To appear in ACM TOG 31(6).

Large-scale Fluid Simulation using Velocity-Vorticity Domain Decomposition

Abhinav Golas*[1], Rahul Narain†[2], Jason Sewall‡[3], Pavel Krajcevski§[1], Pradeep Dubey¶[3], and Ming Lin‖[1]

[1]University of North Carolina at Chapel Hill
[2]University of California, Berkeley
[3]Intel Corporation

(a) (b) (c)

Figure 1: *Examples of fluids simulated with our technique: (a) a city block hit by a tsunami (vortex domain in yellow) (b) seagulls flying through smoke (c) smoke flow around a sphere. We achieve up to three orders of magnitude of performance over standard grid-only techniques.*

Abstract

Simulating fluids in large-scale scenes with appreciable quality using state-of-the-art methods can lead to high memory and compute requirements. Since memory requirements are proportional to the product of domain dimensions, simulation performance is limited by memory access, as solvers for elliptic problems are not compute-bound on modern systems. This is a significant concern for large-scale scenes. To reduce the memory footprint and memory/compute ratio, vortex singularity bases can be used. Though they form a compact bases for incompressible vector fields, robust and efficient modeling of nonrigid obstacles and free-surfaces can be challenging with these methods.

We propose a hybrid domain decomposition approach that couples Eulerian velocity-based simulations with vortex singularity simulations. Our formulation reduces memory footprint by using smaller Eulerian domains with compact vortex bases, thereby improving the memory/compute ratio, and simulation performance by more than 1000x for single phase flows as well as significant improvements for free-surface scenes. Coupling these two heterogeneous methods also affords flexibility in using the most appropriate method for modeling different scene features, as well as allowing robust interaction of vortex methods with free-surfaces and nonrigid obstacles.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation; I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically based modeling;

Keywords: Computational fluid dynamics, vortex methods, stable fluids, Navier-Stokes equations

*golas@cs.unc.edu
†narain@eecs.berkeley.edu
‡jason.sewall@intel.com
§pavel@cs.unc.edu
¶pradeep.dubey@intel.com
‖lin@cs.unc.edu

1 Introduction

State-of-the-art methods for fluid simulation, including velocity-based Eulerian methods and smoothed particle hydrodynamics, model the entire *spatial* extent of the fluid. Discretization of this space is often chosen to be able to sample sufficiently fine details under the restriction of limited computational resources. As a result, scenes with large spatial scales can only be simulated to coarse detail on PCs, relying on procedural methods to infuse detail. The simple computational kernels of these methods are largely *memory-bandwidth bound*, since domains of interest cannot reside in caches of current generation CPUs, and computational complexity cannot mask the cost of memory accesses. An alternate approach to modeling fluids is to model fluid *detail*, represented by the *vorticity* of the fluid, *i.e.* the curl of the velocity field. For incompressible flows, vorticity can be compactly represented by Lagrangian singularity elements. They are thus free of numerical dissipation, which can be a significant issue with Eulerian methods, and do not need to explicitly model the pressure of the fluid. Though this leads to computational savings for scenes with unbounded fluid, robust and efficient modeling of obstacles or free-surfaces with two-way coupling using vorticity methods is challenging. Vortex singularity elements also serve as intuitive models for visual fluid detail, e.g. a smoke ring can be modeled as a vortex curve or filament.

These aspects make *detail* modeling of fluids with vortex singularity elements attractive, especially for large-scale scenes. To extend the applicability of these methods to free-surface fluids and scenes with deformable elements, we propose a hybrid domain-decomposition approach. Since Eulerian methods are adept at modeling such surfaces, coupling them with vortex methods can provide a robust, flexible, and efficient approach to fluid simulation. Particularly for scenes with large spatial scales but concentrated regions of detail, our approach can provide substantial computational and memory savings with reduced numerical dissipation, allowing simulations with more detail than previously possible.

It also poses significant challenges, the biggest of which is coupling heterogeneous methods in the same simulation while ensuring a consistent velocity field that matches the actual fluid velocity. We address this problem by separately coupling fluid flux and vorticity

across simulation boundaries. We propose an iterative coupling algorithm that matches fluid flux across boundaries using appropriate boundary conditions for grid simulations and by creating vortex singularities on the boundary for vortex simulations. To accurately transfer vorticity across boundaries, we use a novel vortex particle creation algorithm and velocity boundary conditions for advection. It is important to note that maintaining vortex surface elements is computationally expensive and numerically ill-conditioned if surface mesh quality is poor. We develop an approach that addresses both of these issues by constructing meshes using grid faces, and efficiently computing fluxes induced by these face elements, even allowing pre-computation. We extend hierarchical approach of [Lindsay and Krasny 2001] for faster computation of flux induced by vortex particles and surface elements.

Main Results: The key contributions of this work are:

- a hybrid fluid simulation algorithm that preserves vortex fluid features by using a compact vortex basis with Lagrangian elements in the interior of the fluid, while enforcing arbitrary boundary conditions such as nonrigid obstacles and free surfaces using an Eulerian grid representation near boundaries;
- a novel two-way coupling between Eulerian Navier-Stokes simulations and Lagrangian vorticity simulations, which conserves vorticity over time and ensures continuity in the velocity and vorticity fields;
- a sampling algorithm to create a vortex particle representation of a given velocity field by minimizing residual in the L^2 norm; and
- an efficient and well-conditioned approach to computing strength of vortex surface elements, and an $O(m \log m)$ algorithm to evaluate flux using hierarchical methods.

The coupling techniques introduced in this paper are quite general and can be used to connect different kinds of fluid simulation techniques in a heterogeneous domain decomposition framework. When applied to grid-based and vorticity-based methods, this method enables efficient simulation of large fluid volumes using a vorticity representation while supporting rich and complex interaction at boundaries. We demonstrate the benefits of this approach on several large-scale scenes (see Fig. 1) which would have a prohibitive computational cost using existing techniques.

2 Related Work

Physically-based simulation of fluids has been a major focus of computer graphics research over the past decade. In this section, we briefly review the work in this area that is most relevant to this work. These may can be classified into two broad categories: *velocity-based* methods, which discretize the velocity field of the fluid directly, and *vorticity-based* methods, which describe it indirectly by its curl (vorticity) instead. These methods may further classified in orthogonal categories based on the type of discretization, i.e. Eulerian or Lagrangian methods.

