis flows do not intersect. Still, $\mathbf{A}$ contains many non-zero entries, and computing interactions for each triplet of basis flows remains expensive.

Second, there is typically no guarantee that the dynamics equations will project well onto each basis flow, mainly due to Equation 4 modeling the linearized instantaneous behavior of the fluid; the dynamics are defined in a space tangent to the simulation and cannot necessarily be accurately captured with a basis designed to represent the fluid velocity. For instance, the term $\nabla \mathbf{b}_i$ in Equation 5 is not divergence-free, so projecting it onto a divergence-free basis is not likely to be accurate.

For these reasons, we avoid direct use of the advection tensor and diffusion matrix. Instead, we design a simulation method tailored to our basis, with a focus on synthesizing the key behaviors of realistic smoke. We emphasize that our method only uses the Navier-Stokes equations to derive a heuristic but physically plausible simulation behavior, and does not aim to produce accurate solutions to these equations. We specifically focus on the *energy cascade* [Dav15] of a fluid undergoing motion, and identify the following behaviors we wish to capture during simulation:

- energy is introduced into the system through, e.g., buoyancy forces or stirring motions, corresponding to $\widetilde{\mathbf{f}}$ in Equation 4,
- energy can be transported in the simulation domain without changing its frequency content, corresponding to rigid transformations of fluid structures in the flow; this component of transport is usually encoded as a part of the advection tensor $\mathbf{A}$,
- energy can also be transferred *between* energy levels, mostly from large structures and vortices decaying into smaller ones under deformation forces; this is also usually encoded as part of $\mathbf{A}$, and
- at the finest scale, energy is dissipated by viscosity; this is usually obtained by applying the dissipation tensor $\mathbf{D}$.

Our simulator represents velocity fields with our basis and simulates all four behaviors by evolving the reduced flow coefficients in time. Smoke particles are passively advected by the velocity field. We detail each of the simulation stages for the phenomena listed above in the following sections, and summarize the main iteration loop of our method in Algorithm 1.

```
Load precomputed t_ij, r_ij, and (B^T B)_ij;        [Section 5.4]
while true do
    Move dynamic obstacles;
    Project dynamic obstacle boundaries;            [Section 6.2]
    Project external forces;                        [Section 5.1]
    Transport and rotate basis flows;               [Section 5.2]
    Transfer basis flow energy;                     [Section 5.3]
    Stretch basis flows;                            [Section 6.1]
    Advect particles;
    Seed new particles;
```

Algorithm 1: Our simulation method, summarizing Sections 5 and 6.

5.1. Projecting External Forces

As discussed in Section 3, any vector field $\mathbf{f} \in \mathcal{F}$ can be projected onto the basis subspace $\mathcal{R}$ as $\widetilde{\mathbf{f}} = (\mathbf{B}^\mathrm{T}\mathbf{B})^{-1}\mathbf{B}^\mathrm{T}\mathbf{f}$. First, computing $\mathbf{B}^\mathrm{T}\mathbf{f}$ involves computing $\int_\Omega \mathbf{b}_i \cdot \mathbf{f}$ for every basis flow $\mathbf{b}_i$. Both $\mathbf{f}$ and $\mathbf{b}_i$ are defined as continuous functions on Ω, but we use a finite grid to numerically compute the integrals. We found that an axial grid resolution of 32 is sufficient and note that, since basis flows of the same anisotropy have the same definition (up to a scale and offset), we can efficiently compute integrals involving these basis flows concurrently.

Second, we use Gauss-Seidel iterations to solve the linear system to invert $(\mathbf{B}^\mathrm{T}\mathbf{B})$. This approach is often slower than other basic schemes, e.g., Jacobi iterations, since entries cannot generally be processed in parallel; however, Gauss-Seidel iterations are guaranteed to converge in this case since the matrix is symmetric positive definite, while Jacobi iterations offer no convergence guarantee and require smaller time steps [BF11].

Given the local support of our basis flows, $(\mathbf{B}^\mathrm{T}\mathbf{B})$ is sparse, and Gauss-Seidel iterations only need to be applied on non-orthogonal neighboring basis flows. Note that these basis flows are easy to find since our tiling scheme is regular. Furthermore, the orthogonality properties we encode into our basis construction further accelerates computations by reducing the number of non-zero coefficients in $(\mathbf{B}^\mathrm{T}\mathbf{B})$. But, more importantly, our orthogonality property creates a *multicolor scheme* [SS99] where basis flows from the same orthogonality group can be iterated on in parallel. This is a powerful advantage of our basis structure, since it allows us to invert $(\mathbf{B}^\mathrm{T}\mathbf{B})$ with the stability of a Gauss-Seidel scheme and the parallelism of a Jacobi scheme.

Other iterative methods could also be used to invert the system while still taking advantage of this multicolor property, for instance, when computing preconditioners for the conjugate gradient method [BF11]. However, we found that simple Gauss-Seidel iterations are sufficient and yield a stable and parallelizable method to solve the inversion. This is crucial to the performance of any model reduced simulator, as force projection is one of the more computationally expensive steps of these methods. For each of our results (see Section 7), we use no more than 10 Gauss-Seidel iterations to project external forces onto our basis.

5.2. Rigid Transport and Rotation

The term $\mathbf{A}_{ijl}$ in Equation 5 can be interpreted as the influence of basis flow j on basis flow i, with the result being projected onto basis flow l. We use this interpretation to simulate the transport and rotation of basis flows.

Although our basis flows have a fixed position, we can simulate their motion by interpreting them as rigid objects free to move about the simulation domain. The influence of basis flow $\mathbf{b}_j$ on basis flow $\mathbf{b}_i$ is evaluated as

$$\mathbf{t}_{ij} = \left(\int_{\mathcal{S}_i} \mathbf{b}_j(\mathbf{x})\, d\mathbf{x}\right) \Big/ \left(\int_{\mathcal{S}_i} d\mathbf{x}\right). \tag{18}$$

The total instantaneous displacement of $\mathbf{b}_i$ can therefore be evaluated by summing the influence of all neighboring basis flows as

$\sum_j \widetilde{u}_j \mathbf{t}_{ij}$. We obtain a new position for $\mathbf{b}_i$ by scaling this instantaneous displacement by the time step Δt.

