To appear in the ACM SIGGRAPH conference proceedings

Modular Bases for Fluid Dynamics

Martin Wicke[†] Matt Stanton[‡] Adrien Treuille[‡]
[†]Stanford University [‡]Carnegie Mellon University
[†]Max Planck Center for Visual Computing and Communication

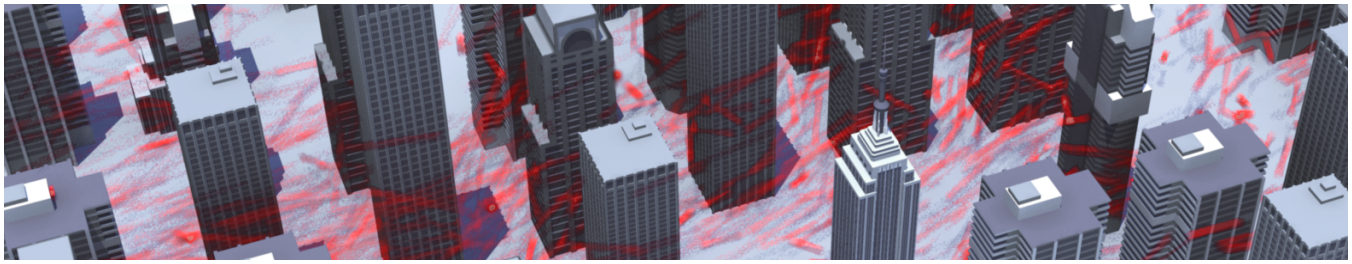

Figure 1: *In less than 2 seconds per frame, our method simulates fluids through this detailed city with over 120 million voxels, an* $4000\times$ *speedup compared to standard techniques. Our modular approach allows the user to rearrange building tiles at runtime.*

Abstract

We present a new approach to fluid simulation that balances the speed of model reduction with the flexibility of grid-based methods. We construct a set of composable reduced models, or *tiles*, which capture spatially localized fluid behavior. We then precompute coupling terms so that these models can be rearranged at runtime. To enforce consistency between tiles, we introduce *constraint reduction*. This technique modifies a reduced model so that a given set of linear constraints can be fulfilled. Because dynamics and constraints can be solved entirely in the reduced space, our method is extremely fast and scales to large domains.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation; I.6.8 [Simulation and Modeling]: Types of Simulation—Animation;

Keywords: reduced models, fluid simulation, domain decomposition, constraint reduction

1 Introduction

Offline physics-based animation can produce stunning examples of fluid motion. However, realtime applications generally simulate fluids only over small domains as a secondary effect. Fluid effects distant from the user are usually simulated at very low resolution or not at all. In the future, however, we expect interactive virtual environments with millions of simultaneous users where complex dynamic effects will play a primary role in the interaction: moving vehicles will splash through puddles as buildings leave turbulent eddies in the wind. Enabling such applications requires that such phenomena be computed *everywhere* with high resolution, and moreover that the dynamical state be consistent for all users. The difficulty is computing high resolution dynamics over such large environments. Existing fluid simulation approaches cannot scale to these large, realtime scenarios.

In 3D geometry, one of the principle solutions to creating complex virtual environments has been to exploit combinatorial explosion. For example, detailed cities can be created by tiling buildings drawn from a comparatively small set. New creatures and vehicles can be assembled from component parts. However, there has thus far been no analogue to this process in simulation.

This work proposes such a perspective on fluid dynamics. We precompute modular simulation *tiles* which capture fluid behavior given specific boundary conditions such as the presence of an obstacle. Each simulation tile is a reduced model created from high-resolution simulations. As shown by Treuille *et al.* [2006], these models allow very fast simulations with runtime complexity independent of the grid resolution. Tiles can be assembled at runtime to simulate novel fluid configurations. For example, we simulate wind through a large city composed of tiles specialized for each building type. Our results demonstrate that such tilings can scale to very large domains.

Our main contributions relate to tile coupling. We show that simulation operators can be precomputed, decomposed, and reconfigured at runtime based on the tiling, thus enabling novel tile configurations without additional precomputation. Simulation is fast because the dynamics operate entirely in the reduced space. We also show that the tiles can be constructed so as to maintain pairwise consistency at runtime. In general, reduced models have so few degrees of freedom that maintaining consistency quickly over-constrains the system. This can lead to severe artifacts at tile boundaries and unnatural behavior in the interior of the coupled tiles. We address this problem by introducing *constraint reduction*, an algorithm that modifies fluid tiles so that they can exactly fulfill a large number of linear constraints in the full-dimensional space. We show that this technique generalizes to arbitrary linear constraints. Like simulation, constraint satisfaction can be solved entirely in the reduced space. These techniques allow us to flexibly assemble fluid simulations on a scale previously unattainable in computer graphics.

2 Related Work

Fluid simulation in computer graphics has focused on three basic fluid representations: grid and mesh-based Eulerian simulations,

meshless Lagrangian methods, and model reduction. An important step in Eulerian fluid simulation was the introduction of an unconditionally stable advection step by Stam [1999]. Improvements to this method based on hierarchical space decomposition [Losasso et al. 2004] and non-uniform meshes [Feldman et al. 2005; Elcott et al. 2005] have produced impressive results, but do not typically allow real-time simulation. Recent work has mapped a number of Eulerian methods to the GPU, including stable advection [Wu et al. 2005], pressure projection [Krüger and Westermann 2003; Bolz et al. 2003; Goodnight et al. 2003], Lattice Boltzmann methods [Li et al. 2003], and the coupled map lattice [Harris et al. 2002]. These implementations enable real-time performance for medium-sized fluid domains; however, they fundamentally have the same time complexity as their CPU variants. Lagrangian particle representations such as vortex methods [Angelidis and Neyret 2005; Angelidis et al. 2006; Park and Kim 2005; Selle et al. 2005] and smoothed particle hydrodynamics [Müller et al. 2003; Zhu and Bridson 2005; Keiser et al. 2005; Adams et al. 2007] do not depend on grid resolution, but the effective resolution of the fluid depends on the particle density.

Our method uses a model reduced fluid representation, most closely resembling the approach introduced by Treuille *et al.* [2006]. In model reduction, the number of variables does not depend on the spatial resolution, making this technique ideally suited for real-time simulation of high-resolution dynamics. Model reduction has been used in a number of other branches of graphics including nonlinear deformation [James and Fatahalian 2003], finite element methods [Barbič and James 2005], and precomputed radiance transfer [Sloan et al. 2002]. Recently, Barbič and Popović [2008] demonstrate real-time control of such methods. Model reduction is also a topic of active research in the field of computational fluid dynamics and applied mathematics, where the technique is known as proper orthogonal decomposition (POD), Karhunen-Loéve decomposition, or subspace integration [Holmes et al. 1996; Rowley et al. 2006; Ausseur et al. 2004; Sirovich 1987; Couplet et al. 2005; Marion and Temam 1989; Sirisup and Karniadakis 2004; Lumley 1970].