2.1 Velocity-based methods

Most of the *de facto* standard techniques for fluid simulation in computer graphics use a velocity-based representation. In such methods, one solves a discretization of the Navier-Stokes equations, which describe the evolution of the velocity field of a fluid over time.

A popular approach to solving these equations is by using finite difference methods on Eulerian grids — this was introduced to computer graphics by the pioneering work of Foster and Metaxas [1996]. Stam [1999] proposed semi-Lagrangian advection schemes to allow for unconditionally stable fluids, and Foster and Fedkiw [2001] developed methods for producing realistic, robust liquid surfaces in simulations. These methods have the advantage of being simple to implement and producing visually compelling results, including interactions with rigid and deformable objects [Chentanez et al. 2006; Batty et al. 2007; Robinson-Mosher et al. 2008]. Similar techniques have also been proposed using tetrahedral meshes [Wendt et al. 2007; Klingner et al. 2006; Chentanez et al. 2007] instead of rectilinear grids. However, Eulerian simulators have traditionally faced two challenges. First, they suffer from significant numerical dissipation, typically exceeding the desired viscosity in the fluid, causing flow detail to be undesirably damped out. Recent developments in improved advection schemes offer benefits in this aspect [Kim et al. 2007; Selle et al. 2008; Mullen et al. 2009; Lentine et al. 2011; Zhu and Bridson 2005], while techniques for introducing additional detail have also been proposed [Fedkiw et al. 2001; Selle et al. 2005; Kim et al. 2008; Narain et al. 2008; Schechter and Bridson 2008; Pfaff et al. 2010]. However, a second, more fundamental challenge is that Eulerian methods require voxelization of the entire simulation domain. The elliptical pressure projection operator needed to solve these equations demands that all pressure values be strongly coupled with each other, leading to large computational and memory requirements for expansive scenes. Some recent work has attempted to address this issue, with level-of-detail representations [Losasso et al. 2004], coarse grid projections [Lentine et al. 2010], and model reduction [Treuille et al. 2006; Wicke et al. 2009], but these are often incompatible with techniques for reducing dissipation.

An alternative approach is that of smoothed particle hydrodynamics, which models the fluid volume as a system of particles with pairwise forces between them. This approach has been employed for interactive simulation of liquids [Müller et al. 2003]. Similar to Eulerian grids, these methods can lead to a large number of particle primitives to sample fluid extent. In addition, enforcing incompressibility correctly can be expensive, owing to irregular computational elements. These concerns have been addressed partly with adaptive sampling [Adams et al. 2007] and predictive-corrective schemes for incompressiblility projection [Solenthaler and Pajarola 2009]. Sin et al. [2009] proposed a point-based approach that combines features from particle-based and grid-based methods. However, for the large scales under consideration, computational costs remain substantial for all such techniques.

2.2 Vorticity-based methods

Vortex simulations are a class of methods which were originally devised for aircraft wing design, and have recently begun to receive attention in computer graphics as well. These methods model the evolution of fluid *vorticity*, which is the curl of the velocity field, instead of velocity itself. With the exception of Elcott et al. [2007]'s Eulerian approach representing vorticity on a tetrahedral mesh, most of the methods in this category are Lagrangian, using singularities with the Green's function of the Laplace operator.

In this formulation, singularity methods can be applied, representing the vorticity distribution as a superposition of singularities such as particles [Chorin 1973; Park and Kim 2005], curves/filaments [Angelidis and Neyret 2005; Weißmann and Pinkall 2009], or surfaces/sheets in 3D space. Vortex singularities are excellent basis functions for fluid velocity for a number of reasons. They offer a compact and exact representation of fluid velocity in unbounded domains, automatically ensure incompressibility, and are immune to numerical dissipation. Due to their Lagrangian nature, vortex singularity methods also allow easier user control of the simulation as compared to grid methods. These features have made these methods popular for interactive simulations [Angelidis et al. 2006; Weißmann and Pinkall 2010]. Curve or filament representations also ensure in-

Figure 2: *Several seagulls flying through clouds of smoke to demonstrate airflow around their wings. On this scenes, our simulation achieved speedups of >1,000x*

compressibility of vorticity, but can only model inviscid fluids. This is not the case with vortex particles, which allow viscous fluids to be modeled, at the cost of a slightly compressible vorticity field. The divergence-free constraint can be enforced iteratively through particle strength exchange methods [Cottet and Koumoutsakos 1998], which can also model viscosity.

Enforcing boundary conditions requires the solution of a dense linear system. Though capable of arbitrary accuracy — conditions are enforced at mesh resolution — for nonrigid obstacles, precomputation of the linear system is impossible, which makes these the limiting factor in vortex simulations. Also, due to the singular nature of vortex singularities, proximity of elements has a major impact on the conditioning of this linear system, to the extent of rendering the system unsolvable due to poor conditioning. Large variations in the size of elements have a similar effect on conditioning. These issues make efficient and robust modeling of nonrigid obstacles non-trivial. It is also difficult to model the dynamics of free surfaces in this framework.

While there have been a number of hybrid simulation techniques combining, for example, rectilinear grids with tetrahedral meshes [Feldman et al. 2005], or grids with particle-based methods [Losasso et al. 2008], all these work solely with velocities, and we are aware of no work in computer graphics that allows combining velocity- and vorticity-based methods in the same simulation.

The remainder of the paper is organized as follows. The hybrid domain decomposition algorithm is described in Section 3. We propose some tools for improving the efficiency of vortex simulations in Section 4. Finally, results and analysis of our implementation are described in Section 5.

3 Hybrid Fluid Simulation

Given velocity-based and vorticity-based methods have complementary advantages, we propose a hybrid approach that combines the respective strengths of both techniques. In particular, many different techniques have been proposed to support different types of boundary conditions for velocity-based methods, including free surfaces, deformable and thin objects, and two-way coupling. On the other hand, vorticity-based methods can compactly represent effectively infinite volumes of fluid where detail in the fluid motion is spatially limited. Therefore, we propose to combine both methods through a domain decomposition approach (Section 3.2), representing the fluid using velocity-based methods near boundaries such as obstacles and free surfaces, and employing vorticity-based methods in the large interior region of the fluid.

