Basis Enrichment and Solid-fluid Coupling for Model-reduced Fluid Simulation

Dan Gerszewski
University of Utah

Ladislav Kavan
University of Pennsylvania

Peter-Pike Sloan
Activision

Adam W. Bargteil
University of Utah

Abstract

We present several enhancements to model-reduced fluid simulation that allow improved simulation bases and two-way solid-fluid coupling. Specifically, we present a basis enrichment scheme that allows us to combine data driven or artistically derived bases with more general analytic bases derived from Laplacian Eigenfunctions. We handle two-way solid-fluid coupling in a time-splitting fashion—we alternately timestep the fluid and rigid body simulators, while taking into account the effects of the fluid on the rigid bodies and vice versa. We employ the vortex panel method to handle solid-fluid coupling and use dynamic pressure to compute the effect of the fluid on rigid bodies.

Keywords: Fluid simulation, model reduction, solid-fluid coupling

1 Introduction

One of the most significant drawbacks of physics-based animation is "the curse of dimensionality"—the quest for ever-higher fidelity leads to an explosion in the number of degrees of freedom. This problem naturally leads to the consideration of dimensionality reduction techniques. By constructing a problem-specific model of our fluid, we can reduce the number of degrees of freedom to only the ones needed for our specific problem. While this is less accurate than a full dynamics simulation, we can tailor the simulation to our specific time and computational requirements.

In this paper, we present several enhancements to the basic reduced fluid simulation pipeline. Specifically, we present a basis enrichment scheme for combining analytic, data-driven, and artistically authored bases as well as a new approach to two-way solid-fluid coupling that scales to a large number of rigid bodies.

The analytic bases act somewhat like regularization allowing our approach to generalize outside the training data and thus requiring significantly less training data without the risk of over-fitting. We treat two-way solid-fluid coupling in a time-splitting fashion—we first compute the effect of the solid on the fluid and then compute the effect of the fluid on the solid. We employ a *vortex panel method* to compute obstacles' effects on the fluid and dynamic pressure to compute forces induced on the obstacle by the surrounding fluid. In a precomputation step, we account for the geometric boundary of each object, which involves assembling and inverting a dense "panel matrix;" however, at runtime solid-fluid coupling reduces to a matrix multiplication for each object. We handle multiple obstacles by iteratively computing the coupling in a way similar to Schwarz alternating methods.[19] Fluid-solid coupling is achieved using dynamic pressure to compute forces on solid objects from fluid velocities; these forces are then treated as external forces in a rigid body simulator. Our results demonstrate that our enhancements are practical for two-way coupled reduced fluid simulation with rigid bodies.

2 Related Work

Dimensionality reduction for computational fluid dynamics (CFD) has been well studied in engineering. These methods are variously referred to in the literature as proper orthogonal decomposition (POD), Karhunen-Loéve decomposition, or subspace integration. The POD method has been used in many applications involving dimensionality reduction of complex flows and to investigate coherent structures in turbulent flows.[12,13,8] The snapshot POD method introduced by Sirovich[17] for the study of coherent structures can be used to create a reduced model from a series of snapshots of a simulation. Treuille and colleagues[20] introduced this snapshot technique to graphics and described how each step of a fluid simulation can be performed in the reduced space.

Since that work researchers have also developed modular techniques by connecting fluid *tiles*, which capture specific boundary conditions, at run time to create large novel reduced fluid simulations.[21] Additionally, researchers have also experimented with different bases for fluid simulation. Gupta and colleagues[6] used the Legendre polynomials for both simulation and rendering of participating media. Long and colleagues[10] improve upon Fourier-based solutions by shifting to the discrete sine/cosine transform to handle boundary conditions. However, this method is limited to simple domains. More recently, DeWitt and colleagues[4] used static analysis of the domain to construct a basis from Eigenfuctions of the Laplacian. In some simple domains, this basis even has a closed form. In our work, we combine this basis with the snapshot POD method. Other researchers have extended reduced fluid methods by applying a cubature approach for non-linear functions[9] and including inverse operators for solid-fluid coupling.[18]

Treuille and colleagues[20] handled solid obstacles by defining a local basis on a fixed size grid surrounding each obstacle. This local basis was created by computing the

velocity field that cancels the flow into the obstacle induced by each mode of the fluid simulation basis and then applying the same snapshot POD technique to compute a compressed basis. This process was repeated for a number of translations and rotations of the obstacle. Additionally, the rigid body motion of each object was also sampled to incorporate object movement into the local basis. At runtime, the local basis for canceling the normal flow is determined based on the location, rotation, and movement of the object. In contrast to this approach, our approach to solid-fluid coupling, based on vortex panel methods, does not limit interaction to a small region around the obstacle, has a small runtime memory footprint, avoids expensive precomputation, and allows for direct interaction between obstacles.