Model reduced simulations are very fast, but highly inflexible: the user has little leeway to alter the simulation conditions after the model is constructed. Our work addresses this problem by allowing the user to rearrange a set of coupled reduced models. Perhaps the best known examples of reconfigurable tiles in computer graphics are Wang tiles [Cohen et al. 2003] which can be arranged to produce non-periodic textures or complex geometric scenes. Similarly, for fluids, Chenney [2004] introduces flow tiles, which produce divergence-free flows so long as the tiles are appropriately combined. In contrast to flow tiles, which are static vector fields, our method enables dynamic simulation.

Our tile representation requires explicit consistency constraints. A similar need to maintain consistency across simulation domains is encountered in finite element simulations of fluids, where the technique is called *domain decomposition*. To enforce coupling constraints arising on the boundaries between subdomains, one can add penalty terms [Farhat et al. 2001; Farhat et al. 2003; Tezaur et al. 2008; Zhang et al. 2006], or enforce strict compliance via Lagrange multipliers [Babuška 1973; Farhat et al. 2000; Tezaur and Farhat 2006]. Toselli and Widlund [2005] provide a good overview of these techniques. Finite element methods use analytic basis functions which are specifically designed such that boundary constraints can be satisfied. By contrast, the basis vectors used in our model are more expressive, but do not satisfy coupling constraints by construction.

To our knowledge, there is little work on coupling model reduced fluid simulations. LeGresley and Alonso [2003] decompose the simulation domain into a model reduced fluid simulation but use full resolution fluid simulation to capture fine scale features in certain areas. However, this work does not address the case of coupling two separately computed reduced simulations. Borggaard *et al.* [2006] tackle the problem of performing singular value decomposition (SVD) on a very large fluid simulation spatially partitioned across a set of processors. Their intent is to produce a single basis from this data, not a set of coupled reduced models. Perhaps the closest work to our own is that of Lucia and King [2002], who perform domain decomposition to isolate regions that contain shockwaves and then combine a set of model reduced simulations in order to capture shockwaves in high-speed flow fields. They use a penalty-based method to enforce continuity across boundaries, but do not address the problem of creating bases that can be spatially reconfigured while fulfilling continuity constraints at runtime.

Contrary to existing work on reduced fluid models which relies on SVD to compute the basis, we use SVD-based models as a starting point, and then modify the basis to enable coupling of previously incompatible tiles.

3 Fluid Tiles

The central idea of our approach is to cover the simulation domain with a small set of simulation primitives, called *tiles*. Each tile consists of a velocity basis representing the possible flow within its subdomain. For example, if one tile represents the fluid flow around the Empire State Building, then its basis spans a subset of possible flows around this obstacle. The boundaries between subdomains correspond to tile *faces*. To satisfy constraints across faces, we develop a set of rules governing the assembly of tiles: we create a small set of *boundary bases* associated with the tile faces. As long as adjacent faces share the same boundary basis, our construction guarantees that all constraints within the tiling can be satisfied.

3.1 Monolithic Fluid Reduction

In our algorithm, each tile corresponds to a spatially-localized linear model of fluid velocities. Simulation on such a representation is called model reduction. In this section, we briefly review the necessary basics of model reduction for fluid simulations, and we refer the reader to [Treuille et al. 2006] for more details.

First, consider a simulation with only one tile. The entire domain is then spanned by a single velocity basis. The unreduced simulation state is represented by a vector $\mathbf{u} \in \mathbb{R}^N$ consisting of the velocities defined at sample points. Our implementation uses a MAC grid [Foster and Metaxas 1996], although other velocity discretizations are possible. The reduced order model operates in an m-dimensional space spanned by basis states $\mathbf{B} = [\mathbf{b}_1 \dots \mathbf{b}_m]$. These basis states are computed as the right singular vectors to the m smallest singular values of a matrix of simulation snapshots $\mathbf{U} = [\mathbf{u}_1 \dots \mathbf{u}_k]$. In the following, we will denote the process of extracting these first m basis states as

$$\mathbf{B} = \mathrm{svd}_m\{\mathbf{U}\}. \tag{1}$$

Reduced states can be converted to full states by multiplying by the basis: $\mathbf{u} = \mathbf{B}\mathbf{r}$. Conversely, assuming $\mathbf{B}$ is orthonormal, we can project the full state into the reduced space with the transpose basis: $\mathbf{r} = \mathbf{B}^T\mathbf{u}$. Given any linear operator $\mathbf{X}$ acting on the full space, a reduced version of $\mathbf{X}$ can be computed as $\mathbf{B}^T\mathbf{X}\mathbf{B}$.

As demonstrated in [Treuille et al. 2006], the Navier Stokes equations can be reduced to

$$\frac{d\mathbf{r}}{dt} = \left(\mu\mathbf{D} + \sum_i \mathbf{A}_i r_i\right)\mathbf{r}, \tag{2}$$

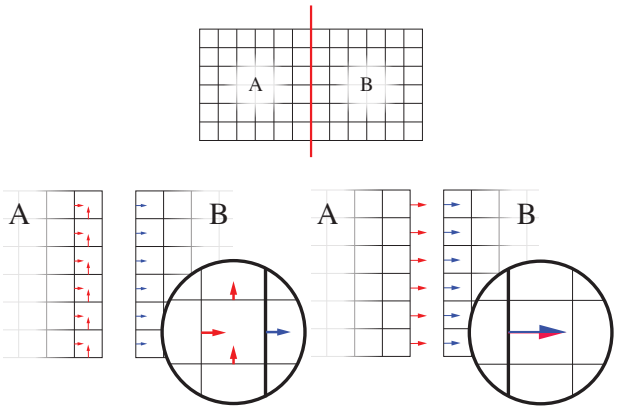

Figure 2: *Domain decomposition: Left: Decomposition into two domains, inducing divergence constraints in cells split between A and B. Right: Duplication of boundary values leads to equivalent equality constraints for duplicated values. The divergence of the cells can be enforced in each tile separately.*

where μ is the diffusion coefficient, $\mathbf{D}$ denotes the reduced diffusion operator, and $\mathbf{A}_i$ is a reduced linear operator that advects $\mathbf{r}$ using the velocity field $\mathbf{b}_i$. It is useful to consider the matrices $\mathbf{A}_i$ as slices of an *advection tensor* $\hat{\mathbf{A}}$ of rank three. During simulation, the advection matrix is computed by summing all slices weighted by their corresponding reduced state coefficient. Because the basis $\mathbf{B}$ spans only divergence-free velocity fields, incompressibility need not explicitly be enforced, unlike traditional Eulerian fluid simulations.