However, the disparate nature of velocity and vorticity methods makes it challenging to combine them into a single heterogeneous simulation. An essential problem is that of coupling together the two representations at the interface between them, so that they represent a consistent velocity field for the entire fluid. We present a novel two-step coupling algorithm to address this problem: first matching normal velocities at the interface using an alternating scheme (Section 3.3), and then transferring vorticity information across subdomains through particle seeding (Section 3.4). As we show below, matching both velocity and vorticity across the interface is necessary to obtain a consistent and convergent simulation in this framework.

3.1 Subdomains

Our simulation consists of Eulerian ($\mathbf{G}_i$) and Vortex ($\mathbf{V}_i$) subdomains. For notational simplicity, we use $\mathbf{G}$ and $\mathbf{V}$ to refer to the union of all Eulerian subdomains and Vortex subdomains respectively. Eulerian subdomains model the Navier Stokes equations using uniform grids, with velocity $\mathbf{u}$ sampled on a staggered grid. Operator splitting is used to integrate each term one by one, with a BFECC Semi-Lagrangian scheme for advection, explicit integration for external forces, and a sparse Poisson solve for enforcing incompressibility using pressure. Advection and Incompressibility solve steps take volume flux boundary conditions from the hybrid simulator, instead of zero flux enforced in traditional solvers. For more details we refer the reader to [Bridson and Müller-Fischer 2007; Carlson 2004].

Vortex subdomains ($\mathbf{V}_i$) model the evolution of vorticity $\boldsymbol{\omega} = \nabla \times \mathbf{u}$, using the vorticity form of the Navier Stokes equations:

$$\frac{\partial \boldsymbol{\omega}}{\partial t} + (\mathbf{u} \cdot \nabla)\boldsymbol{\omega} = (\boldsymbol{\omega} \cdot \nabla)\mathbf{u} + \nu \nabla^2 \boldsymbol{\omega}, \tag{1}$$

$$\nabla \cdot \boldsymbol{\omega} = 0, \tag{2}$$

under the assumption of no density gradients. Under this formulation, velocity can be expressed using the Green's function of the Laplace operator, giving rise to the Biot-Savart formula:

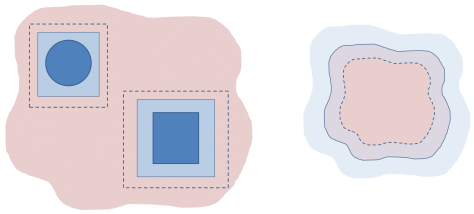
$$\mathbf{u}(\mathbf{x}) = \frac{1}{4\pi} \int_{R^3} \boldsymbol{\omega}(\mathbf{z}) \times \frac{\mathbf{x} - \mathbf{z}}{\|\mathbf{x} - \mathbf{z}\|^3} d\mathbf{z}. \tag{3}$$

for a vorticity distribution in an infinite domain. Note the absence of a pressure term, as the velocity so defined is incompressible by definition. As mentioned before, this distribution can be represented as a superposition of discrete primitives such as points (particles), curves (filaments), or surfaces/meshes (sheets) giving rise three types of vortex singularity methods. We use a particle representation owing to its ease of use, and the possibility of modeling viscosity, which is not possible using filaments. Also, due to poor long term stability of ideal singularities, regularized singularities or "vortex blobs" are typically preferred. A popular choice is the Rosenhead-Moore kernel, a regularized form of the Biot-Savart kernel with a constant smoothing radius a to give

$$\mathbf{u}(\mathbf{x}) = \frac{1}{4\pi} \int_{R^3} \boldsymbol{\omega}(\mathbf{z}) \times \frac{\mathbf{x} - \mathbf{z}}{(\|\mathbf{x} - \mathbf{z}\|^2 + a^2)^{3/2}} d\mathbf{z}. \tag{4}$$

The smoothing radius governs the concentration of vorticity represented by a vortex blob, which affects the scale of vorticity features that can be represented by it.

The vortex particle algorithm proceeds by advecting vortex particles, perturbing particle strengths to model vortex stretching and viscosity, and creating vortex sheets to model obstacles. Particles whose strength falls below a minimum threshold are culled. For more details about advection, stretching and viscosity, we refer the reader to [Cottet and Koumoutsakos 1998]. Obstacles are modeled by



(a) (b)

Figure 3: *Decomposition of domain into vortex (red) and Eulerian grid (blue) subdomains for (a) single-phase flows, and (b) free-surface flows. Dotted lines denote grid region boundaries, while solid lines denote the vortex coupling sheet*

creating vortex sheets on their surface. [Weißmann and Pinkall 2010] propose creating filaments along the edges of a polygonal mesh for this purpose, where the strength (Γ_i) of each filament (f_i) is determined to enforce zero flux through each face of the mesh, i.e. $\sum_i \Gamma_i flux_i(f_j) + flux_{\mathbf{V}}(f_j) = 0$. We utilize a similar algorithm relying on vortex particles instead of filaments to enforce flux. In the hybrid case, we create sheets to match a non-zero flux, resulting in the equation for the strength of each filament: $\sum_i \Gamma_i flux_i(f_j) + flux_{\mathbf{V}}(f_j) = flux_{\mathbf{G}}(f_j)$. In case of multiple vortex domains, either all vortex sheets can be computed using one solve, or by iteratively solving for the strength of each vortex sheet separately. This choice usually depends on whether linear systems for each component can be precomputed or not.

3.2 Hybrid Domain Decomposition

In our hybrid approach, we divide the simulation domain into a number of non-degenerate, overlapping subdomains, each of which is simulated using either the vortex method or Eulerian Navier-Stokes simulation.

It is natural to define one large region $\mathbf{V}$, consisting of the interior of the fluid at least a distance d away from boundaries, on which the vortex method is applied. This region consists of one or more disjoint subdomains $\mathbf{V}_i$. The other subdomains, labeled $\mathbf{G}_i$, use Eulerian Navier-Stokes simulations, and may contain boundaries such as static or moving obstacles and free surfaces. We assume that all Eulerian subdomains $\mathbf{G}_i$ are disjoint from each other (if not, we may merge any grids that overlap), and each of them overlaps with the vortex subdomain $\mathbf{V}$, in order to apply our coupling algorithm. Thus, the burden of supporting various boundary conditions is lifted from the vortex method, and placed on grid-based methods for which numerous techniques are available.