However, basis flows are fixed in space, so $\mathbf{b}_i$ cannot directly move to its new location. Furthermore, there is in general no basis flow centered at this new location, so we cannot directly transfer $\widetilde{u}_i$ to another basis flow to simulate the displacement. Instead, we represent the translation of $\mathbf{b}_i$ as an interpolation of neighboring basis flows on the lattice, using an important property of eigenflows: any translated eigenflow can be expressed exactly as a combination of eigenflows on a regular lattice. For a lattice with edge length ϕ, we have (in 2D, for scalars $\alpha, \beta \in [0, \phi]$),

$$\mathbf{e}_{(\alpha,\beta)} = \csc(\pi\phi)^2 \sum_{i,j\in\{0,1\}} \sin(\pi|i\phi-\alpha|)\sin(\pi|j\phi-\beta|)\, e_{(i\phi,j\phi)}. \quad (19)$$

A similar expression also holds in 3D. Since our basis flows are constructed from eigenflows, are arranged on a regular lattice, and the influence of the fundamental octave is made as prominent as possible, this translation property of eigenflows also approximately holds for our basis flows. Therefore, to simulate the translation of $\mathbf{b}_i$, we project it onto the four basis flows nearest to its new location on the lattice (eight, in 3D; see Figure 5-right).

Instead of the exact weights in Equation 19, we use linear weights:

$$\mathbf{b}_{(\alpha,\beta)} \approx \sum_{i,j\in\{0,1\}} |i-\alpha/\phi|\,|j-\beta/\phi|\, b_{(i\phi,j\phi)}. \quad (20)$$

These weights are simpler to evaluate and are a good approximation to the trigonometric weights of Equation 19. Another benefit of this weighting is that, while the trigonometric weights preserve the L^2 norm of the decomposition, linear weights preserve the L^1 norm: since we consider linear combinations of translated basis flows, the linear weights lead to an energy preserving method, while the trigonometric weights do not. For instance, if a single basis flow is transported in the domain, distributed to the four closest basis flows on the lattice (in 2D), and these four basis flows are then all transported back to the starting location, trigonometric weights will increase the magnitude of the resulting basis flow coefficient, while linear weights will recover the initial coefficient exactly.

We apply a similar strategy to rotate basis flows in 3D: assuming basis flows rotate about their center, we can compute the total rotation of $\mathbf{b}_i$ caused by the influence of all other basis flows as

$$\sum_j \widetilde{u}_j \mathbf{r}_{ij} \text{ with } \mathbf{r}_{ij} = \left(\int_{\mathcal{S}_i} \frac{(\mathbf{x}-\mathbf{c}_i)\times\mathbf{b}_j(\mathbf{x})}{\|\mathbf{x}-\mathbf{c}_i\|^2} d\mathbf{x}\right) \Big/ \left(\int_{\mathcal{S}_i} d\mathbf{x}\right). \quad (21)$$

We express the resulting rotation as a vector (axis & amplitude) scaled by the time step. Since each basis flow is collocated in a triplet of basis flows aligned along x, y and z, we have an orthogonal frame in which we can express the rotated axis. The coefficient of the rotated basis flow is thus transferred to the three collocated axis-aligned basis flows, and we preserve energy by normalizing the L^1 norm of this coefficient transfer. Note that we ignore rotations in 2D since basis flows have a roughly circular vorticity profiles, and therefore do not significantly change under rotation.

5.3. Energy Transfer and Diffusion

So far, we have only treated energy transfers within a given frequency layer. Next, we show how to also transfer energy across frequencies, which is fundamental for simulating complex, turbulent behaviors.

As illustrated for 2D in Figure 6, we arrange frequency layers in a regular graph. To simulate forward energy cascades, each basis flow receives energy from its coarser neighboring basis flows and transfers energy to its finer neighboring basis flows. Not all basis flows are able to process this regular energy transfer: first, basis flows on the coarsest layer cannot receive energy from any coarser layers. They instead receive energy from forces in the system (Section 5.1). Second, basis flows at the highest frequency have no other basis flows to which they can transfer their energy. Here, we simply remove this energy from the system, replicating the dissipation of energy at the finest scales of turbulent flows. This imitates the effect of the explicit application of the diffusion matrix $\mathbf{D}$ in Equation 4.

For all other "middle" layers, we control the rate at which energy is transferred: let $\|\mathbf{k}\| = \sqrt{k_x^2 + k_y^2}$ be the scalar *wavenumber* associated with a given frequency layer, let $\zeta(\mathbf{k})$ be the total energy in layer $\mathbf{k}$, and let $\tau(\mathbf{k})$ be the portion of the energy the layer transfers away at each time step. From Kolmogorov theory [Dav15], we know that the physically-correct distribution of energy in fully-developed turbulent flows should be proportional to $\|\mathbf{k}\|^{-5/3}$. We design a transfer mechanism based on this steady-state distribution.

Of the energy $\zeta(\mathbf{k}_0)\tau(\mathbf{k}_0)$ transferred by frequency $\mathbf{k}_0$, let $q(\mathbf{k}_0 \to \mathbf{k})$ be the proportion given to frequency $\mathbf{k}$. Note that, since our transfer mechanism is regular across all frequency layers, we have $\sum_{\mathbf{k}_0\in\mathcal{K}} q(\mathbf{k}_0 \to \mathbf{k}) = 1$. We choose to distribute energy equally across all other layers of frequency at most double that of layer $\mathbf{k}_0$ (in any direction), as depicted in Figure 6. Each layer, thus, transfers energy to three other layers in 2D, or seven layers in 3D.