Treating the advection velocities as constant throughout each time step, we can integrate (2) analytically, leading to an integration that is both unconditionally stable and energy-preserving if the fluid is inviscid. Given a time step Δt, we compute the next reduced state as

$$\mathbf{r}(t+\Delta t) = e^{\Delta t\left(\mu\mathbf{D}+\sum_i \mathbf{A}_i r_i\right)}\mathbf{r}(t). \tag{3}$$

This matrix-vector product can be computed using iterative Taylor or Padé approximation, making the integration fast even for high-dimensional reduced models.

3.2 Tiled Fluid Reduction

The monolithic reduction described above is fast but inflexible. Even small changes to the simulation domain require complete recomputation of the model. We therefore replace the monolithic basis with a modular set of tiles that can be assembled at runtime. To obtain these tiles, we decompose the simulation domain. Within each subdomain, the fluid flow is computed by a reduced model.

Fig. 2 shows an illustration of domain decomposition. Consider a discretized domain which we split in two parts A and B as shown in Fig. 2. The Navier-Stokes equations include the constraint that the divergence must be zero across the simulation domain. This constraint must still be enforced when we split the domain. If cells are split by the decomposition, the divergence constraints lead to constraints involving all adjacent tiles (see Fig. 2, left). If the decomposition splits the domain such that velocities normal to the interface are defined on the boundary, we can also duplicate boundary velocities, as shown in Fig. 2, on the right. In this case, the aforementioned divergence constraints can be enforced in each tile separately. In order for the discretization to be consistent, the velocities twice defined on the boundary need to be equal, leading to equality constraints. Both interpretations are equivalent. In the following sections, we will use the second convention, since it simplifies that algorithmic description. We will now discuss how to compute tensors for a tiled domain, before returning to the contraints in Sec. 4.

Computing the necessary tensors for simulation using tiles can be treated as a special case of monolithic model reduction. Consider two reduced models A and B corresponding to domains $\mathcal{D}_A$ and $\mathcal{D}_B$ as in Fig. 2. A reduced basis $\mathbf{B}_A$ of A covers only $\mathcal{D}_A$, and we can consider it zero everywhere else. The same holds for $\mathbf{B}_B$. We can recombine the two bases into a combined basis of size $m = m_A + m_B$:

$$\mathbf{B} = \begin{bmatrix} \mathbf{B}_A & \mathbf{0} \\ \mathbf{0} & \mathbf{B}_B \end{bmatrix}. \tag{4}$$

Then, the state of the complete system can be written as $\mathbf{r} = [\mathbf{r}_A^T, \mathbf{r}_B^T]^T$. Diffusion and advection operators can now be computed as before. In particular, for the basis in equation (4), the diffusion operator becomes

$$\mathbf{D} = \begin{bmatrix} \mathbf{D}^{AA} & \mathbf{D}^{AB} \\ \mathbf{D}^{BA} & \mathbf{D}^{BB} \end{bmatrix}, \tag{5}$$

where the *interior* terms $\mathbf{D}^{AA}$ and $\mathbf{D}^{BB}$ depend only on $\mathbf{B}_A$ or $\mathbf{B}_B$, respectively, while the *coupling* terms $\mathbf{D}^{AB}$ and $\mathbf{D}^{BA}$ depend on both $\mathbf{B}_A$ and $\mathbf{B}_B$.

The situation for the advection tensor $\hat{\mathbf{A}}$ is similar. However, since $\hat{\mathbf{A}}$ is a rank three tensor, it has eight components $\hat{\mathbf{A}}^{AAA}, \hat{\mathbf{A}}^{AAB}, \ldots, \hat{\mathbf{A}}^{BBB}$. The six blocks with mixed superscripts are coupling terms. Because the full-dimensional diffusion and advection operators are sparse and highly localized in space, the cost of computing the coupling terms is proportional only to the size of the interface, not the full dimension N.

This construction can be generalized to arbitrary tilings of the domain. Because the adjacency graph between spatial subdomains is sparse, the combined advection and diffusion operators are also block-sparse.

4 Constraints

As described in 3.2, velocities are defined twice along the interface $\mathcal{F}$ between adjacent subdomains $\mathcal{D}_A$ and $\mathcal{D}_B$. To keep the simulation consistent between tiles, we therefore obtain equality constraints

$$\mathbf{u}_A(\mathbf{x}) = \mathbf{u}_B(\mathbf{x}) \qquad \forall \mathbf{x} \in \mathcal{F}. \tag{6}$$

We can assemble these constraints into a constraint matrix $\mathbf{C}$ which is satisfied when $\mathbf{CBr} = 0$. The system is overconstrained as soon as the number of samples in $\mathcal{F}$, N_b, exceeds the number of combined degrees of freedom $m_A + m_B$. Solving for $\mathbf{r}$ will therefore yield the trivial solution $\mathbf{r} = 0$. A naïve solution turns these constraints into a penalty by solving for the updated state $\mathbf{r}'$ that minimizes

$$\|\mathbf{CBr}'\|^2 + \alpha\|\mathbf{r} - \mathbf{r}'\|^2. \tag{7}$$

The regularization parameter α balances between constraint satisfaction and state modification. Large values of α allow inconsistent boundary velocities, leading to serious simulation artifacts. On the other hand, as $\alpha \to 0$, large corrections are applied to achieve an admissible state. If the tiles are too different, this leads to locking: only a very low-dimensional subspace of states can be represented by adjacent tiles, and the simulation will be *locked* into this subspace. The accompanying video shows an example of these artifacts.

4.1 Constraint Reduction

To solve this problem, we introduce the *constraint reduction* method. We will modify the basis vectors $\mathbf{B}$ to allow exact constraint satisfaction while preserving sufficient degrees of freedom for the simulation.

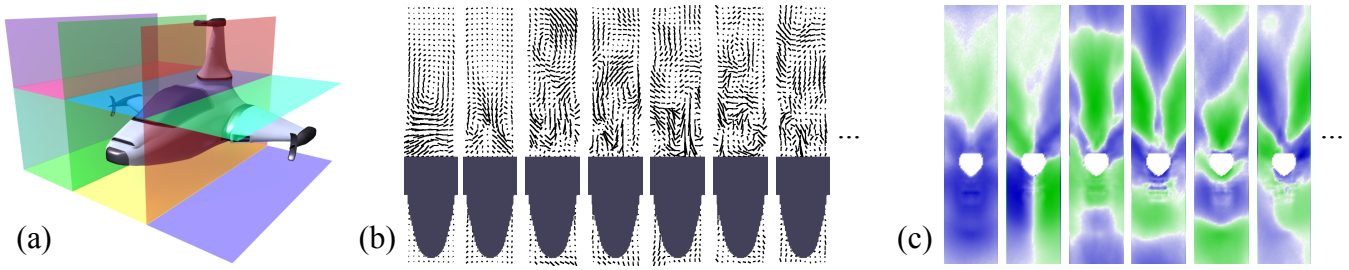

Figure 3: *(a) Decomposition of the spacecraft. The domain is split into six subdomains. Two parts are empty, while the other subdomains contain the wings, the body and the tail, respectively. (b) Slices through the velocity basis of the spacecraft body. (c) Basis vectors from the boundary basis between body and tail of the spacecraft. Blue and green represent positive and negative flow across the boundary.*