This decomposition is illustrated in Figure 3. For single-phase simulations like smoke, the domain consists of grids immersed in a vortex simulation, with the vortex domain extending to a user-specified distance d inside every grid. For free-surface simulations like water, the vortex simulation is embedded inside an Eulerian grid, with the boundary of the vortex domain being a distance d inside the fluid surface. As boundaries move, so do their corresponding subdomains, so that at all times the boundaries are contained entirely within grids and remain at least a distance d from the extent of the vortex subdomain.

Fluid velocities in the vortex subdomain are determined using the grid velocities as boundary conditions. That is, the boundary of the vortex subdomain, which is a surface that lies entirely within the grid, is treated as an obstacle whose normal velocities are given by the grid velocity field, and a vortex sheet is computed on this surface to match the corresponding fluxes. Thus, the velocity field can be evaluated at any point in the interior using the Rosenhead-Moore kernel (4).

We define the timestepping scheme of the hybrid simulation as follows. Each subdomain is advanced independently over one time step Δt while assuming that velocities in the others are constant. Assuming constant velocities in other subdomains introduces an error on the order of $O(\Delta t)$ in both vortex and grid regions: in the vortex region because boundary conditions are determined by the grid; and in the grid because advection carries velocities in from the vortex region. In general, this magnitude of error is acceptable because the rest of the simulation (like most techniques for fluid simulation in graphics) is also first-order accurate. At the end of the time step, we couple the simulations back together by matching normal velocities at the boundaries, thus enforcing incompressibility, and matching vorticity in the interior of the overlap region. It can be shown that when the overlap region is simply connected, if both grid and vortex velocity fields have equal flux at the boundary and equal vorticity in the interior, they must be identical [Cantarella et al. 2002]. For overlap regions with complex topologies, additional circulation constraints are needed. Thus, we can ensure that the combined simulation is consistent: where the grid and the vortex domains overlap, they agree on the fluid velocity.

In the following two subsections, we describe our proposed coupling techniques in more detail.

3.3 Velocity coupling

Each Eulerian solver stores a velocity field which determines the velocity $\mathbf{u}_{\mathbf{G}_i}$ at any point within its grid $\mathbf{G}_i$. Similarly, the vortex method determines the velocity $\mathbf{u}_{\mathbf{V}}$ at any point in its subdomain $\mathbf{V}$. By allowing domains $\mathbf{G}_i$ and $\mathbf{V}$ to overlap, we can enforce consistency between the corresponding simulations by ensuring that $\mathbf{u}_{\mathbf{G}_i}$ and $\mathbf{u}_{\mathbf{V}}$ are equal in the overlap region $\mathbf{O}_i = \mathbf{G}_i \cap \mathbf{V}$.

To enforce this constraint, we recall that the velocity field in any region is uniquely determined by the distribution of vorticity within it, and the flux at the boundaries of the region. Therefore, discrepancies between the velocities seen by different subdomains can only occur from having incorrect vorticity information in the interior or incorrect flux at the boundary. Assuming consistent vorticity in the overlap region, we need to ensure that flux across the boundary of the overlap region is consistent, i.e. velocity induced by both simulations is the same $\mathbf{u}_{\mathbf{G}_i} = \mathbf{u}_{\mathbf{V}}$ at the boundary .

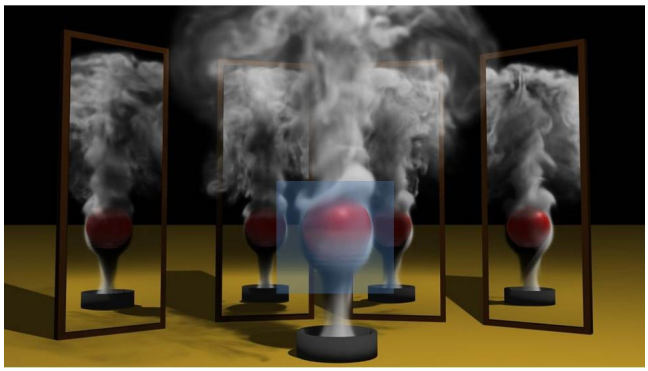
Velocity can be matched at $\partial\mathbf{G}_i$ by enforcing the desired velocity as boundary conditions for the incompressibility projection step. To do the same with vortex simulation requires the creation of a vortex sheet $\mathbf{S}$ to match normal flux through $\partial\mathbf{V}$, i.e.

$$(\mathbf{u}_{\mathbf{S}} + \mathbf{u}_{\mathbf{V}}) \cdot \mathbf{n} = \mathbf{u}_{\mathbf{G}_i} \cdot \mathbf{n} \tag{5}$$

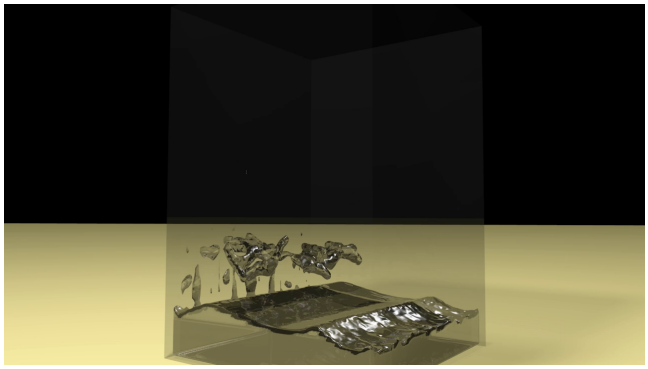
on the sheet, where $\mathbf{u}_{\mathbf{S}}$ is the velocity induced by the vortex sheet $\mathbf{S}$.

Thus the velocity coupling can be formulated as a fixed point iteration, one iteration of which can be expressed as follows:

1. Determine the velocity $u_{\mathbf{V}} + u_{\mathbf{S}}$ at the boundary $\partial\mathbf{G}_i$ of the Eulerian subdomain
2. Using $u_{\mathbf{V}} + u_{\mathbf{S}}$ as the boundary condition, perform the incompressibility projection on the grid $\mathbf{G}_i$
3. Determine the velocity $u_{\mathbf{G}_i}$ at the boundary $\partial\mathbf{V}$ of the subdomain $\mathbf{V}$
4. Compute strength of the vortex sheet $\mathbf{S}$ to match vortex velocity $u_{\mathbf{V}}$ to $u_{\mathbf{G}_i}$



(a)



(b)

Figure 4: *Examples of fluid flow and domain decomposition (a) static, for single-phase flows, (b) dynamic, for free-surface flows changing with the topology of water extent*

Coupling iterations are performed till $u_{\mathbf{G}}$ and $u_{\mathbf{V}} + u_{\mathbf{S}}$ converge. For coupling multiple Eulerian subdomains with the vortex subdomain, this iteration can be performed in lockstep for every pair of overlapping subdomains.