Suppose $\tau(\mathbf{k}) = \lambda\|\mathbf{k}\|^\varepsilon$ for some scalars λ and ε. At steady state, the energy transferred and received by a frequency layer should be equal, therefore

$$\zeta(\mathbf{k})\tau(\mathbf{k}) = \sum_{\mathbf{k}_0\in\mathcal{K}} \zeta(\mathbf{k}_0)\tau(\mathbf{k}_0)\, q(\mathbf{k}_0 \to \mathbf{k}) \quad (22)$$

$$\Leftrightarrow \|\mathbf{k}\|^{-5/3}\lambda\|\mathbf{k}\|^\varepsilon = \sum_{\mathbf{k}_0\in\mathcal{K}} \|\mathbf{k}_0\|^{-5/3}\lambda\|\mathbf{k}_0\|^\varepsilon\, q(\mathbf{k}_0 \to \mathbf{k}) \quad (23)$$

and choosing $\varepsilon = 5/3$ yields the desired steady state distribution. The λ parameter cancels out and does not affect the steady state, but it controls the rate at which the energy distribution approaches the steady state. Note that lambda must be small enough so that $\tau(\mathbf{k}) < 1$ for all frequencies, but the energy transfer can be performed in multiple passes if faster transfer rates are desired.

In practice, we transfer energy between basis flows, and not between entire frequency layers. We must therefore define the energy of a single basis flow, which is not trivial since not all basis flows are orthogonal. Furthermore, the natural choice of using $(\widetilde{u}_i)^2$ as the energy of basis flow $\mathbf{b}_i$ would require non-linear transfers between basis coefficients, and would not distinguish between clockwise and counter-clockwise swirls. We instead directly use $\widetilde{u}_i$ as a signed replacement for the energy of basis flow $\mathbf{b}_i$. Given this, we

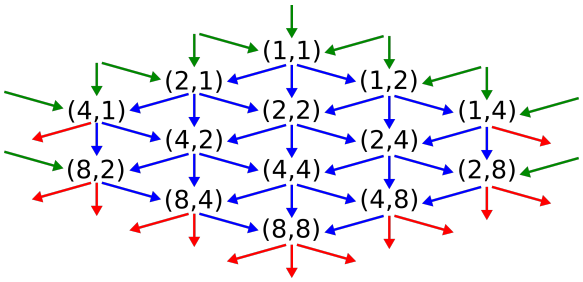

Figure 6: *Our energy distribution graph (Section 5.3). Energy enters the system at the coarsest frequencies (green arrows). At each time step, each basis flow distributes part of its energy to the neighbors indicated with blue arrows. At the finest scale, energy cannot be transferred to other basis flows, so we remove it from the system to simulate dissipation (red arrows).*

withdraw energy from basis flow $\mathbf{b}_i$ by simply reducing its coefficient value by $\widetilde{u}_i \tau(\mathbf{k}_0)$.

This is a coarse approximation of the physically-correct behavior, but it leads to satisfactory energy cascades in practice. Note that, since energy measures are now linear instead of quadratic, we adjust the expected energy distribution exponent to $\varepsilon = 5/6$. The steady-state of Equation 23 does not exactly hold with this signed energy, but $\varepsilon = 5/6$ still gives a good default value about which other energy distributions can be explored.

We distribute each energy "packet" $(\widetilde{u}_i\, \tau(\mathbf{k}_0)\, q(\mathbf{k}_0 \to \mathbf{k}))$ to all neighboring basis flows of frequency $\mathbf{k}$; let $\mathcal{B}^{\mathbf{k}}$ be the indices of all basis flows with frequency $\mathbf{k}$. We define the proportion given to each basis flow as

$$v(\mathbf{b}_i \to \mathbf{b}_j) = (B^T B)_{ij} / \textstyle\sum_{l \in \mathcal{B}^{\mathbf{k}}} |(B^T B)_{il}| \,. \tag{24}$$

With this weighting, a basis will transfer most of its energy to neighboring basis flows with similar structures (i.e., with inner product close to 1), leading to natural deformations. Note that the numerator above must be signed, otherwise a clockwise swirl could decay unnaturally into a counter-clockwise swirl.

Energy is finally transferred from basis flow $\mathbf{b}_i$ of frequency $\mathbf{k}_0$ to basis flow $\mathbf{b}_j$ of frequency $\mathbf{k}$ by adding

$$\widetilde{u}_i\, \tau(\mathbf{k}_0)\, q(\mathbf{k}_0 \to \mathbf{k})\, v(\mathbf{b}_i \to \mathbf{b}_j) \tag{25}$$

to the coefficient of basis $\mathbf{b}_j$.

Figure 7 shows the effect of varying energy transfer parameters λ and ε in our energy transfer method. In general, smaller lambdas cause the simulation to be dominated by advection (Figure 7 left). Larger lambdas, instead, dissipate energy more quickly, causing the simulation to be dominated by buoyancy forces (Figure 7 right). Using a moderate λ (Figure 7 middle-left and middle-right) leads to more interesting energy transfers. Using $\varepsilon = 5/6$ as previously suggested then leads to a natural smoke plume, while using $\varepsilon = -11/6$ leads to a more erratic, exaggerated behavior.

Even though our method uses heuristics, it creates visually acceptable smoke dynamics. Figure 8 and our supplementary video compare our method with a full-scale simulation [Sta99] for a simple smoke plume, to show that we can recreate the overall motion and features of the smoke in this case.

5.4. Reusing Dynamics Coefficients

The entire simulation dynamics are reduced to computing the *interaction coefficients* $\mathbf{t}_{ij}$, $\mathbf{r}_{ij}$, and $(\mathbf{B}^{\mathrm{T}}\mathbf{B})_{ij}$. Note that, contrary to the original advection tensor $\mathbf{A}$ that operates on triplets of basis flows, our interaction coefficients only involve pairs of basis flows, which greatly reduce the number of coefficients to compute and store.

Furthermore, we can reuse most of the interaction coefficients since the interaction of two basis flows only depends on their *relative* position, and our basis flows are distributed regularly throughout the domain. Similarly, interactions only depend on the relative scale of the basis flows involved, and so they can be computed once for coarse frequencies and reused for higher frequencies.