Consider again the two adjacent tiles A and B. The mixed constraints between those tiles (i. e. the constraints depending on values from both A and B) are the equality constraints (6). We can write these constraints as:

$$\mathbf{CBr} = \mathbf{C}_r\mathbf{B}_A\mathbf{r}_A - \mathbf{C}_l\mathbf{B}_B\mathbf{r}_B = \mathbf{e}_A - \mathbf{e}_B = \mathbf{0}, \tag{8}$$

where $\mathbf{C}_r$ selects the right face of A, and $\mathbf{C}_l$ selects the corresponding velocities on the left face of B. Both $\mathbf{e}_A$ and $\mathbf{e}_B$ are of dimension N_b, the number of velocity samples on the interface. If we can make sure $\mathbf{e}_A$ and $\mathbf{e}_B$ lie in the same linear space $\mathcal{S}$ with low dimension $s < m$, only s degrees of freedom are needed to fulfill the constraints. We can enforce this condition for all basis vectors. We will therefore construct a new basis $\tilde{\mathbf{B}}_A = [\tilde{\mathbf{b}}^A_1, \ldots, \tilde{\mathbf{b}}^A_{m_A}]$ such that

$$\mathbf{C}_r\tilde{\mathbf{b}}^A_i \in \mathcal{S} \qquad \forall i \in \{1 \ldots m_A\}, \tag{9}$$

and similarly for $\mathbf{B}_B$. In this case, the space $\mathcal{S}$ represents the allowed boundary states at the interface between A and B.

We construct $\mathcal{S}$ as a reduced model of observed boundary states. We extract the boundary values of all bases that should be made compatible in a database $\mathbf{E}$. In our example involving only two bases, this database is $\mathbf{E} = [\mathbf{C}_l\mathbf{b}^A_{1\ldots m_A}, \mathbf{C}_r\mathbf{b}^B_{1\ldots m_B}]$. Given $\mathbf{E}$, we compute an $N_b \times s$ *boundary basis* for $\mathcal{S}$: $\mathbf{S} = \mathrm{svd}_s\{\mathbf{E}\}$. See Fig. 3 (c) for an example. Often, more than two tiles share a compatible boundary. In these cases, $\mathbf{E}$ is assembled from all bases involved, or form a different set of examples representative of the flow patterns across the boundary.

Given the boundary basis, we can compute the modified bases $\tilde{\mathbf{B}}_A$ and $\tilde{\mathbf{B}}_B$. For each basis vector, we first project each of its faces into the appropriate boundary basis. Since this breaks the zero-divergence constraints inside the domain, we then fix the boundaries and find the closest divergence-free field given the boundary conditions by Helmholtz-Hodge decomposition. Finally, we reorthonormalize the basis using the standard Gram-Schmidt process.

Our modified basis now satisfies (9), and we can proceed to solve the constraint satisfaction problem. Since the constraint violations $\mathbf{e}_A$ and $\mathbf{e}_B$ lie in $\mathcal{S}$, we can transform them into a basis for $\mathcal{S}$ by rewriting (8) as

$$\mathbf{S}^T\mathbf{C}\tilde{\mathbf{B}}\mathbf{r} = \mathbf{M}\mathbf{r} = \mathbf{0}, \tag{10}$$

where we recombined the constraints into one system $\mathbf{C}$ and use the combined modified basis $\tilde{\mathbf{B}}$. $\mathbf{M}$ is an $s \times m$ matrix, revealing the effective dimension of the constraints applied to the modified basis. Note that we have not compromised on any of the constraints — all constraints are fulfilled exactly. We did, however, modify the basis in order to do so, potentially losing optimal reconstruction properties of the SVD-based construction.

In order to find a state that satisfies the constraints, we can now solve (10). We can choose m and s such that enough degrees of freedom are left even if a tile is constrained from all sides. In a three dimensional tiling of space, there will be three constrained boundaries per tile. For m dimensions per tile and s dimensional boundary bases, we therefore have to choose $m > 3s$ to avoid locking. When building large scenes from many coupled tiles, the reduced constraint matrix $\mathbf{M}$ is block-sparse as only adjacent tiles have non-zero entries.

Note that due to the way we decompose the domain, we only constrain velocities normal to the interface. These constraints are equivalent to zero-divergence constraints that arise when cells are split between domains instead of duplicating values. One caveat is that the constraints do not enforce smoothness of the velocity field. It is therefore possible that the velocities tangential to the interface are discontinuous at the interface. Smoothness is typically maintained by viscosity, but it can be enforced by adding constraints on the tangential velocities close to the interface. This can increase numerical accuracy at the cost of requiring a higher value of s to represent the higher-dimensional boundary states. We have included a comparison of simulations with both types of constraints in the accompanying video material.

4.2 General Constraints

Before we turn to the algorithmic details in Sec. 5, let us discuss the case of general linear constraints. We again have a set of linear constraints involving a number of reduced models A_i with bases $\mathbf{B}_{A_i}$:

$$\sum_i \mathbf{C}_{A_i}\mathbf{B}_{A_i}\mathbf{r}_{A_i} = \mathbf{0}. \tag{11}$$

As before, we modify each basis $\mathbf{B}_{A_i}$ such that each $\mathbf{C}_{A_i}\mathbf{B}_{A_i}\mathbf{r}_i$ lies in a small space $\mathcal{S}$ with dimension $s < m_{A_i}$. Note that the constraints do not need to be spatially localized or sparse. Potentially, all components of $\mathbf{B}_{A_i}\mathbf{r}_{A_i}$ could be referenced in each constraint. However, it is crucial that the bases $\mathbf{B}_{A_i}$ can be modified such that the constraint violations for each reduced model A_i lie in a small subspace $\mathcal{S}$. In the general setting, we then find bases $\tilde{\mathbf{B}}_{A_i} = [\tilde{\mathbf{b}}^{A_i}_1 \ldots \tilde{\mathbf{b}}^{A_i}_m]$ such that

$$\mathbf{C}_{A_i}\tilde{\mathbf{b}}^{A_i}_j \in \mathcal{S} \quad \forall j \in \{1 \ldots m_{A_i}\}, \tag{12}$$

while minimizing the distortion to the bases:

$$\|\tilde{\mathbf{b}}^{A_i}_j - \mathbf{b}^{A_i}_j\|. \tag{13}$$

Note that $\mathbf{C}_{A_i}$ can include constraints only affecting a single basis $\mathbf{B}_{A_i}$, such as zero-divergence constraints.