This algorithm belongs to the class of methods known as the Schwarz alternating methods [Toselli and Widlund 2004], which are commonly used in domain decomposition methods. Schwarz alternating methods are guaranteed to converge to a unique solution for second order PDEs, and thus this iteration ensures that the velocity field is consistent at the boundaries of all subdomains, and consequently in the entire domain.

3.4 Vorticity exchange

Velocity coupling on the subdomain boundaries will yield consistent velocities over the entire overlap region only if the vorticities seen by both representations are equal. However, even if the vorticities are equal in the initial conditions, they will gradually go out of sync over time, as vorticity is transported into the overlap region through advection. Therefore, to ensure consistency at all times, we need to account for this by exchanging vorticity between the grid and the particle domains. We derive this procedure by considering the two cases where vorticity is brought into the overlap region from the vortex particles and from the grid domain respectively.

From the vortex particle domain, vorticity enters the overlap region when a vortex particle flows in and crosses the boundary of the grid. In this case, transferring vorticity into the grid's velocity field can be done with appropriate boundary conditions. When we perform velocity advection on the grid using, say, a semi-Lagrangian scheme, it typically requires velocity information at locations outside the grid; we fill this in using the velocities determined by the vortex particle representation. Thus, as a vortex particle enters the overlap region, advection on the grid automatically pulls in its corresponding vorticity. Further, when the particle moves into the grid-only region, it may be deleted as its vorticity remains represented on the grid, or preserved to drive vorticity confinement.

Handling the transfer from the grid to vortex particles is somewhat more involved. As advection on the grid moves velocities around, the vorticity present in the grid-only region may be transported into the overlap region. This vorticity is unaccounted for by existing vortex particles in the overlap region, creating error in the representation. Therefore, we must insert new vortex particles in the overlap region to make the vorticities match.

We do this in a greedy fashion, at each iteration inserting the particle which best reduces the difference between the vorticity due to the particles and the vorticity present in the grid, denoted $\Delta\omega$. This is the particle at position x_p with strength α_p which minimizes the L_2-norm of the vorticity difference,

$$\varepsilon = \int_{G_i \cap V} \big(\Delta\omega_p(x) + k(x - x_p, \alpha_p)\big)^2 dV, \tag{6}$$

where k is the Rosenhead-Moore kernel. We found that the smoothing kernel used to obtain the Rosenhead-Moore kernel from the Biot-Savart kernel is well approximated by a Gaussian of standard deviation $a/2$. The choice of Gaussian smoothing is motivated by its smoothing properties in scale space, due to which features smaller than a are smoothed away, and computational efficiency afforded due to its linear separability property. Therefore, we smooth $\Delta\omega$ with the Gaussian and choose x_p at which the smoothed field attains its maximum magnitude in the overlap region. Once x_p is fixed, it is straightforward to find the particle strength α_p which minimizes ε, as k is linear in α_p. We add this particle into the simulation and then repeat the process, until ε or $\|\alpha_p\|$ fall below chosen thresholds.

With this process, we maintain consistency between the grid and the vortex particles in the overlap region. To reduce the particle count, we also merge particles which are within a certain fraction of the smoothing radius a of each other. At every time step, we consider the O(dn^2) cells in the overlap region. Though in the worst case, all these cells may result in new vortex particles, temporal coherency results in the creation of O(dn) particles per step. We note that even though the Rosenhead-Moore kernel has infinite support, vortex particles can be created with finite information offered by the grid, since the distribution of vorticity around a particle decays rapidly with distance.

The outline of our resulting algorithm is shown in Figure 5.

4 Efficient vortex particle simulation

Vortex particle simulation is one of the major underlying components of our method. In this section, we discuss our implementation, including optimizations — some derived from theoretical concerns, others from practical ones. Efficient algorithms for purely Eulerian fluid simulation already exist, thus we do not delve into them here. As discussed in Section 2, the three main steps of vortex particle simulation are advection, stretching, and obstacle handling. Advection requires the computation of velocity at every particle position, and naïve summation using the Biot-Savart law leads to an O(p^2) algorithm for p particles. [Lindsay and Krasny 2001] propose a hierarchical summation approach to compute velocities, that reduces the cost of this step to O($p \log p$). However, in the presence of obstacles, the most expensive step is evaluating flux through the obstacle

For each time step:
1. Advance level-set surfaces, if any
2. Advect velocity fields $u_{\mathbf{G}_i}$ and apply any external forces
3. Convect vortex particles in $\mathbf{V}$
4. Repeat until convergence or maximum iterations:
   (a) Perform incompressibility projection on all grids $\mathbf{G}_i$ using boundary conditions $u_{\mathbf{V}} + u_{\mathbf{S}}$ from vortex particle simulation
   (b) Rebuild vortex sheet(s) $\mathbf{S}$ in the middle of grid-vortex overlap
   (c) Determine strength of vortex sheet(s) $\mathbf{S}$ to match vortex and grid velocities as per Equation (5)
5. Vorticity exchange
   (a) Create vortex particles to minimize vorticity residual
   (b) Perform vortex stretching to model fluid viscosity

Figure 5: *The main steps of our method.*

surface, and the subsequent linear system solve. In addition, mesh quality plays a pivotal role in the conditioning of this system.

4.1 Well-conditioned vortex sheet computation

The computation of vortex sheets that match normal velocities at their surface is a major step in vortex singularity simulation and our velocity coupling algorithm. Like any other vortex singularity element, the sheet induces a velocity field determined by the Biot-Savart law. In practice, these sheets are discretized as polygonal meshes, where each mesh face is a singularity with a distribution of vorticity. We use the approach proposed by [Weißmann and Pinkall 2010], which assumes a constant strength for each face and represents each with a filament geometry defined by the edges of the face. To determine the strength of each face, a linear system is constructed that matches normal velocities or total flux at the face in accordance with equation (5). Due to reciprocity between filaments, the resulting matrix is symmetric. We first evaluate the net flux on each boundary polygon due to all vortex particles in the scene, giving $\mathbf{u_V} \cdot \mathbf{n}$, and then solve a linear system to compute vortex sheet strengths that will produce the desired change in flux, $(\mathbf{u}_{\mathbf{G}_i} - \mathbf{u_S}) \cdot \mathbf{n}$.