We apply a lazy evaluation scheme, re-scaling the two basis flows involved in an interaction so that their largest dimension has unit length, before translating them so the first basis flows is centered at $(0,0)$. We maintain a dictionary of all interaction coefficients, indexed by the relative sizes and positions of the resized basis flows. If the interaction coefficient for the resized basis flows is not present in the dictionary, it is computed (as detailed earlier) and added to the dictionary.

Each interaction coefficient scales to higher frequencies as a simple function of the scaling ratio. These relations are derived directly from Equations 15 and 17, and summarized in Table 2. The interaction coefficient for the resized basis flows is thus re-scaled back to the original basis flow size, and used to evolve the fluid in time.

Since basis flow interactions are local and do not depend on the simulation domain, we can save the dictionary and reuse it for any other domain. Furthermore, we can reduce the cost of dictionary lookups by locally caching the coefficients required by each basis flow. This reduces computation time by a factor of $10-50\times$ depending on the scene, at the cost of a $1.5-4\times$ increase in memory.

Figure 7: *Smoke plumes with different energy transfer parameters.*

Figure 8: *Smoke plume generated with a full-space simulation [Sta99] (left) and our method (right). Since our method uses model reduction, it does not perfectly recreate the behavior of the full-space method. However, our heuristic dynamics do recreate some visually important features of the smoke plume, such as the smaller vortices shedding from the sides of the plume.*

Coefficient	Scaling Factor Exponent (α)	
	2D	3D
$\mathbf{B}^T\mathbf{B}$	0	0
$\mathbf{t}$	1	$3/2$
$\mathbf{r}$	—	$5/2$

Table 2: *Factors for scaling interaction coefficients in Section 5.4. We obtain coefficients for basis flows with* $\min(\boldsymbol{k}) > 1$ *by multiplying the corresponding precomputed coefficient by* $(\min(\boldsymbol{k}))^{\alpha}$*, where* α *is given above.*

6. Improved Coverage and Obstacle Coupling

In the basis construction of Section 4, we spaced every basis flow on the same frequency layer by half their support size ($\phi = 1/2$). However, this leads to gaps in coverage, since each layer contains points where the flow is always zero (see Figure 9, left). One solution is to increase the number of basis flows in each layer (Figure 9, right). For instance, doubling the number of basis flows and separating them by a quarter of their support size ($\phi = 1/4$) improves coverage, but at the cost of additional computation. Note that each frequency layer would now have four orthogonal groups, but each group can still be processed in parallel during force projections (see Section 5.1).

Another solution is to maintain $\phi = 1/2$ spacing, but to offset basis flow layers independently with respect to each other, e.g. with an offset of $(o(k_x), o(k_y))$, where

$$o(k) = \sum_{i=0}^{(\log_2 k)-1} \frac{\phi}{4}\left(\frac{1}{2}\right)^i = \frac{\phi\,(1-1/k)}{2}. \tag{26}$$

Figure 9 (middle) illustrates how using this approach improves coverage without increasing the number of basis flows (or computation cost). Note also that this offsetting scheme maintains the regular structure of our basis across frequency layers, and all the basis

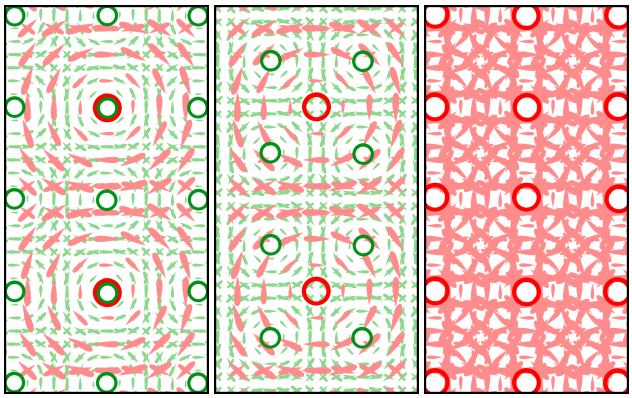

Figure 9: *With* $\phi = 1/2$ *and no offset, the coverage across two layers (*$\boldsymbol{k} = (1,1)$ *in red and* $\boldsymbol{k} = (2,2)$ *in green) has points of zero flow (left, at the intersection of red and green centers). By offsetting basis flows in each layer, finer layers can cover the gaps left by coarser layers (middle). Alternatively, using* $\phi = 1/4$ *creates a much denser coverage (only* $\boldsymbol{k} = (1,1)$ *is shown; right). The basis flows are represented as simplified circular vector fields to help visualization.*

reusability advantages of Section 5.4 remain. Unfortunately, this offsetting strategy is incompatible with anisotropic basis flows in 3D: since we apply the offset independently for each axis, basis flows with different alignments get a different offset, and we lose the property required in Section 5.2 for basis flows to be grouped as collocated triplets.

For all results in this paper, we use $\phi = 1/4$ as it provides significantly better coverage than half-support separation. The improvement in coverage is even more significant in 3D because of the three collocated basis flows at each point. For that reason, we did not find necessary to use anisotropic basis flows in 3D with the $\phi = 1/4$ modification. We thus also use the offset strategy for all our results, which further improves coverage at no additional cost.

User-driven Coverage. We can additionally apply a spatially varying basis placement tailored, e.g., to scene complexity or artist-driven simulation constraints. For example:

- if the simulation domain has both large and narrow regions, it may be sufficient to place finer-scale basis flows exclusively in the narrow regions and near boundaries, or
- if we desire view-dependent simulation accuracy and refinement, we can place higher-frequency basis flows closer to the camera (i.e., where a viewer is most likely to notice finer-scale details), or
- we can place basis flows only where smoke particles are present.

These three strategies are illustrated in Figure 10 and two are used in the results of Figures 13 and 15.

Although they further approximate the simulation dynamics, such basis flow placement strategies can provide a significant advantage in computation speed. Note that some strategies may require adding and/or removing basis flows during simulation; since we precompute all local interactions once (Section 5.4), doing so does not incur any significant overhead.