Other approaches for handling moving boundary conditions in reduced fluid simulations involve taking the difference of the normal velocity and the desired normal velocity, and projecting it onto the velocity basis and then subtracting the result from the reduced state.[4] This method approximates the forces up to the resolution representable by the basis modes. In contrast, our method is able to increase the resolution of our boundary conditions independent of our fluid basis.

Our approach to solid-fluid coupling makes use of the vortex panel method, which was developed to study flow around airfoils[7,3] and was introduced to graphics by Park and Kim[14] to handle obstacles in a vortex particle method. More recent variations have been used to simulate smoke as a surface.[15,2]

3 Methods

In this section, we will first briefly review the mechanics of reduced fluid simulation, then introduce our basis enrichment scheme, and finally present our approach for two-way solid-fluid coupling.

3.1 Reduced Fluid Simulation

The basic mechanics for reduced fluid simulation were introduced by Treuille and colleagues.[20] We begin with the incompressible Navier-Stokes equations which describe the motion of a viscous fluid,

$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u}\cdot\nabla)\mathbf{u} - \nu\nabla^2\mathbf{u} + \nabla p + \mathbf{f}_e \tag{1}$$

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

where $\mathbf{u}$ is the velocity, ν is the viscosity parameter, p is the pressure, and $\mathbf{f}_e$ are the external forces. The goal of reduced simulation is to reduce the dimensionality of $\mathbf{u}$ through Galerkin projection onto a low-dimensional basis,

$$\tilde{\mathbf{r}} = \mathbf{B}^T\mathbf{u} \tag{3}$$

where, $\tilde{\mathbf{r}} \in \mathbb{R}^r$ represents the reduced coefficients and $\mathbf{B}$ is the basis represented as a matrix with r columns, each representing a basis function.

A typical fluid simulation in computer graphics employs *operator splitting* breaking the simulation into several individual steps: advection, applying external forces, applying viscosity, and projection onto a divergence-free field. To perform reduced fluid simulations, we must address each of these steps.

Fortunately, because we only include divergence free fields in our basis, we can only represent divergence free fields removing the need for the expensive projection step. External forces are easily handled by Galerkin projection onto the basis. Specifically, given external forces, $\mathbf{f}_e$, we compute reduced forces

$$\tilde{\mathbf{f}}_e = \mathbf{B}^T\mathbf{f}_e. \tag{4}$$

These are simply scaled and added to the reduced velocity coefficients,

$$\tilde{\mathbf{r}} := \tilde{\mathbf{r}} + s\tilde{\mathbf{f}}_e, \tag{5}$$

for some scaling factor s that accounts for density, grid-spacing, and timestep.

The diffusion term is also easily handled. Being a linear operator, the discretization of the diffusion operator $\nabla^2\mathbf{u}$ can be represented as a matrix $\mathbf{D}$. Projecting into the subspace we get the reduced diffusion matrix

$$\tilde{\mathbf{D}} = \mathbf{B}^T\mathbf{D}\mathbf{B}, \tag{6}$$

which is precomputed for a given domain.

The non-linear advection operator, $-(\mathbf{u}\cdot\nabla)\mathbf{u}$, is more complicated. The non-linearities preclude it from being written as a single reduced matrix. Instead, a reduced advection matrix for each basis function can be precomputed and then at runtime combined into the final reduced advection operator. The discretization of the advection operator for a given velocity field, $\mathbf{u}$, can be expressed as a matrix, $\mathbf{A}_\mathbf{u}$. This matrix, when applied to a field, $\mathbf{v}$, (i.e $\mathbf{A}_\mathbf{u}\mathbf{v}$) has the effect of advecting $\mathbf{v}$ through $\mathbf{u}$.

Thus, we precompute, for each basis function or *mode*, $\mathbf{b}_i$, in the basis $\mathbf{B} = [\mathbf{b}_1 \ldots \mathbf{b}_r]$ a matrix, $\mathbf{A}_{\mathbf{b}_i}$, that represents advection through the velocity field $\mathbf{b}_i$. Each of these matrices can be reduced

$$\tilde{\mathbf{A}}_{\mathbf{b}_i} = \mathbf{B}^T\mathbf{A}_{\mathbf{b}_i}\mathbf{B}, \tag{7}$$

during precomputation. During simulation, the reduced advection matrix is computed by summing all mode advection matrices weighted by their corresponding reduced state coefficient

$$\tilde{\mathbf{A}} = \sum_i \tilde{\mathbf{A}}_{\mathbf{b}_i} r_i. \tag{8}$$

Viscosity and advection can be combined into a single update from time t to $t+\Delta t$ and can be written as:

$$\tilde{\mathbf{r}}^{t+\Delta t} = \left(e^{\Delta t(\nu\tilde{\mathbf{D}}+\tilde{\mathbf{A}})}\right)\tilde{\mathbf{r}}^t. \tag{9}$$

This matrix-vector product is computed efficiently using an iterative Taylor approximation.[21]

We note that while the reduced simulation can proceed without the notion of a *grid*, for collecting training data and visualization purposes a grid is useful. In our system, we explicitly use the grid for solid-fluid coupling.