In the presence of topological holes, fluxes are insufficient to uniquely determine the velocity field, and a circulation constraint must be specified for each hole. For example, if the sheet is a torus, normal fluxes alone cannot capture a purely tangential flow through the torus. For a formal discussion, we refer the reader to [Cantarella et al. 2002]. The condition number of this problem depends heavily on the quality of the underlying mesh. When matching flux through each face, a high variance in face areas can result in a poorly conditioned system. Similarly, due to the singular nature of these vortex elements, the minimum separation between faces also affects conditioning. Very low separation among a few faces can skew the eigenvalues of the matrix, making it poorly conditioned to the extent of losing rank.

To remedy this, we propose constructing vortex sheets using grid faces. Since we construct sheets at a distance d from any surface, this is equivalent to measuring distance using the infinity norm. The resulting mesh has two important properties, firstly that faces have zero variance of area, and that minimum separation between two faces is at least the grid cell width Δx. Using these meshes results in linear systems with much better conditioning than those obtained using the traditional marching cubes algorithm, which relies on the 2-norm. In addition, the entries of any possible matrix constructed can be precomputed, since the set of all possible mesh faces is a finite set.

4.2 Hierarchical methods for flux computation

As noted earlier, flux computation is one of the major computational kernels of our algorithm. Since the velocity induced by a vortex element is the curl of its corresponding vector potential $\mathbf{\Phi}$, Stokes' theorem lets us express its flux through a polygon η as

$$flux(\boldsymbol{\eta}) = \iint_{\eta} \mathbf{u}(x) \cdot d\mathbf{A} = \sum_i \int_{\partial\eta_i} \mathbf{\Phi}(x) \cdot d\mathbf{l} \qquad (7)$$

where

$$\Phi_p(x) = \frac{1}{4\pi} \int_{R^3} \frac{\boldsymbol{\omega}(\mathbf{z})}{\sqrt{\|\mathbf{x}-\mathbf{z}\|^2 + a^2}}. \qquad (8)$$

To create the linear system we need to determine flux induced by a vortex particle or filament though a polygon. Though closed forms can be obtained for both, a better approach is to extend the hierarchical structure created to compute velocities. This structure can be easily extended to compute the vector potential induced by a set of vortex particles at any point in space. Then, a quadrature rule can be used to evaluate the integral in equation (7). We found that a 5th order Gauss-Legendre quadrature achieves a relative error < 0.01 when compared to the closed form.

Creating a hierarchical approach for filaments is non-trivial. However, using the same idea of evaluating integrals with Gaussian quadrature, a filament edge can be discretized into a number of vortex particles placed according to the quadrature sample positions, with their strengths being the filament strength multiplied by quadrature weights, strength vectors being aligned to the edge. This is especially simple for the mesh we create, since edge lengths are equal, but can be applied to any polygonal mesh.

Thus the same hierarchical approach used for vortex particles can be used for vortex sheet velocity and flux computation, creating a unified representation for the vortex simulation. This suggests an iterative approach to solving the linear system to determine vortex sheet strength, since for a mesh containing m faces, instead of using O(m^2) operations for matrix vector multiplication, we can do the same computation hierarchically in O($m \log m$) operations. This is especially helpful since for a grid of size O($n \times n \times n$), m can be O(n^2), thus a matrix-vector multiply of complexity O(n^4) can be performed with O($n^2 \log n$) operations. By using Fast Multipole Methods, this cost can be brought down further to O(n^2).

As to the choice of iterative methods, we use the GMRES algorithm with restarts [Barrett et al. 1994], since the matrix is indefinite. We observe convergence in O(n) iterations on average, O(n^2) iterations in the worst case. Thus, the strength of a vortex sheet can be determined in O($n^4 \log n$) time in the worst case, and O($n^3 \log n$) on average.

It is important to note that the underlying bounding volume hierarchy constructed for this method is also used to accelerate neighborhood queries needed for the vortex stretching step, bringing its complexity down from O(p^2) to O(kp) where k is the number of neighbors considered.

5 Results

Our method was used to simulate a number of challenging examples. Vorticity confinement was not used in any scenario.

Coupled simulation in unbounded space: Seagulls in flight (Figure 2) are used to demonstrate how our method can be used for a scenario which would be challenging to simulate with either vortex or velocity methods alone. In this scene, three seagulls fly through

Example	dt	Grid Resolution (Eulerian)	Simulation Time Eulerian (s)	Grid Resolution (Hybrid)	Simulation Time Hybrid (s)	Max. Vortex Particles	Average coupling sheet size	Speedup
Dam Break	0.0167	128x192x128	93.375	-	83.122	15860	490x490	1.123
		256x384x256	1313.94	-	514.41	32089	2316x2316	2.55
Smoke around	0.04	72x118x72	3.66	32x34x32	1.033	11516	1232x1232	3.54
sphere	0.04	144x236x144	47.712	64x68x64	5.959	23976	1232x1232	8.006
	0.04	144x236x144	47.712	64x68x64	29.7295	20105	4928x4928	1.604
	0.04	288x474x288	1277.37	128x136x128	177.044	34298	5640x5640	7.214
Wave	0.04	600x100x120	403.89	-	118.897	1567	2742x2742	3.4
City	0.04	312x60x210	87.51	-	66.801	7208	1722x1722	1.31
Seagull	0.04	520x256x520	NA	84x60x84	15.31	25674	1296x1296	>1000

Table 1: *Single thread performance for our examples (All time values for one simulation step)*

four plumes of laminar smoke flow. To best demonstrate the effect of obstacle interaction, buoyancy is not modeled — the flow is kept laminar and does not become turbulent on its own. Vortex methods cannot be coupled robustly with deformable objects, while a domain of this size cannot be voxelized in desktop or mobile PC memory. The deforming wings of the seagulls induce vorticial details into the flow, which is carried by vortex particles even after the birds have moved on to different parts of the scene. Although we have applied a bound to the domain by determining the extents of all elements of the scene, the domain is unbounded in principle; the memory savings noted are lower bounds of what can be obtained with even larger scenes.