The two-step technique described in Sec. 4.1 finds an approximate minimum of Eq. 13 while fulfilling the constraints exactly. Since

	// Decompose domain, choose basis dimension m	
1	**forall** *raw bases* $\mathbf{B}_i$ **do**	
2	Compute examples $\mathbf{U}_{\mathbf{B}_i}$	$\mathcal{O}(n_e N^{4/3})$
3	Compute raw basis $\mathbf{B}_i = \mathrm{svd}_m\{\mathbf{U}_{\mathbf{B}_i}\}$	$\mathcal{O}(n_e^2 N + n_e^3)$
4	**forall** *boundary types* $\mathbf{S}_j$ **do**	
	// Collect boundary states $\mathbf{E}_j$ *from all relevant* $\mathbf{B}_i$*, choose* s	
5	Compute boundary basis $\mathbf{S}_j = \mathrm{svd}_s\{\mathbf{E}_j\}$	$\mathcal{O}((tm)^2 N_b + (tm)^3)$
6	**forall** *tiles* k **do**	
	// Choose basis i *and boundary types* $j_1 \ldots j_6$	
7	Compute modified basis $\tilde{\mathbf{B}}_k$	$\mathcal{O}(mN^{4/3})$
8	Compute interior tensor blocks $\hat{\mathbf{A}}^{AAA}$ and $\mathbf{D}^{AA}$	$\mathcal{O}(m^3 N)$
9	**forall** *pairs of tiles* $(\tilde{\mathbf{B}}_1, \tilde{\mathbf{B}}_2)$ **do**	
10	Compute coupling tensors between $\tilde{\mathbf{B}}_1$ and $\tilde{\mathbf{B}}_2$	$\mathcal{O}(m^3 N_b)$

Algorithm 1: *Tile creation from examples.*

the modifications to each basis vector are typically small, not exactly finding the global minimum does not lead to noticeable artifacts.

5 Algorithmic Details

Algorithm 1 summarizes the necessary steps to set up a tiled simulation. For each of the t tiles, we first compute n_e examples by taking snapshots of a full simulation within the domain of the tile which has N degrees of freedom (line 2). These examples are distilled into a raw basis of dimension m as described in Sec. 3.1 (line 3). Before computing the SVD, we choose an orthogonal set of examples. Typically, this drastically cuts the number of examples taken into account in the SVD, while retaining virtually all information. We are then left with a $N \times \mathcal{O}(n_e)$ matrix. By computing only right singular values, the computational complexity of performing SVD is only linear in N.

Similarly, we extract the boundary velocities relevant for each boundary type j from the raw bases and collect these boundary states in a matrix $\mathbf{E}_j$ which is of size $N_b \times \mathcal{O}(tm)$, where N_b is the number of samples in a boundary face. An SVD extracts a boundary basis S_j of dimension m_b (line 5).

The raw bases and boundary bases are then combined into tiles. For each raw basis $\mathbf{B}_i$, we chose boundary bases for each face that will be coupled to other tiles. We then perform constraint reduction and compute a modified basis $\tilde{\mathbf{B}}_i$ (line 7). As described in Sec. 4.1, this process requires solving a sparse linear system for each basis vector, yielding a total cost of $\mathcal{O}(mN^{4/3})$ using conjugate gradients[1].

Finally, we compute advection and diffusion operators for the tile bases $\tilde{\mathbf{B}}_i$ (line 8). Computing the interior advection tensors $\hat{\mathbf{A}}^{AAA}$ for each tile dominates the computational cost. The interior diffusion tensors $\mathbf{D}^{AA}$ require only $\mathcal{O}(m^2 N)$ time. The coupling terms must be computed for each pair of tiles that is to be coupled; for k tiles, there are $\mathcal{O}(k^2)$ such pairs. However, the coupling tensor computation involves iterating over the boundary region only, yielding the more favorable $\mathcal{O}(m^3 N_b)$ time complexity (line 10).

Once a good set of boundary bases has been computed, we can add tiles to our library without having to touch existing tiles. After choosing appropriate existing boundary bases, we perform constraint reduction and compute interior tensors *only* on the new tile (lines 7 and 8). To enable coupling to all existing tiles, we need to compute $\mathcal{O}(k)$ coupling tensors (line 10).

[1]The cost for solving a linear system of size N representing a 3D finite difference discretization of an elliptic boundary value problem can be solved in $\mathcal{O}(N^{4/3})$ time using conjugate gradients [Shewchuk 1994]. The systems treated herein are of this type.

1	**if** *not initialized or connectivity changed* **then**	
2	Assemble tensors $\hat{\mathbf{A}}$ and $\mathbf{D}$	$\mathcal{O}(k)$
3	Contract advection tensor $\mathbf{A} = \sum_i \mathbf{A}_i \mathbf{r}_i$	$\mathcal{O}(km^3)$
4	Assemble $\mathbf{M} = \Delta t(\mathbf{A} + \mu\mathbf{D})$	$\mathcal{O}(km^2)$
5	Compute preliminary state $\mathbf{r}' = e^{\mathbf{M}}\mathbf{r}$	$\mathcal{O}(km^2)$
6	Project state: solve (7) for new state $\mathbf{r}$	$\mathcal{O}(k^{4/3}m^{8/3})$
7	Advect particles	$\mathcal{O}(n_p(m+k))$

Algorithm 2: *Computations performed in each time step.*

Algorithm 2 summarizes the computations during runtime. We are given the adjacency graph between tiles, and assemble the global tensors from their precomputed parts at the start of the simulation and whenever the adjacency graph changes (lines 1–2). In each time step, we first contract the global advection tensor. Since each tile instance has only a fixed number of neighbors (six in three dimensions), the assembled global operators are block-sparse with $7m$ entries per row or column, leading to a total cost of $\mathcal{O}(km^3)$ for evaluating line 3. Time integration is performed using Eq. 3. We use Taylor expansion to approximate the matrix-vector multiplication with the matrix exponential (line 5). This requires only one sparse matrix-vector multiplication and one vector addition per iteration. In all our experiments, the Taylor approximation of the matrix exponential converges to machine precision in fewer than 20 iterations.

After integration, we project the system into the admissible space defined by the constraints. This requires the solution of the block-sparse system (10). Eq. 10 is underconstrained, and has an $m - s$ dimensional solution space. To disambiguate the system, we add a small regularization term as in Eq. 7: we use $\alpha = 10^{-8}$ for double and $\alpha = 10^{-4}$ for single precision. Since constraints are restricted to adjacent tile instances, the number of non-zero entries per row never exceeds $7m$, similar to the advection and diffusion tensors (line 6).

We thus obtain a sequence of reduced states that can be used for evaluation or visualization. In our experiments, we use massless marker particles for flow visualization. For each particle, we need to check which of the k tile instances currently affect it, and compute the velocity at its current position by computing a weighted sum of all m basis velocities at the particle position. For n_p particles, this leads to the total cost of $\mathcal{O}(n_p(m + k))$ for particle advection.

Note that particle advection is the only step that requires the full basis present in memory. All other computations require only reduced size structures (column "Tensors" in Table 1). Note also that particle advection is trivially parallelizable.