3.2 Basis Enrichment

The divergence free bases used in reduced fluid simulations have been constructed in either of two ways. The first method involves running a training simulation and then extracting a reduced basis using a Singular Value Decomposition (SVD). This process is accomplished by concatenating velocity-field snapshots of a high-resolution fluid simulation into a matrix, computing the SVD, and then selecting r singular vectors.[20] A basis generated in this way can capture motion similar to the training data very well in the least squares sense, however, it suffers from a number of problems. Arbitrary motion during runtime can be problematic as the basis may not generalize well to motion outside of the training simulation, e.g. using a training simulation where an obstacle generates flow in one half of the domain for a runtime where the obstacle moves to the other half. To minimize problems from over fitting, a significant amount of simulation data has to be precomputed. Additionally, it can be difficult for artists to know what kinds of training simulations to run in order to generate a suitable basis, not to mention the large amount of precomputation space and time needed.

The second method involves creating a basis using an analytic approach, for example choosing Eigenfunctions of the Laplacian operator. For a few simple domains, these bases can be computed in closed form. In more general domains, the Eigenfunctions of the discrete Laplacian operator are computed using an Eigendecomposition.[4] In simple domains like a box, the advection operators can be computed analytically and because the modes are only loosely coupled, the resulting matrices are sparse. The Eigenfunction modes work well for gross flow and do not suffer from over-fitting, but detailed flow can require an impractically large number of modes.

To give artists control over generating a basis, we provide a velocity drawing tool. After the velocity has been drawn, it is projected onto a divergence-free field and the artist can timestep the simulation to generate the desired velocity field. This process allows an artist to create different flow effects, such as vortices or laminar flow paths, with minimal training data. Alternatively, artists can simply interact with the simulation to generate training data. We will now describe how to combine different bases, a similar approach has been used in the context of reduced bases for direct to indirect radiance transfer.[11]

To exploit any sparsity that might exist in the Laplacian Eigenfunctions, we would like to keep this basis intact when including the data driven, artist generated modes. Thus, given a Laplacian Eigenfunction basis, $\mathbf{E}$, and velocity fields generated by an artist, $\mathbf{D} = [\mathbf{d}_1 \dots \mathbf{d}_N]$, where each column is a user generated velocity field scaled to unit length, we would like to construct a combined basis that keeps the structure of $\mathbf{E}$ intact. First, the SVD of $\mathbf{D} = \mathbf{USV}^T$ is computed and the left singular vectors, $\mathbf{U}$, with corresponding singular values greater than zero are retained. $\mathbf{U}$ is then deflated against the basis,

$$\mathbf{U}_d = \mathbf{U} - \mathbf{E}\mathbf{E}^T\mathbf{U}, \tag{10}$$

where the columns of $\mathbf{U}_d$ now contain the parts of the velocity fields, $\mathbf{U}$, that could not be represented by the basis, $\mathbf{E}$. The columns of matrix $\mathbf{U}_d$ are now orthogonal to the columns of $\mathbf{E}$ but may no longer be orthogonal to each other, i.e., $\mathbf{U}_d^T\mathbf{U}_d$ may not be the identity. To generate a basis that spans the same subspace we simply compute the SVD of $\mathbf{U}_d$ and retain the singular vectors corresponding to non-zero singular values[1], resulting in an orthonormal basis $\mathbf{R}$. Concatenation of $\mathbf{E}$ and $\mathbf{R}$ forms an orthogonal basis, perfectly valid for reduced fluid simulation. From now on we therefore assume that $\mathbf{B}$ is the concatenated matrix $[\mathbf{E}|\mathbf{R}]$.

We would also like the ability to specifically activate the artist generated modes during runtime. If one wishes to directly excite an artist created mode during run time, the projection of those modes into $\mathbf{B}$ can be precomputed. At run time the resulting coefficients can be added to the reduced state. No projection is necessary during run time.

3.3 Two-way Solid-fluid Coupling

We use the reduced fluid simulation engine described in Section 3.1 and Box2D[1] for rigid body simulation. To couple them we use a time splitting technique and alternately timestep each simulator while taking into account the effects of the fluid on the rigid bodies and vice versa.