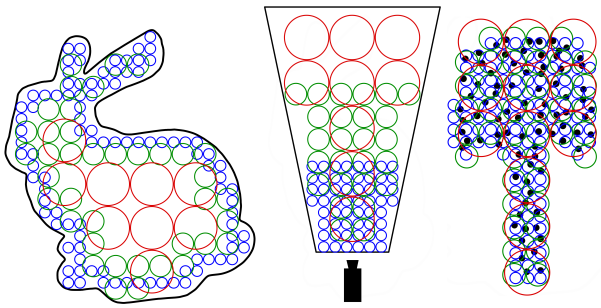

Figure 10: *Spatially-varying coverage strategies. Left to right: placing finer-scale basis flows closer to boundaries; view-dependent coverage generates richer dynamics closer to the viewer; basis flow placement biased to regions with particles, since velocities further from particles contribute less to their behavior.*

6.1. Curved Boundaries

We adapt our simulation method to new domain shapes by tilling the domain with our basis flows. However, given their rectangular support and the regular lattice positioning, our basis does not naturally adapt well to angled or curved boundaries. Figure 11 (bottom left, dotted line) illustrates "staircasing" artifacts due to poor coverage near boundaries for coarser simulations.

We propose a basis deformation scheme to solve this problem: instead of only tiling basis flows in the simulation domain Ω, we allow basis flows to overlap boundaries, as long as their center is within the simulation domain and the distance from their center to the nearest boundary exceeds $1/4$ their support width. We represent domain boundaries and obstacles with signed distance functions (SDF), and we displace and warp basis flows that overlap boundaries as follows: for each corner of the basis flow support located inside an obstacle, we uniformly squash the basis flow along the direction of the SDF gradient, leaving the opposite corner fixed, until the initial corner is out of the obstacle. We then uniformly stretch the basis flow in the direction perpendicular to the SDF gradient to recover the original area (volume in 3D) of the basis flow.

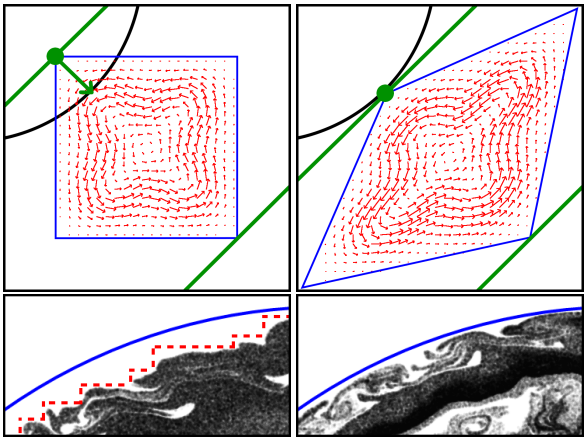

Figure 11: *Top: we displace basis flows (blue) that overlap a boundary (black) along the SDF gradient (green arrow) and deform them (right) to maintain their area. Bottom: without basis deformation, the extent of the basis flows (dotted line) does not properly cover the domain, and coarse simulations can exhibit staircase artifacts. Our deformation eliminates these issues (right).*

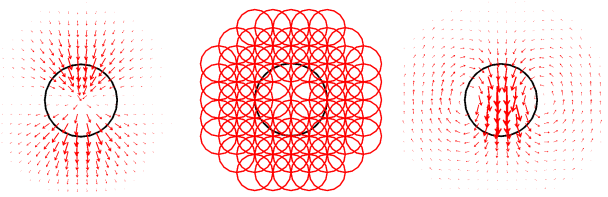

Figure 12: *Obstacle coupling. Left to right: we evaluate the obstacle's distance-weighted normal velocity near its boundary; the velocity is projected onto basis flows near the boundary, allowing basis flows to intersect the object; the resulting divergence-free flow approximates the normal velocity of the object.*

The deformed basis flow, even after area preservation, may no longer be divergence-free if more than one corner has been moved. The stretching nevertheless helps reduce staircase artifacts and results in a more visually pleasing vector field (i.e., one where particles do not seem to be "compressed" in free-space for no apparent reason). Note that, in moving a corner and stretching the basis flow, we sometimes force another corner out of the simulation bounds. In practice, we iterate once over all corners while preserving the area, and then make a second pass to move each corner into the domain without trying to preserve the area, to ensure all corners are within the simulation domain.

Given the deformed basis flow support, we map the original flow $\mathbf{b}$ to a point $\mathbf{x}$ of the new support as $\mathbf{M}\left(\mathbf{b}\left(\mathbf{M}^{-1}\left(\mathbf{x}\right)\right)\right)$, where $\mathbf{M}$ is the bilinear transform from the original axis-aligned support to the deformed support. We compute $\mathbf{M}^{-1}$ with Newton iterations, using no more than five iterations in all our tests.

In principle, new interaction coefficients should be computed for each of the deformed basis flows. This would however be computationally expensive, and we instead simply use the coefficients of the original undeformed basis flows. This further approximates the fluid behavior, but still yields satisfactory results in practice.

6.2. Obstacle Coupling

In order for moving obstacles to influence our simulation, we project the normal velocity at the boundary of dynamic obstacles onto our basis. Figure 12 illustrates this process for a circular obstacle moving downward.

We evaluate the normal flow near obstacle boundaries using the difference of the SDF of a dynamic obstacle at two consecutive time steps, $\mathrm{SDF}^{(t-1)}$ and $\mathrm{SDF}^{(t)}$, as

$$\left(\left(\mathrm{SDF}^{(t-1)} - \mathrm{SDF}^{(t)}\right)/\Delta t\right) \nabla \mathrm{SDF}^{(t)} \max\left(1 - |\mathrm{SDF}^{(t)}|/\sigma, 0\right) \tag{27}$$

where $\nabla \mathrm{SDF}^{(t)}$ is the unit gradient of the current SDF, Δt is the time step, and $\sigma > 0$. The last term reduces the strength of the normal flow as we move away from the boundary, which makes the