6 Results

To assess the properties of coupled fluid tiles, we have conducted a variety of experiments and comparisons. Timings exclude visualization (particle advection, particle filtering, and rendering).

Simulation Error. To measure our algorithm's approximation error, we perform simple tests in 2D and 3D. In both cases, we ran a simulation of horizontal wind evolving into vortices over 200 frames. Animations are contained in the accompanying video material. We compare the full simulation at different resolutions to monolithic model reduction [Treuille et al. 2006] with 32 basis states (64 in 3D), and coupled model reduction over a pair of adjacent tiles with 16 basis states each (32 in 3D) and a 6-dimensional boundary basis. The results are summarized in Figure 4.

Averaging over the whole domain, the errors for tiled and monolithic simulations are similar. In terms of $\mathcal{L}^2$-error, the coupled re-

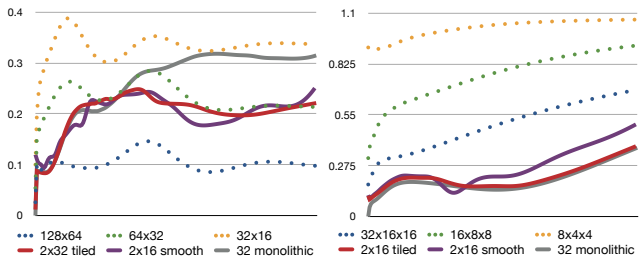

Figure 4: ***Simulation Error*** *compared to a ground truth simulation. Left: 2D test with* 256×128 *ground truth. Right: 3D test with* $64 \times 32 \times 32$ *ground truth. Shown is the relative* $\mathcal{L}^2$ *error:* $\delta(\mathbf{x}, \mathbf{y}) = ||\mathbf{x} - \mathbf{y}||/||\mathbf{x}||$ *plotted against time for 200 frames.*

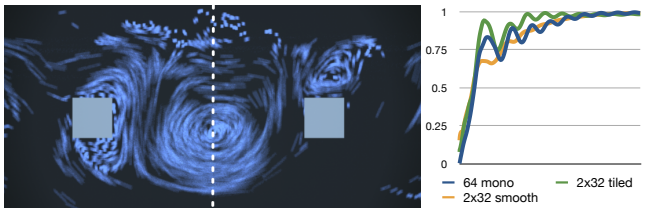

Figure 5: ***2D Boxes*** *example. Left: Vortices and other features can cross from one tile to another by virtue of the coupling basis. The dotted line indicates the tile boundary. The two squares are fixed obstacles. Right: Relative error of monolithic, tiled simulation, with and without smoothness constraints, plotted against time for 1000 frames.*

duced simulation outperforms a full-scale simulation that is downsampled by a factor of 16 (8 in 3D). Especially for larger domains, this indicates that reduced models are significantly faster than full simulations, even if we compare to a downsampled simulation incurring similar errors. These results measure error in cases where the test case is close to the training data.

2D Boxes. Modular reduced models show simulation errors similar to monolithic reduced models even in situations where tiles are combined in novel ways. To demonstrate this, we ran a 150×50 simulation of horizontal flow through a domain containing two square obstacles. We then split the domain into three 50×50 subdomains called, from left to right, A, B, and C. Both A and C contain obstacles; B does not. Finally, we discard B and compute tiles for A and C independently. The reduced models for A and C are then coupled to form a tiled simulation (see Fig. 5).

For comparison, we ran a full simulation using the same boundary conditions as in the coupled simulation seen in Fig. 5. We also compute a monolithic reduced model from this new full-dimensional simulation. We then compare the tiled simulation and the monolithic reduced model against the new "ground truth" simulation. Note that in this experiment, the monolithic model is tested with its own training data, while the tiled model is not. Nevertheless, both techniques show similar errors. The relative $\mathcal{L}^2$ error approaches one because the reduced models dissipate energy faster than the full simulation (we calibrated the diffusion in all models be visually similar, leading to higher dissipation in the reduced models).

The $\mathcal{L}^2$ error is a crude measure of simulation quality, especially in the case of turbulent flows. Absent a good error measure, we have visually evaluated the performance of our approach by testing its ability to transfer vortices across tile boundaries. In order to highlight interesting areas of the flow, we advect a large number of particles and filter for a subset whose paths have high curvature. These particles tend to best show the important features of the flow.

Tiled simulation using boundary bases produces results that are far superior to simpler alternatives. We refer the reader to the accompanying video for animated comparisons. Without coupling constraints, divergence along the boundary creates severe artifacts. Overlapping the two bases and blending between their velocities yields an approximately divergence-free flow, but flow features such as vortices are not transferred across the tile boundaries. Fulfilling the consistency constraints approximately without performing constraint reduction leads to a locked simulation (i. e. a small value for α in Eq. 7), or insufficient coupling and divergence artifacts (i. e. a large value for α in Eq. 7). Constraint reduction solves these issues by making the tiles compatible, thus allowing for information exchange across the boundary while fulfilling the consistency constraints exactly.

Spacecraft. This example demonstrates our algorithm's ability to recombine pieces of geometry simulated entirely independently. To compute the tiles for this scene, we ran a total of 15 full dimensional simulations of incompressible fluid flow on a three million voxel $248\times196\times71$ domain. We simulated five different "wind" directions for each of the three spacecraft designs. We then decomposed each spacecraft into wings, tail, and body. We thus obtained 14 tiles: three wing pairs, three body parts, three tails, and two additional tiles above the wings.

We created 16-dimensional boundary bases for each of the possible boundaries shown in Fig. 3 (a). This enables like parts to be interchanged at runtime. Finally, we created 64-dimensional simulation bases for each of the 14 tiles. The accompanying video shows real-time simulations including part substitutions and wind variations within a 90° range. The results show that our approach captures the turbulent wakes left by this obstacle even for tile combinations not originally simulated.

City. We simulated a large city, demonstrating that our approach enables interactive simulation on huge domains. We performed 90 fluid simulations for 200 frames of various 2×2 building configurations on a $114\times114\times146$ domain. The simulations were initialized with wind from one of the four compass directions or a turbulent initial state without net wind. For each building type, we extracted time series for a $57\times57\times146$ domain from the simulations. We then built two 12-dimensional coupling bases for the x and y faces, respectively, and computed 72-dimensional simulation bases for each building from these coupling bases. In this setting, every building can be coupled with every other building. We use these bases to build a set of cities ranging from 5×5 to 16×16 blocks. Again, we visualize flow using filtered particles. The results in the accompanying video show that our technique captures small scale simulation features, turbulent wakes that cross tile boundaries, and enables runtime part substitution and simulation modification such as the introduction of tornadoes or the removal or changing of tiles. Fig. 1 shows pathlines of particles advected in the flow field of a city consisting of 16×16 blocks.