3.3.1 Solid to Fluid Coupling

To account for the effect of rigid bodies on the fluid flow, we adopt a vortex panel method.[3,14] This approach has two advantages over previous work. First, obstacles are not limited to a finite range of spatial influence. In fact, they have global influence, though the fall-off is quite fast. Second, we avoid the substantial precomputation of sampling the object's effect at various positions and orientations in the domain. Our only precomputation involves inverting matrices. Finally, we note that our approach generalizes beyond reduced fluid simulation and could be used in other contexts, such as smoothed particle hydrodynamics, Eulerian, or semi-Lagrangian methods.

In two dimensions, objects are discretized into M piecewise linear segments called panels. In our system, the panel lengths are chosen to be on the order of the fluid simulation's grid spacing. The panels are then used both as quadrature points and as vorticity sources that cancel flow normal to the obstacle.

The velocity, $\mathbf{u} = (u, v)$, generated by a panel at a point $\mathbf{x}$ in the local coordinate system of the panel is given by

$$u = \frac{\gamma\beta}{2\pi}, \quad v = \frac{\gamma}{2\pi} ln \frac{d_o + \epsilon}{d_e + \epsilon}, \tag{11}$$

where γ is the *panel strength*, β is the angle subtended by the panel from the point $\mathbf{x}$, d_o, and d_e are the distances from $\mathbf{x}$ to the origin and end of the panel respectively, and ϵ is a small constant to avoid division by zero (see Figure 1).

[1] While $\mathbf{U}$ is full rank, if there is a large overlap between $\mathbf{U}$ and $\mathbf{E}$, deflation will result in a rank deficient matrix $\mathbf{U}_d$ (with zero singular values).

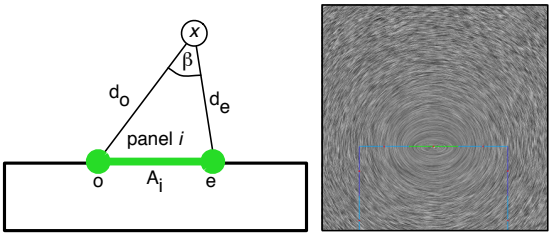

Figure 1: Left: Panel coordinate system. Right: Velocity field induced by the panel.

To cancel the flow normal to an object we must consider the interactions between all the panels of the object. To do so we compute a coupling matrix $\mathbf{P} \in \mathbb{R}^{M \times M}$ that encodes the influence of the strength of panel i on the velocity at panel j. Specifically, let $\bar{\mathbf{u}}_{ij}$ be the velocity induced at the mid-point of panel j by panel i when panel i has unit strength (i.e. $\gamma_i = 1$). Then the P_{ji} is given by

$$P_{ji} = -\bar{\mathbf{u}}_{ij} \cdot \mathbf{n}_j, \tag{12}$$

where $\mathbf{n}_j$ is the normal vector of the j-th panel.

Given $\mathbf{P}$ and a velocity field, $\mathbf{u}$, to cancel the flow normal to the obstacle we must solve the linear system,

$$\mathbf{P}\gamma = \mathbf{b} \tag{13}$$

where γ is the panel strength vector, and $\mathbf{b}$ is a vector encoding the violation of the boundary condition. Specifically,

$$b_i = A_i \left(\mathbf{u}_f - \mathbf{u}_o\right) \cdot \mathbf{n}_i \tag{14}$$

where b_i is the violation at panel i, A_i is the panel area, $\mathbf{u}_f$ is the fluid velocity evaluated at the midpoint of the panel, and $\mathbf{u}_o$ is the velocity of the object. This approach corresponds to a 1-point quadrature rule. Of course, higher order methods could be used.

As described, the $M \times M$ panel coupling matrix $\mathbf{P}$ is singular and an additional constraint must be added in order to obtain a unique solution. We add the constraint that there is zero circulation around the boundary, i.e.

$$\sum_i^M A_i \gamma_i = 0. \tag{15}$$

This constraint is encoded by adding a row to the panel matrix containing the panel lengths and a zero to the end of $\mathbf{b}$. The panel matrix is computed in object space, allowing for rigid body transformations without modification. $\mathbf{P}$ can be inverted during precomputation; at runtime panel strengths are computed with a single matrix-vector product.

Some distributions of panels are problematic when objects contain symmetries. For example, a square with two panels per side is unable to cancel the normal velocities induced from rigid body rotation. In such cases it suffices to use an odd number of panels per side.