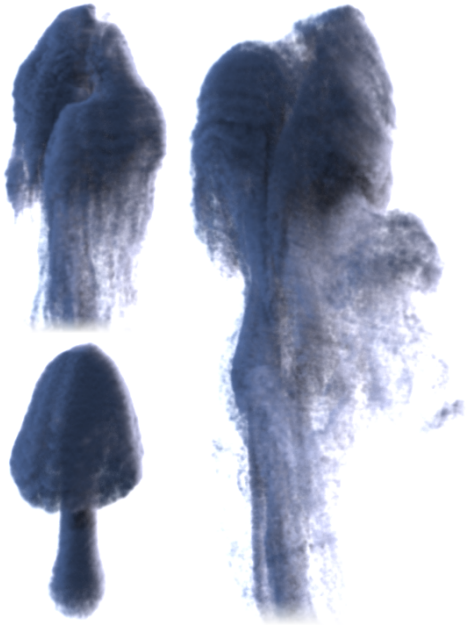

Figure 13: *3D smoke plume. Buoyancy and advection initially lead to the characteristic mushroom shape, after which energy transfers introduce turbulent behavior. We use only two frequency levels in this simple simulation, with a coverage that activates basis flows only around smoke particles, as in the rightmost part of Figure 10.*

normal flow continuous in Ω, and σ effectively controls the distance at which moving obstacles influence the simulation.

We then project this normal flow onto only the basis flows that fall within a neighborhood band of width σ around the boundary, including basis flows that intersect the interior of the obstacle. Here, we do not apply basis deformation (Section 6.1) during projection. The projected normal flow is divergence-free, which makes its behavior more natural around the object. For instance, in Figure 12, the projected normal flow pushes and drags smoke particles in the direction of obstacle motion, as expected, but it also "rolls" particles around the sides of the obstacle, which is a desirable behavior not present in the normal flow itself prior to its projection.

This additional projected boundary flow is then added to the velocity field when advecting smoke particles. We also add the computed boundary flow coefficients to the original basis flow coefficients when computing fluid interactions (as per Section 5), so that obstacle movement can generate secondary motions.

7. Results and Discussion

We apply our method to various smoke simulation scenarios and refer readers to the supplemental video for full animation sequences. Figure 13 illustrates a simple 3D smoke plume. Energy enters the system from external buoyancy forces, which we model as small vertical vectors projected onto our basis at each particle location. Figure 1 demonstrates interactions with a static triangle mesh obstacle. As explained in Section 6.1, our basis flows are deformed slightly to better adjust to the obstacle shape.

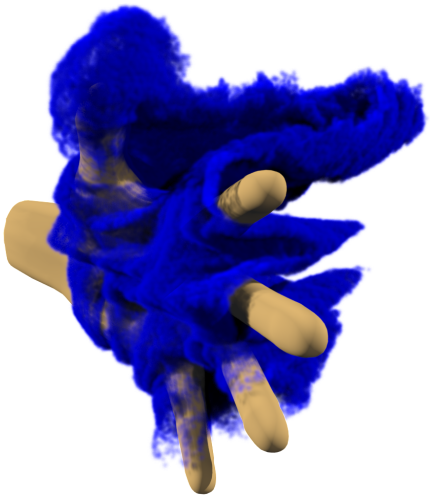

Figure 14: *Hand pushing through a smoke cloud.*

Figure 14 shows a hand moving through a smoke cloud. No buoyancy is used in this scene, and energy is only created from the hand's movement. We again use a triangle mesh to represent the obstacle, which is converted to a signed distance field in order to compute dynamic obstacle interactions, as explained in Section 6.2. Our obstacle coupling method is approximate, and particles can still collide with moving objects (e.g., see the 2D moving obstacles in our accompanying video). This is generally unavoidable since our basis flows are not tailored to specific obstacle geometries, but the moving hand in Figure 14 is illustrative of the effectiveness of our method with complex obstacles.

Figure 16 shows how increasing the number of frequency layers impacts the resolution of the simulation and the ability of our method to cover narrow simulation domains. This test uses the basis stretching method of Section 6.1, which further improves domain coverage.

Finally, Figure 15 shows a smoke simulation inside of a glass

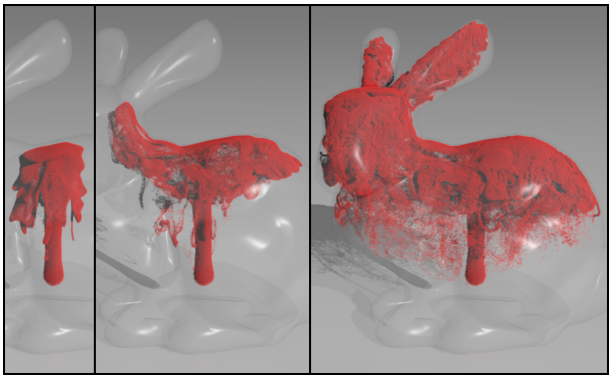

Figure 15: *Smoke plume inside a glass bunny. Three frequency levels are used, which is made possible by using the coverage strategy of Figure 10-left. The finest basis flows are only used where necessary (e.g., in the ears), reducing the number of basis flows by* $10\times$.

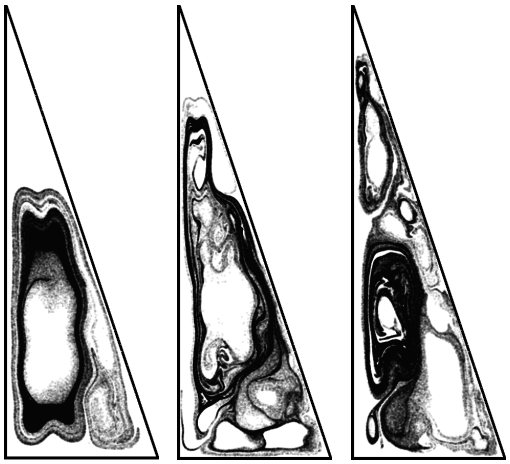

Figure 16: *Comparison of simulations using different numbers of frequency layers. Higher frequency layers have smaller basis flows, which improves simulation details and domain coverage.*

bunny, again illustrating our method's ability to adapt to complex simulation domains without any need for customized basis precomputation. We apply the tiling strategy in Figure 10-left for this example, where finer basis flows are only placed in narrow regions of the domain (e.g. the bunny's ears) and near boundaries. Tiling the entire domain with basis flows at all scales would require 412K basis flows, whereas our adaptive placement uses only 41K (i.e., 10× fewer basis flows) while still allowing the particles to reach and fill the ears of the bunny.