Preservation of vortex features: Figures 4(a) and 4(b) show fluid flow around simple boundaries where vortex features are preserved. It is important to note that speedup obtained for the sphere scene is not as significant as the seagull scene, since it has high-detail vortices in nearly its entire domain of interest, and thus cannot benefit as greatly from a compact representation of vorticity.

Tsunami striking a city (Figure 1(a)). A tsunami breaks over a city block, demonstrating fluid detail created by a high number of rigid objects; this scenario involves interaction with a large area of obstacle surfaces. The complexity of our approach scales with the total surface area of all elements in the scene — this scene is not expected to enhance performance as well as other cases. However, our approach helps in preserving flow details even in the absence of vorticity confinement.

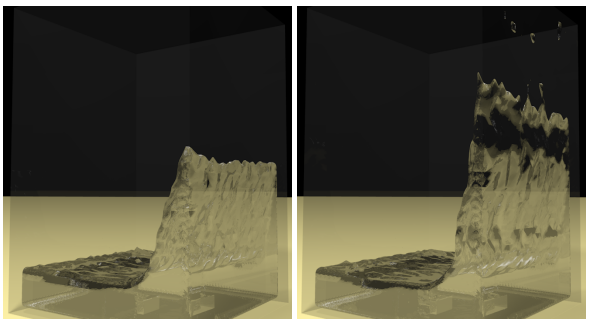
Velocity coupling/comparison: In this scene, we show wave forming, cresting, and breaking. Visual fidelity and detail are maintained, resulting in a qualitatively similar simulation as previous methods, while affording performance benefits.

Scenes are modeled and rendered using Blender® and Maya®. Water is rendered as a mesh with an appropriate water shader, while smoke is rendered as a density field. Smoke is advected passively through the flow, but can also be used to create density estimates for buoyancy forces.

5.1 Performance

For each of these examples, the vortex smoothing radius a is chosen to be close to the grid cell width; in all the examples shown here, this was in the range $[\Delta x, 4\Delta x]$. Correspondingly, the size of the overlap region d is kept between $[a, 2a]$ to allow sufficient information to create vortex particles, since the decay of the vorticity kernel is faster than linear. In practice, we have observed that an average of 2-3 coupling iterations are sufficient to allow the simulation regions to converge with an error of 10^{-4} in the L2 norm of the residual.

We measured the performance of our method on an Intel® Xeon® X5560 processor running at 2.8GHz on a system with 48GB of RAM and 8MB lowest-level cache. Our method is implemented in C/C++, and some components use SIMD (4-wide single-precision SSE) instructions to improve performance and resource utilization.



(a) (b)

Figure 6: *Dam Break example at t=0.48s using PIC(left) and FLIP(right) showing maximum height reached by water*

The timing and performance results for all of our examples are shown in Table 1; here we show a performance comparison between purely Eulerian and hybrid simulation for single phase and free-surface simulations demonstrating the speedups obtained by using our method. Overall our approach gives varying speedups that are most pronounced for single-phase simulations. This is expected; free-surface scenes require regeneration of the coupling vortex sheet every step, which represents additional overhead in our technique not present in standard fluid solvers and proportionally mitigates (but does not eliminate) the overall advantage of our technique in efficiency.

Additional scope for optimization of these results exists, by using appropriate preconditioners for GMRES, and more optimized solvers. In addition, for free-surfaces, the same coupling sheet can potentially be used for multiple frames and linear solvers can be warm started with values from the previous time steps leading to more performance benefits.

5.2 Controlling Dissipation

As some of our examples may appear diffusive, we offer some insight into controlling diffusion. First of all, it is important to note that smoke sources in our scenes introduce purely laminar flow, with no model for buoyancy. This is done in order to highlight the preservation of vorticity across subdomain boundaries, since such an observation would be difficult in turbulent flow. Because our coupling algorithm matches flux and vorticity, any error in the latter would be observed as spurious vorticity at subdomain boundaries.

Three main sources of possible diffusion while using our algorithm are the choice of smoothing radius a, grid advection, and vortex stretching algorithms chosen. In figure 6 we show the difference in using PIC v.s. FLIP advection algorithms [Zhu and Bridson 2005].

The choice of advection algorithms has an equally large impact in single-phase simulations, and some of our examples exhibit such dissipation as well. However, as shown by the example, this can be readily addressed by the use of less dissipative advection algorithms. The impact of advection algorithms on grid vorticity is analogous to the effect of vorticity stretching in vortex domains along with the choice of smoothing kernel. Accurate and stable stretching of vortex particles requires careful creation of new particles as needed, and the enforcement of the divergence-free vorticity constraint. Deviation from the constraint results in a tradeoff between numerical stability and diffusive behavior. Though a higher order kernel can improve accuracy, third order and higher kernels induce negative vorticity which is visually undesirable. The choice of smoothing radius is also important since it controls the highest frequency of vorticity detail that can be modeled.

6 Conclusion

We have presented a hybrid simulation algorithm for simulating fluids in large-scale scenes with a reduced memory and computational footprint that is proportional to the total area of all surfaces. It provides memory and execution time improvements from 2x–1000x, depending on the scene. This performance gain is achieved via a novel algorithm that couples vortex singularity methods and Eulerian velocity simulations. We also propose a vortex particle creation algorithm, which creates particles to compactly represent a velocity field by minimizing the L_2 norm of the difference. Our generalized approach also offers a flexibility to choose different regimes and numerical methods for distinct regions of a scene.

6.1 Limitations and future work

Our method discretizes the vorticity space – rather than spatial extent – using one set of basis functions, defined by the smoothing radius a. To achieve high performance, our method relies on sparsity in this space. For scenes that contain dense vortex detail, our method's computational advantage diminishes, bringing its performance closer to other techniques. For specific static scenes, more compact bases could be derived, but such an approach would not scale to dynamic scenes without substantial pre-computation.

We also note that our particle seeding algorithm chooses a subset of possible vortex bases: those centered at grid cell centers. Though this does not reduce the applicability of the method, the possibility of expanding the set of bases to different smoothing radii and generalized particle placement would add to the efficiency and compactness of the vortex representation.

Our formulation uses overlapping subdomains for coupling since it simplifies the coupling algorithm. A non-overlapping coupling, if possible, could allow the formulation of a vortex boundary condition for free-surfaces and two-way coupled obstacles. This would reduce the need for a full-fledged grid simulation, and afford greater flexibility in the choice for simulation methods.