Multiple Bodies Thus far we have described how to handle a single object. To handle multiple objects we must account for their interaction. Ideally, we would compute a single coupling matrix encoding the interactions of all panels in the system. However, this would require solving a new and much larger linear system every step, removing the ability to precompute an inverse.[2] Instead, we employ a fixed point iteration approach that takes advantage of the precomputed inverse panel matrices. First, the panel strengths of each object are computed to satisfy the boundary conditions of the reduced velocity field, i.e. for all objects i we compute

$$\gamma_i = \mathbf{P}_i^{-1}\mathbf{b}_i. \tag{16}$$

We then iteratively solve for panel strengths that additionally satisfy object-object interactions.

Each iteration, for each object i in our simulation:

1. Compute $\mathbf{b}_i^{obj}$, which is the boundary violation induced by all other objects.
2. Store the previously computed panel strengths.
3. Solve for the new panel strengths,
$$\gamma_i = \mathbf{P}_i^{-1}(\mathbf{b}_i^{orig} + \mathbf{b}_i^{obj}). \tag{17}$$
4. Compute the norm of the difference in panel strengths.

Iterations are performed until the panel strengths converge, or a user specified tolerance or iteration limit is reached. This scheme, which falls into the class of Schwarz alternating methods,[19] is guaranteed to converge to a unique solution for second order PDE's. Golas et al.[5] successfully demonstrate an alternating method to couple Eulerian grids with vortex particle methods.

This alternating scheme may fail due to the singularities that occur when evaluating the velocity very near a panel. Velocities evaluated too close to a panel should not be relied upon and instead another approach should be taken, such as interpolating from reliable positions.[7]

Domain Boundaries When an object approaches the domain boundary, the velocity field induced by its vortex panels will not generally respect the solid wall boundary conditions. For simple domains with closed form Laplacian Eigenfunctions, we employ the *method of images*—used in electrostatics to handle wall boundary conditions—to accurately and efficiently enforce the wall boundary conditions. To do so, objects that violate the solid wall boundary conditions above an error threshold are reflected across the solid wall. The resulting combined velocity field will only have tangential components along the solid wall. The velocities induced from the reflected panels are evaluated only at positions that fall inside the domain. This approach will correctly satisfy the domain wall boundary conditions by canceling the normal components of the velocity induced by the original object. This

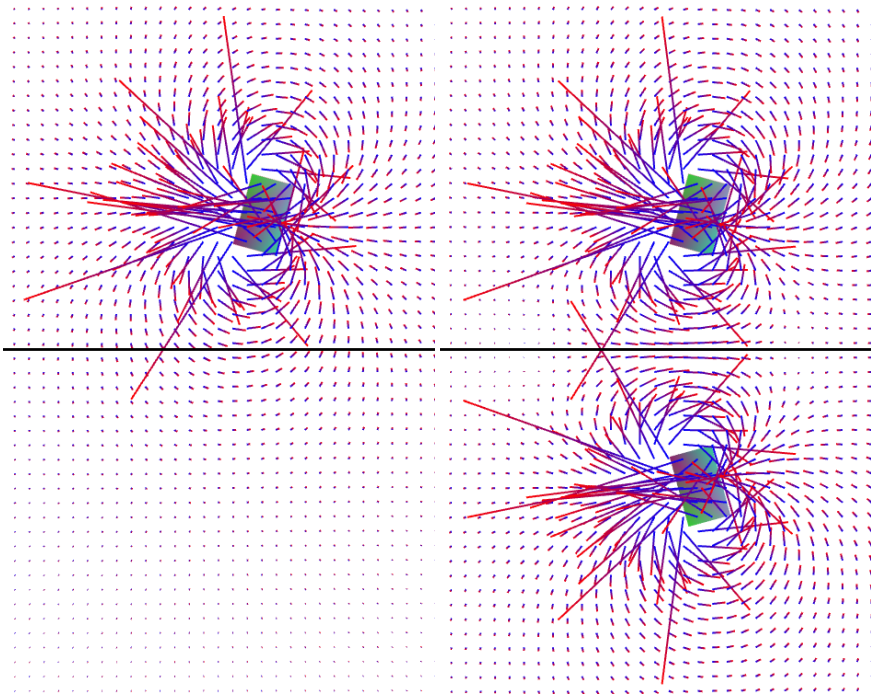

Figure 2: Left: A visualization of the velocity field of an object near a domain boundary. Note that, along the black line, the velocities point into and out of the domain. Right: After the addition of a mirrored object below the black line, there is no flow across the domain boundary.

method is similar in spirit to Long and colleagues[10] who used the reflection properties of the discrete sine/cosine transform to handle solid wall domain boundaries.