7.1. Computation Times

We present a breakdown of computational costs for all our 3D scenes in Table 3. All results are computed on an Intel i7 quad core running at 3.4 GHz with 32GB of RAM.

The actual dynamics computations (Basis Advection and Energy Transfer columns) are never the bottleneck of our method, which showcases the benefits of only having to compute interactions between a few basis flows in local neighborhoods. The cost of basis flow deformation (Section 6.1) depends greatly on the scene; all scenes (except Figure 13, which has no obstacles) use mesh obstacles, where SDF computation is costly but could be made more efficient with proper acceleration structures. This is most problematic in the "Glass Bunny" scene (Figure 15) where, given the low number of basis flows, the relative cost of basis stretching (which depends heavily on the number of particles) increases.

As is often observed with model-reduced methods, the two most costly operations are the computation of external forces and particle advection, since these operations require conversions between the reduced- and full-spaces. Our particle advection implementation is not performance-optimized: we only parallelize inner loops on the CPU using OpenMP. An end-to-end GPU implementation is likely to improve performance measurably, whereas we currently only leverage the GPU to accelerate external force projection. Note that particle and basis data are both well-suited for representation in textures, and velocity fields can be directly rasterized into output buffers. We leave such accelerations to future work.

Our method is usually memory bound, especially in the scene of Figure 1. Recall that our interaction coefficient cache (Section 5.4) amplifies the memory costs, trading memory for performance, but we still find the improvements in compute cost significant enough to justify the additional required memory.

Overall, the computational cost of our simulations is higher than those reported by previous model-reduced methods [DWLF12, KD13, LMH*15]. However, we operate in a different regime; previous methods typically only use a few hundred globally-supported basis functions, while we use several *thousand* localized basis flows. This inevitably increases the cost of total flow reconstruction and particle advection, for instance, since the contribution of all basis flows must be accumulated. Nevertheless, we believe the coverage flexibility and the ability to adapt to dynamic domains justify the higher cost of our method.

8. Conclusion and Future Work

We presented an end-to-end model reduction method for adaptive multiscale smoke simulations. We detailed a novel vector field basis construction that yields divergence-free basis flows with localized support and beneficial orthogonality properties. We showed how basis flow interactions can be precomputed and reused across different simulation domains and with dynamic obstacles.

Even though our method does not aim to compute a physically-accurate solution of the Navier-Stokes equations, it succeeds in generating visually plausible simulations. We therefore believe that the flexibility our method offers with its dynamic basis placement strategies, interaction coefficients reusability over any simulation domains, and simple energy transfer controls is pertinent for computer graphics applications.

In the future, we would like to investigate new coverage schemes during basis construction: for instance, even if our power-of-two refinement scheme is a natural choice, we may be able to obtain better coverage from, e.g., a power-of-$\sqrt{2}$ refinement. We would also like to investigate the fluid authoring capabilities offered by the local nature of our basis flows, and are interested in extending our basis construction to other tiling patterns such as hexagonal grids. We are also interested in extending our technique to other simulation problems, such as the simulation of free-surface behavior, where our ability to adapt to changing simulation domains could lead to efficient model-reduced liquid simulations.

References

[Ang17] ANGELIDIS A.: Multi-scale vorticle fluids. *ACM TOG 36*, 4 (2017), 104. 2

[ANSN06] ANGELIDIS A., NEYRET F., SINGH K., NOWROUZEZAHRAI D.: A controllable, fast and stable basis for vortex based smoke simulation. In *SCA '06: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation* (Aire-la-Ville, Switzerland, Switzerland, Jan. 2006), Eurographics Association, pp. 25–32. 2

[BF11] BURDEN R. L., FAIRES J. D.: Numerical analysis, 2011. 7

Scene	# particles	# flows	Computation Time (seconds)						Memory (GB)
			Buoyancy	Deform.	Boundary	Basis Adv.	Energy	Particle Adv.	
Bunny Teaser	738K	277K	38.67	41.68	—	1.37	1.64	168.78	24.8
Hand	151K	565K	—	62.24	27.69	0.90	1.61	274.95	14.5
Glass Bunny	807K	41K	4.74	221.96	—	0.09	0.06	320.18	1.2
3D Plume	300K	99K	12.79	—	—	0.48	0.56	28.48	8.7

Table 3: *Scene statistics for 3D results (Section 7). Each value represents the average over all simulation frames.*

[BHL93] BERKOOZ G., HOLMES P., LUMLEY J. L.: The proper orthogonal decomposition in the analysis of turbulent flows. *Annual review of fluid mechanics 25*, 1 (1993), 539–575. 2

[BJ05] BARBIČ J., JAMES D. L.: Real-time subspace integration for st. venant-kirchhoff deformable models. In *ACM TOG* (2005), vol. 24, ACM, pp. 982–990. 2

[Coo32] COOLIDGE J. L.: A treatise on algebraic plane curves. *Bull. Amer. Math. Soc. 38 (1932), 163-165* (1932), 0002–9904. 5

[CP96] CHARTON P., PERRIER V.: A pseudo-wavelet scheme for the two-dimensional navier-stokes equation. *Computational and Applied Math. 15* (1996), 139–160. 2

[CSK18] CUI Q., SEN P., KIM T.: Scalable laplacian eigenfluids. *ACM TOG 37*, 4 (2018), 1–12. 2

[Dav15] DAVIDSON P.: *Turbulence: an introduction for scientists and engineers*. Oxford University Press, USA, 2015. 7, 8

[DP06] DERIAZ E., PERRIER V.: Divergence-free and curl-free wavelets in two dimensions and three dimensions: application to turbulent flows. *Journal of Turbulence*, 7 (2006), N3. 2

[DWLF12] DE WITT T., LESSIG C., FIUME E.: Fluid simulation using laplacian eigenfunctions. *ACM TOG 31*, 1 (2012), 10. 2, 4, 13