Note that the tornado tile was added to our library after all the other building blocks had already been computed. As pointed out in Sec. 5, if the boundary bases are left unchanged, the existing tiles do not have to be modified in order to add new tiles.

Table 1 shows statistics on scene preparation, precomputation times and runtime performance. Both the full-dimensional and coupled simulations were performed on a quad-core 1.1 GHz AMD Opteron with 16GB of RAM. The reported simulation times include simulation of the coupled reduced system and constraint handling. Timings do not include particle advection, particle filtering, and rendering. We use Mental Ray to render the scenes.

			Reduced Sim				Runtime			Memory		
Name	**Shape**	**Voxels**	t	m	s	**Precomp.**	**Full**	**Reduced**	**Speedup**	**Full**	**Bases**	**Tensors**
Spacecraft	$248\times196\times71$	3.4M	14	64	16	33h	191s	0.024s	**7919**×	53MB	5.4GB	128MB
City 5x5	$285\times285\times146$	12M	7	72	12	26h	436s	0.108s	**4029**×	181MB	1.5GB	31MB
City 7x7	$399\times399\times146$	23M	7	72	12	26h	∼850s†	0.250s	∼3400×	355MB	2.3GB	73MB
City 16x16	$912\times912\times146$	121M	7	72	12	26h	∼4,400s†	1.626s	∼3900×	1.8GB	2.7GB	120MB

Table 1: ***Timing and memory summary***. *t, m and s denote the number of tiles, reduced dimension and the dimension of the boundary basis, respectively. "Precomp." is the precomputation time in hours, while the "Full" and "Reduced" runtimes are measured in seconds. Memory usage for the full simulation includes velocity and pressure fields, "tensors" includes all coupling terms (even those not used in the scene). All memory requirements assume single precision floating point storage. †Simulation time estimated by extrapolation.*

The speedups we reported are approximate. We did not heavily optimize either our full dimensional or our coupled simulation runtime. The former could certainly be improved by using preconditioning in the projection step, and the latter could be optimized by removing dynamic memory allocation from the simulation loop and improving cache coherence. Furthermore, for the 7×7 and 16×16 cities, the full simulation was too large to run on our machines (and implementations), and the simulation runtime is approximated by assuming the simulation cost per time step is linear in the number of voxels. This is an underestimation, the asymptotic complexity is $\mathcal{O}(N^{4/3})$ assuming the full-dimensional projection step is solved to convergence.

7 Conclusion

We have presented a novel approach to interactive fluid simulation. The central idea is to distill high-resolution simulation data into fluid tiles that can be combined in a modular fashion. Our results demonstrate that tilings can scale to very large domains. Runtime complexity is low because dynamics and coupling operate entirely in the low-dimensional reduced space. We believe this technique brings us closer to complex virtual environments endowed with high resolution dynamics.

Our technical contributions relate to coupled simulation and constraint satisfaction across tiles. The tile representation induces simulation operators which can be precomputed and decomposed, enabling assembly and reconfiguration at runtime. Moreover, the approach enables extensible libraries of tiles: we can introduce new tiles without recomputing information about existing tiles.

Our other main contribution is constraint reduction. We modify the simulation bases so that they can satisfy a large set of consistency constraints while preserving sufficient freedom in the representation to prevent locking. This technique generalizes to arbitrary linear constraints on any reduced model. Beyond enforcing consistency, the method opens up interesting possibilities for adapting reduced models to linear constraints in a post-process.

The models generated this way are one step beyond models generated using plain singular value decomposition. Using our method, tiles can be assembled at runtime to produce fluid simulations with obstacles not contained in the original data. This modularity overcomes one of the principle limitations of model reduction: the inability to adapt the model once it has been computed.

Otherwise, our approach shares some of the limitations of monolithic model reduction. Depending on the size of the domain, the memory cost of storing bases can be high. Also, representational limits of the basis incur greater accuracy costs than full simulation. As expected from a data-driven technique, these errors are particularly noticeable when the reduced system is presented with inputs far from the training data. Finally, like other reduced order techniques for fluid simulation, our method cannot be used to simulate multi-phase flows. In particular, fluids with free surfaces cannot be properly handled.

Generating the simulation data to construct fluid tiles requires some care. It is possible to create fundamentally incompatible tiles which lose much of the representational power during constraint reduction. This suggests a difficult but exciting open problem: Can we generate bases that preserve their full representational power when subject to coupling?

We hope to extend the scope of this approach to a wider range of phenomena, such as explosions, elastic dynamics, free-surface fluids, and other complex phenomena. This includes coupling different types of reduced models in the same simulation.

We close by observing that fluid tiles are flexible, but less so than grid representations, and they are efficient, but not as fast as pure model reduction. As such, this work opens a continuum between these two extreme approaches. Instead of grid cells with three degrees of freedom, tiles represent richer fluid behavior within their larger subdomains. Extending this analogy, we are excited to adapt fluid tiles to general velocity samplings, such as overlapping tiles, and to a Lagrangian setting where the tiles are allowed to move with respect to each other.

Acknowledgments

We would like to thank Moshe Mahler for rendering and producing the video, Autodesk for donating Maya software, and the Moore Foundation for donating our compute cluster. This paper has been greatly improved by the comments of the anonymous reviewers. Martin Wicke is funded by a postdoctoral fellowship of the Max Planck Center for Visual Computing and Communication.

References

ADAMS, B., PAULY, M., KEISER, R., AND GUIBAS, L. J. 2007. Adaptively sampled particle fluids. In *Proc. SIGGRAPH '07*.

ANGELIDIS, A., AND NEYRET, F. 2005. Simulation of smoke based on vortex filament primitives. In *Proc. SCA '05*.

ANGELIDIS, A., NEYRET, F., SINGH, K., AND NOWROUZEZAHRAI, D. 2006. A controllable, fast and stable basis for vortex based smoke simulation. In *Proc. SCA '06*.

AUSSEUR, J., PINIER, J., GLAUSER, M., AND HIGUCHI, H. 2004. Predicting the Dynamics of the Flow over a NACA 4412 using POD. *APS Meeting Abstracts*, D8.

BABUŠKA, I. 1973. The finite element method with Lagrangian multipliers. *Numer. Math. 20*, 3.

BARBIČ, J., AND JAMES, D. 2005. Real-time subspace integration for St. Venant-Kirchhoff deformable models. In *Proc. SIGGRAPH '05*.

BARBIČ, J., AND POPOVIĆ, J. 2008. Real-time control of physically based simulations using gentle forces. *ACM Transactions on Graphics 27*, 5.

BOLZ, J., FARMER, I., GRINSPUN, E., AND SCHRÖDER, P. 2003. Sparse Matrix Solvers on the GPU: Conjugate Gradients and Multigrid. In *Proc. SIGGRAPH '03*.

BORGGAARD, J., GUGERCIN, S., AND ILIESCU, T. 2006. A domain decomposition approach to POD. *IEEE Conference on Decision and Control*.