Feedback The resulting velocity field is a combination of the reduced fluid velocity, $\mathbf{u}_r$, and a panel velocity field, $\mathbf{u}_p$, where

$$\mathbf{u}_p = \sum_i \mathbf{u}_i \tag{18}$$

and $\mathbf{u}_i$ is the velocity field induced by panel i. $\mathbf{u}_p$ can be evaluated at any specific point in space through evaluation of Equation (11) and a straightforward summation. For example, to advect a tracer particle we can combine the reduced velocity, reconstructed in the neighborhood of the particle, with the velocity evaluated from the panels.

However, the panel strengths have no memory and must be recomputed from scratch each timestep. Thus, we need to feed their contribution back into the reduced fluid simulation to preserve momentum. This step can be accomplished by iterating over the panels and summing their contribution to the background grid. The resulting velocity field, $\mathbf{u}_p$, is then be projected into the reduced space and added to the reduced coefficients. However, naïvely evaluating Equation (11) at every background grid velocity sample is computationally expensive and can be especially wasteful if there are large errors when $\mathbf{u}_p$ is projected into the reduced basis.

Instead, we approximate the contributions of panels to distant background grid samples using a quadtree data structure. Specifically, we build a quadtree over the background grid where the root corresponds to the entire domain and the leaf nodes correspond to disjoint sub-grids. In our examples the maximum size of a leaf node sub-grid is 4×4, corresponding to 4 u and 4 v velocity samples. We use a precomputed error metric to determine how deep to descend the quadtree when evaluating $\mathbf{u}_i$. To precompute this error metric, we consider a unit strength vortex panel and evaluate $\mathbf{u}_i$ at the center of the quadtree node, $\mathbf{c}$, and additional sample points inside the quadtree node, $\mathbf{s}_j$. Then the maximum error induced by using a constant approximation of $\mathbf{u}_i$ for the quadtree node is

$$\max_j \|\mathbf{u}_i(\mathbf{c}) - \mathbf{u}_j(\mathbf{s}_j)\|. \tag{19}$$

We compute these error samples for quadtree nodes at a number of distances and directions from the panel and store the maximum error incurred at a given level of the quadtree for a given distance.

At runtime, when computing the contribution of $\mathbf{u}_i$ to $\mathbf{u}_p$, which is stored on the background grid, we use these precomputed values to determine the error induced by approximating the velocities using the center of a quadtree node. If the error is below a threshold, the panel veloc-

ity is evaluated at the center of the quadtree node and this value is added to all the background velocity values covered by the quadtree node, otherwise we descend the tree.

When using our quadtree acceleration, we still must project $\mathbf{u}_p$ onto the reduced basis. Note that some details of the velocity field will be lost in this projection and, in particular, the reduced velocity field may not respect obstacles boundaries. However, before this feedback the velocity field $\mathbf{u}_r + \mathbf{u}_p$ does satisfy the boundary conditions and can be evaluated exactly at any point in space in time linear in the number of panels and the number of reduced coefficients. This velocity should be used for, e.g., advecting tracer particles.

3.3.2 Fluid to Solid Coupling

We incorporate fluid to solid coupling by computing the dynamic pressure on the boundary of the rigid body. From the dynamic pressure we compute the force, which is then added to the rigid body simulation. The dynamic pressure, sometimes called the velocity pressure, is

$$q = \frac{1}{2}\rho \mathbf{u}^T \mathbf{u}, \tag{20}$$

where ρ is the density of the fluid, and $\mathbf{u}$ is the fluid velocity. For each panel we have already computed the difference in relative velocity between the obstacle and fluid when solving for the panel strengths. From that velocity, we compute the dynamic pressure q at panel centers and then multiply by the panel area to get forces,[16] which are normal to the panels. Specifically, the force on panel i is

$$\mathbf{f} = A_i q \mathbf{n}_i, \tag{21}$$

which is then applied to the rigid body at the panel centers.

Buoyancy forces can optionally be included with

$$\mathbf{f}_i = -\rho A_i h_i g \mathbf{n}_i, \tag{22}$$

where h_i is the depth of the panel center and g is the scalar gravitational constant. The minus sign is to signify that the force is in the direction opposite the surface normal of the panel.

Both forces integrate over surfaces and require that objects are closed.

4 Results

In our first example, we have a single data driven mode with 63 Eigenmodes. The artist input and the input after it has been projected to be divergence-free is shown in Figure 3. The Eigenmodes poorly capture this "jet," but represent gross flow well, while our enhanced basis captures both the gross flow and the jet well, see Figure 4.

Our second example uses a 128x128 grid with 67 Eigenmodes and contains two pairs of falling objects, each pair has one object above the other. After being released, the objects above catch up to the objects below, closing the gap between them. The objects that start out above, draft off of the objects below allowing them to fall faster through the fluid demonstrating the effects of solid-fluid coupling and object-object interaction, see Figure 5.