[FM96] FOSTER N., METAXAS D.: Realistic animation of liquids. *Graphical models and image processing 58*, 5 (1996), 471–483. 2

[FR88] FARGE M., RABREAU G.: Transformée en ondelettes pour détecter et analyser les structures cohérentes dans les écoulements turbulents bidimensionnels. *CR Acad. Sci. Paris Série II b 307* (1988), 433–440. 2

[FSK99] FARGE M., SCHNEIDER K., KEVLAHAN N.: Non-gaussianity and coherent vortex simulation for two-dimensional turbulence using an adaptive orthogonal wavelet basis. *Physics of Fluids (1994-present) 11*, 8 (1999), 2187–2201. 2

[GKSB15] GERSZEWSKI D., KAVAN L., SLOAN P.-P., BARGTEIL A. W.: Basis enrichment and solid–fluid coupling for model-reduced fluid simulation. *Computer Animation and Virtual Worlds 26*, 2 (2015), 109–117. 2

[GNS*12] GOLAS A., NARAIN R., SEWALL J., KRAJCEVSKI P., DUBEY P., LIN M.: Large-scale fluid simulation using velocity-vorticity domain decomposition. *ACM TOG 31*, 6 (2012), 148. 2

[HTC*14] HAHN F., THOMASZEWSKI B., COROS S., SUMNER R. W., COLE F., MEYER M., DEROSE T., GROSS M.: Subspace clothing simulation using adaptive bases. *ACM TOG 33*, 4 (2014), 105. 2

[JSK16] JONES A. D., SEN P., KIM T.: Compressing fluid subspaces. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2016), Eurographics Association, pp. 77–84. 6

[KD13] KIM T., DELANEY J.: Subspace fluid re-simulation. *ACM TOG 32*, 4 (2013), 62. 2, 13

[KHP15] KADRI HAROUNA S., PERRIER V.: Divergence-free wavelet projection method for incompressible viscous flow on the square. *Multiscale Modeling & Simulation 13*, 1 (2015), 399–422. 2

[KRFC09] KRY P. G., REVÉRET L., FAURE F., CANI M.-P.: Modal locomotion: Animating virtual characters with natural vibrations. In *Computer Graphics Forum* (2009), vol. 28, Wiley Online Library, pp. 289–298. 2

[KV05] KEVLAHAN N. K., VASILYEV O. V.: An adaptive wavelet collocation method for fluid-structure interaction at high reynolds numbers. *SIAM Journal on Scientific Computing 26*, 6 (2005), 1894–1915. 2

[LMH*15] LIU B., MASON G., HODGSON J., TONG Y., DESBRUN M.: Model-reduced variational fluid simulation. *ACM TOG 34*, 6 (2015), 244. 2, 13

[LR92] LEMARIE-RIEUSSET P. G.: Analyses multi-résolutions non orthogonales, commutation entre projecteurs et dérivation et ondelettes vecteurs à divergence nulle. *Revista Matemática Iberoamericana 8*, 2 (1992), 221–238. 2

[LR94] LEMARIÉ-RIEUSSET P.-G.: Un théorème d'inexistence pour les ondelettes vecteurs à divergence nulle. *Comptes rendus de l'Académie des sciences. Série 1, Mathématique 319*, 8 (1994), 811–813. 3

[NSCL08] NARAIN R., SEWALL J., CARLSON M., LIN M. C.: Fast animation of turbulence using energy transport and procedural synthesis. In *ACM TOG* (2008), vol. 27, p. 166. 2

[NVBDV15] NEJADMALAYERI A., VEZOLAINEN A., BROWN-DYMKOSKI E., VASILYEV O. V.: Parallel adaptive wavelet collocation method for pdes. *Journal of Computational Physics 298* (2015), 237–253. 2

[PTG12] PFAFF T., THUEREY N., GROSS M.: Lagrangian vortex sheets for animating fluids. *ACM TOG 31*, 4 (2012), 112. 2

[PW89] PENTLAND A., WILLIAMS J.: *Good vibrations: Modal dynamics for graphics and animation*, vol. 23. ACM, 1989. 2

[SS99] SAAD Y., SOSONKINA M.: Enhanced parallel multicolor preconditioning techniques for linear systems. In *PPSC* (1999). 7

[SSW*13] STANTON M., SHENG Y., WICKE M., PERAZZI F., YUEN A., NARASIMHAN S., TREUILLE A.: Non-polynomial galerkin projection on deforming meshes. *ACM TOG 32*, 4 (2013), 86. 2

[Sta99] STAM J.: Stable fluids. In *Proceedings of 26th annual conf. on Computer graphics and interactive techniques* (1999), ACM Press/Addison-Wesley Publishing Co., pp. 121–128. 2, 3, 9, 10

[Ste98] STEWART G. W.: *Afternotes goes to graduate school: lectures on advanced numerical analysis*, vol. 58. Siam, 1998. 3

[Ste11] STEVENSON R.: Divergence-free wavelet bases on the hypercube: Free-slip boundary conditions, and applications for solving the instationary stokes equations. *Mathematics of Computation 80*, 275 (2011), 1499–1523. 2

[SV10] SCHNEIDER K., VASILYEV O. V.: Wavelet methods in computational fluid dynamics*. *Annual Review of Fluid Mechanics 42* (2010), 473–503. 2

[TLP06] TREUILLE A., LEWIS A., POPOVIĆ Z.: Model reduction for real-time fluids. In *ACM TOG* (2006), vol. 25, ACM, pp. 826–834. 2, 3

[Wol17] WOLFRAM RESEARCH: Mathematica, Version 10.3, 2017. Champaign, IL. 5

[WP10] WEISSMANN S., PINKALL U.: Filament-based smoke with vortex shedding and variational reconnection. In *ACM TOG* (2010), vol. 29, ACM, p. 115. 2

[WST09] WICKE M., STANTON M., TREUILLE A.: Modular bases for fluid dynamics. In *ACM TOG* (2009), vol. 28, ACM, p. 39. 2