CHENNEY, S. 2004. Flow tiles. In *Proc. SCA '04*.

COHEN, M. F., SHADE, J., HILLER, S., AND DEUSSEN, O. 2003. Wang tiles for image and texture generation. In *Proc. SIGGRAPH '03*.

COUPLET, M., BASDEVANT, C., AND SAGAUT, P. 2005. Calibrated reduced-order POD-Galerkin system for fluid flow modelling. *J. Comput. Phys. 207*, 1.

ELCOTT, S., TONG, Y., KANSO, E., SCHRÖDER, P., AND DESBRUN, M. 2005. Stable, circulation-preserving, simplicial fluids. In *Discrete Differential Geometry*, Chapter 9 of Course Notes. ACM SIGGRAPH.

FARHAT, C., TEZAUR, R., AND TOIVANEN, J. 2000. A domain decomposition method for discontinuous Galerkin discretizations of Helmholtz problems with plane waves and Lagrange multipliers. *Int. J. Numer. Meth. Engng*.

FARHAT, C., HARARI, I., AND FRANCA, L. P. 2001. The discontinuous enrichment method. *Comput. Methods Appl. Mech. Engrg. 190*.

FARHAT, C., HARARI, I., AND HETMANIUK, U. 2003. A discontinuous Galerkin method with Lagrange multipliers for the solution of Helmholtz problems in the mid-frequency regime. *Applied Mechanics and Engineering 192*, 1389–1419.

FELDMAN, B. E., O'BRIEN, J. F., AND KLINGNER, B. M. 2005. Animating gases with hybrid meshes. In *Proc. SIGGRAPH '05*.

FOSTER, N., AND METAXAS, D. 1996. Realistic animation of liquids. *Graphical Models and Image Processing 58*, 5.

GOODNIGHT, N., WOOLLEY, C., LUEBKE, D., AND HUMPHREYS, G. A. 2003. Multigrid solver for boundary value problems using programmable graphics hardware. In *Proceeding of Graphics Hardware*.

HARRIS, M. J., COOMBE, G., SCHEUERMANN, T., AND LASTRA, A. 2002. Physically-based visual simulation on graphics hardware. In *Graphics Hardware 2002*, 109–118.

HOLMES, P., LUMLEY, J. L., AND BERKOOZ, G. 1996. *Turbulence, Coherent Structures, Dynamical Systems and Symmetry*. Cambridge University Press.

JAMES, D. L., AND FATAHALIAN, K. 2003. Precomputing interactive dynamic deformable scenes. In *Proc. SIGGRAPH '03*.

KEISER, R., ADAMS, B., GASSER, D., BAZZI, P., DUTRE, P., AND GROSS, M. 2005. A unified lagrangian approach to solid-fluid animation. In *Proceedings Symposium Point-Based Graphics*.

KRÜGER, J., AND WESTERMANN, R. 2003. Linear algebra operators for GPU implementation of numerical algorithms. In *Proc. SIGGRAPH '03*.

LEGRESLEY, P. A., AND ALONSO, J. J. 2003. Dynamic domain decomposition and error correction for reduced order models. *41st AIAA Aerospace Sciences Meeting and Exhibit*.

LI, W., WEI, X., AND KAUFMAN, A. 2003. Implementing lattice Boltzmann computation on graphics hardware. *The Visual Computer 19*, 7-8.

LOSASSO, F., GIBOU, F., AND FEDKIW, R. 2004. Simulating water and smoke with an octree data structure. In *Proc. SIGGRAPH '04*.

LUCIA, D. J., AND KING, P. I. 2002. Domain decomposition for reduced-order modeling of a flow with moving shocks. *AIAA Journal 40*, 11, 2360–2362.

LUMLEY, J. L. 1970. *Stochastic Tools in Turbulence*, vol. 12 of *Applied Mathematics and Mechanics*. Academic Press.

MARION, M., AND TEMAM, R. 1989. Nonlinear Galerkin methods. *SIAM J. Numer. Anal. 26*, 5, 1139–1157.

MÜLLER, M., CHARYPAR, D., AND GROSS, M. 2003. Particle-Based Fluid Simulation for Interactive Applications. In *Proc. SCA '03*.

PARK, S. I., AND KIM, M. J. 2005. Vortex fluid for gaseous phenomena. In *Proc. SCA '05*.

ROWLEY, C., WILLIAMS, D., COLONIUS, T., MURRAY, R., AND MACMARTIN, D. 2006. Linear models for control of cavity flow oscillations. *J. Fluid Mech. 547*, 317–330.

SELLE, A., RASMUSSEN, N., AND FEDKIW, R. 2005. A vortex particle method for smoke, water and explosions. In *Proc. SIGGRAPH '05*.

SHEWCHUK, J. R. 1994. An introduction to the conjugate gradient method without the agonizing pain. Tech. Rep. CS-94-125, Carnegie Mellon University, Pittsburgh, PA, USA.

SIRISUP, S., AND KARNIADAKIS, G. E. 2004. A spectral viscosity method for correcting the long-term behavior of POD models. *J. Comput. Phys. 194*, 1, 92–116.

SIROVICH, L. 1987. Turbulence and the dynamics of coherent structures. I - Coherent structures. II - Symmetries and transformations. III - Dynamics and scaling. *Quarterly of Applied Mathematics 45* (Oct.), 561–571.

SLOAN, P.-P., KAUTZ, J., AND SNYDER, J. 2002. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In *Proc. SIGGRAPH '02*.

STAM, J. 1999. Stable Fluids. In *Computer Graphics (SIGGRAPH 99)*.

TEZAUR, R., AND FARHAT, C. 2006. Three-dimensional discontinuous Galerkin elements with plane waves and Lagrange multipliers for the solution of mid-frequency Helmholtz problems. *Int. J. Numer. Meth. Engng 66*.

TEZAUR, R., ZHANG, L., AND FARHAT, C. 2008. A discontinuous enrichment method for capturing evanescent waves in multiscale fluid and fluid/solid problems. *Comput. Methods Appl. Mech. Engrg. 197*.

TOSELLI, A., AND WIDLUND, O. 2005. *Domain Decomposition Methods - Algorithms and Theory*. Springer.

TREUILLE, A., LEWIS, A., AND POPOVIĆ, Z. 2006. Model reduction for real-time fluids. In *Proc. SIGGRAPH '06*.

WU, E., LIU, Y., AND LIU, X. 2005. An improved study of real-time fluid simulation on GPU. *Computer Animation and Virtual Worlds 15*, 3-4.

ZHANG, L., TEZAUR, R., AND FARHAT, C. 2006. The discontinuous enrichment method for elastic wave propagation in the medium-frequency regime. *Internat. J. Numer. Methods Engrg. 66*.

ZHU, Y., AND BRIDSON, R. 2005. Animating sand as a fluid. In *Proc. SIGGRAPH '05*.