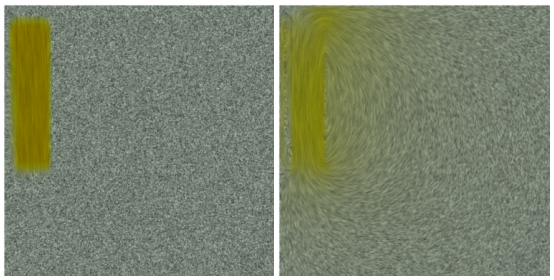

Figure 3: Left: Line integral convolution (LIC) is used to visualize the input from the artist. Right: The input from the artist after it has been projected to be divergence free.

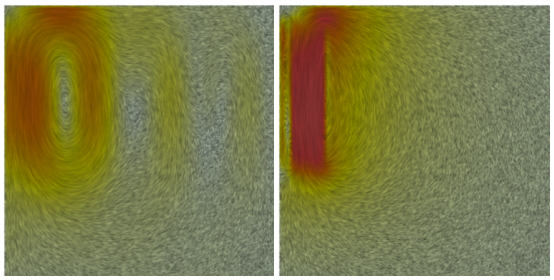

Figure 4: Left: Only Eigenmodes. Right: A data driven mode with Eigenmodes. When exciting the jet with high intensity, the induced flow is not well represented using only the Eigenmodes.

Finally, we have combined both our basis enhancement and two-way coupling into a simple 2D game, see Figure 6. The game uses 73 Eigenmodes and there are 15 objects with a total of 147 panels. Timing results in Table 1 show that the naive approach of computing feedback from the panel velocities to the reduced simulation dominates timing, taking $41ms$ in this example. By using our quadtree feedback approach we can reduce the time spent computing feedback by increasing the error threshold of the approximation. In practice, this error threshold can be quite large because this error is hidden by errors made when projecting the resulting velocity field into the reduced basis.

Description	Time (ms)
Advect	0.744
Diffuse	0.00536
Panel Solves	5.626
Panel Feedback Naive	41.175
Panel Feedback Quadtree	5.825

Table 1: Timings in ms for game scene with 73 modes on a 65x65 staggered grid.

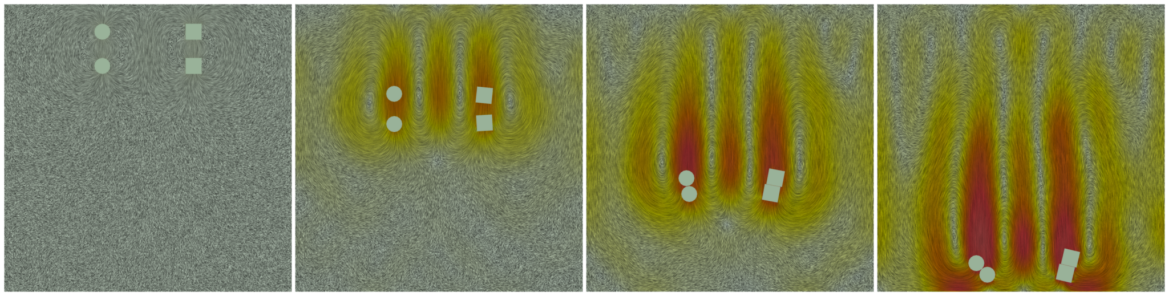

Figure 5: Drafting example: Objects above draft off of and catch up to the objects below. This example demonstrates solid-fluid coupling and object-object interactions.

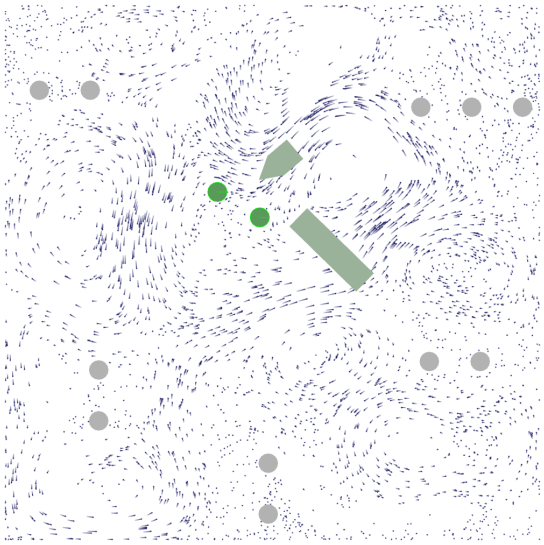

Figure 6: An image from a game using our system, see the video.

5 Conclusion and Future Work

We have presented several enhancements to dimensionally reduced fluid simulations: a basis enrichment scheme to mix data-driven and analytic modes, and a new approach to two-way solid-fluid coupling. Our enrichment scheme enables the combination of the generality of Eigenmodes with the context awareness and art directability of data-driven modes. Our approach to solid-fluid coupling combines vortex panel methods for solid-to-fluid coupling, dynamic pressure for fluid-to-solid coupling, the method of images to handle domain boundaries, and a quadtree-based method to accelerate the solid-to-fluid coupling. This approach enables robust coupling of dynamic objects to the dimensionally reduced fluid simulation and requires no training data.

In future work, we will extend the technique to 3D.

Acknowledgments

The authors wish to thank the anonymous reviewers for their helpful comments. This work was supported in part by a gift from Adobe Systems Incorporated and National Science Foundation Awards CNS-0855167, IIS-1249756, and IIS-1314896.

References

[1] Box2D. Erin Catto, 2011.

[2] Tyson Brochu, Todd Keeler, and Robert Bridson. Linear-time smoke animation with vortex sheet meshes. In *Proc. of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 87–95, 2012.

[3] Georges-Henri Cottet and Petros Koumoutsakos. *Vortex methods: Theory and practice*. Cambridge University Press, June 2000.

[4] Tyler De Witt, Christian Lessig, and Eugene Fiume. Fluid simulation using laplacian eigenfunctions. *ACM Trans. Graph.*, 31(1):10:1–10:11, February 2012.

[5] Abhinav Golas, Rahul Narain, Jason Sewall, Pavel Krajcevski, Pradeep Dubey, and Ming Lin. Large-scale fluid simulation using velocity-vorticity domain decomposition. *ACM Trans. Graph.*, 31(6):148:1–148:9, November 2012.

[6] Mohit Gupta and Srinivasa G. Narasimhan. Legendre fluids: a unified framework for analytic reduced space modeling and rendering of participating media. In *Proc. of the ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 17–25, 2007.

[7] J.L. Hess and A.M.O. Smith. Calculation of non-lifting potential flow about arbitrary three-dimensional bodies. Technical Report E.S. 40622, Douglas Aircraft Division, 1962.

[8] P. Holmes, J.L. Lumley, and G. Berkooz. *Turbulence, Coherent Structures, Dynamical Systems and Symmetry*. Cambridge Monographs on Mechanics. Cambridge University Press, 1996.

[9] Theodore Kim and John Delaney. Subspace fluid resimulation. *ACM Trans. Graph.*, 32(4):62:1–62:9, July 2013.

[10] Benjamin Long and Erik Reinhard. Real-time fluid simulation using discrete sine/cosine transforms. In *Proc. of the symposium on Interactive 3D graphics and games*, pages 99–106, 2009.

[11] Bradford J. Loos, Lakulish Antani, Kenny Mitchell, Derek Nowrouzezahrai, Wojciech Jarosz, and Peter-Pike Sloan. Modular radiance transfer. *ACM Trans. Graph.*, 30(6):178:1–178:10, December 2011.

[12] J. Lumley. The structure of inhomogeneous turbulent flows. *Atmospheric turbulence and radio wave propagation*, page 166178, 1967.

[13] J.L. Lumley. *Stochastic tools in turbulence*. Applied mathematics and mechanics. Academic Press, 1970.

[14] Sang Il Park and Myoung Jun Kim. Vortex fluid for gaseous phenomena. In *Proc. of the ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 261–270, 2005.

[15] Tobias Pfaff, Nils Thuerey, and Markus Gross. Lagrangian vortex sheets for animating fluids. *ACM Trans. Graph.*, 31(4):112:1–112:8, July 2012.

[16] P.G. Saffman. *Vortex Dynamics*. Cambridge University Press, 1995.

[17] L. Sirovich and Brown University. Division of Applied Mathematics. *Turbulence and the Dynamics of Coherent Structures*. Quarterly of applied mathematics. Brown University, Division of Applied Mathematics, 1987.

[18] Matt Stanton, Yu Sheng, Martin Wicke, Federico Perazzi, Amos Yuen, Srinivasa Narasimhan, and Adrien Treuille. Non-polynomial galerkin projection on deforming meshes. *ACM Trans. Graph.*, 32(4):86:1–86:14, July 2013.

[19] Andrea Toselli and Olof Widlund. *Domain Decomposition Methods - Algorithms and Theory*. Springer, 2004.

[20] Adrien Treuille, Andrew Lewis, and Zoran Popović. Model reduction for real-time fluids. *ACM Trans. Graph.*, 25(3):826–834, July 2006.

[21] Martin Wicke, Matt Stanton, and Adrien Treuille. Modular bases for fluid dynamics. *ACM Trans. Graph.*, 28(3):39:1–39:8, July 2009.