Representing velocity fields as vortex particles opens the possibility of compact storage and manipulation of velocity fields. This can be used for artistic control of fluid velocity and even for accurate modeling of fluid turbulence. Further investigation of these avenues would be valuable for increasing fidelity and artistic control of existing algorithms as well.

Acknowledgments: We would like to thank the anonymous reviewers for their valuable suggestions; Adam Lake and Nico Galoppo at Intel for their support during the early stage of concept exploration. This research was supported in part by ARO Contract W911NF-04-1-0088, NSF awards 0917040, 0904990, 100057 and 1117127, and Intel. The second author was supported by UNC Computer Science Alumni Fellowship while working on this project at UNC.

References

ADAMS, B., PAULY, M., KEISER, R., AND GUIBAS, L. J. 2007. Adaptively sampled particle fluids. *ACM Trans. Graph. 26* (July).

ANGELIDIS, A., AND NEYRET, F. 2005. Simulation of smoke based on vortex filament primitives. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, ACM, SCA '05, 87–96.

ANGELIDIS, A., NEYRET, F., SINGH, K., AND NOWROUZEZAHRAI, D. 2006. A controllable, fast and stable basis for vortex based smoke simulation. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, Eurographics Association, SCA '06, 25–32.

BARRETT, R., BERRY, M., CHAN, T. F., DEMMEL, J., DONATO, J., DONGARRA, J., EIJKHOUT, V., POZO, R., ROMINE, C., AND DER VORST, H. V. 1994. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd Edition*. SIAM, Philadelphia, PA.

BATTY, C., BERTAILS, F., AND BRIDSON, R. 2007. A fast variational framework for accurate solid-fluid coupling. *ACM Trans. Graph. 26*, 3, 100.

BRIDSON, R., AND MÜLLER-FISCHER, M. 2007. Fluid simulation: Siggraph 2007 course notes. In *ACM SIGGRAPH 2007 courses*, ACM, SIGGRAPH '07, 1–81.

CANTARELLA, J., DETURCK, D., AND GLUCK, H. 2002. Vector calculus and the topology of domains in 3-space. *Amer. Math. Monthly 109*, 5, 409–442.

CARLSON, M. T., 2004. Rigid, melting, and flowing fluid. Ph.D. Dissertation.

CHENTANEZ, N., GOKTEKIN, T. G., FELDMAN, B. E., AND O'BRIEN, J. F. 2006. Simultaneous coupling of fluids and deformable bodies. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 83–89.

CHENTANEZ, N., FELDMAN, B. E., LABELLE, F., O'BRIEN, J. F., AND SHEWCHUK, J. R. 2007. Liquid simulation on lattice-based tetrahedral meshes. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, Eurographics Association, SCA '07, 219–228.

CHORIN, A. J. 1973. Numerical study of slightly viscous flow. *Journal of Fluid Mechanics 57*, 04, 785–796.

COTTET, G. H., AND KOUMOUTSAKOS, P. D. 1998. *Vortex Methods: Theory and Practice*. Cambridge University Press.

ELCOTT, S., TONG, Y., KANSO, E., SCHRÖDER, P., AND DESBRUN, M. 2007. Stable, circulation-preserving, simplicial fluids. *ACM Trans. Graph. 26* (January).

FEDKIW, R., STAM, J., AND JENSEN, H. W. 2001. Visual simulation of smoke. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, ACM, SIGGRAPH '01, 15–22.

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

FOSTER, N., AND FEDKIW, R. 2001. Practical animation of liquids. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, ACM, SIGGRAPH '01, 23–30.

FOSTER, N., AND METAXAS, D. 1996. Realistic animation of liquids. *Graph. Models Image Process. 58* (September).

KIM, B., LIU, Y., LLAMAS, I., AND ROSSIGNAC, J. 2007. Advections with significantly reduced dissipation and diffusion. *IEEE Transactions on Visualization and Computer Graphics 13*, 135–144.

KIM, T., THÜREY, N., JAMES, D., AND GROSS, M. 2008. Wavelet turbulence for fluid simulation. *ACM Trans. Graph. 27* (August), 50:1–50:6.

KLINGNER, B. M., FELDMAN, B. E., CHENTANEZ, N., AND O'BRIEN, J. F. 2006. Fluid animation with dynamic meshes. In *Proceedings of ACM SIGGRAPH 2006*, 820–825.

LENTINE, M., ZHENG, W., AND FEDKIW, R. 2010. A novel algorithm for incompressible flow using only a coarse grid projection. In *ACM SIGGRAPH 2010 papers*, ACM, SIGGRAPH '10, 114:1–114:9.

LENTINE, M., AANJANEYA, M., AND FEDKIW, R. 2011. Mass and momentum conservation for fluid simulation. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ACM, 91–100.

LINDSAY, K., AND KRASNY, R. 2001. A particle method and adaptive treecode for vortex sheet motion in three-dimensional flow. *J. Comput. Phys. 172* (September), 879–907.

LOSASSO, F., GIBOU, F., AND FEDKIW, R. 2004. Simulating water and smoke with an octree data structure. In *ACM SIGGRAPH 2004 Papers*, ACM, SIGGRAPH '04, 457–462.

LOSASSO, F., TALTON, J., KWATRA, N., AND FEDKIW, R. 2008. Two-way coupled sph and particle level set fluid simulation. *IEEE Transactions on Visualization and Computer Graphics 14* (July), 797–804.

MULLEN, P., CRANE, K., PAVLOV, D., TONG, Y., AND DESBRUN, M. 2009. Energy-preserving integrators for fluid animation. *ACM Trans. Graph. 28* (July), 38:1–38:8.

MÜLLER, M., CHARYPAR, D., AND GROSS, M. 2003. Particle-based fluid simulation for interactive applications. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, Eurographics Association, SCA '03, 154–159.

NARAIN, R., SEWALL, J., CARLSON, M., AND LIN, M. C. 2008. Fast animation of turbulence using energy transport and procedural synthesis. *ACM Trans. Graph. 27* (December), 166:1–166:8.

PARK, S. I., AND KIM, M. J. 2005. Vortex fluid for gaseous phenomena. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, ACM, SCA '05, 261–270.

PFAFF, T., THUEREY, N., COHEN, J., TARIQ, S., AND GROSS, M. 2010. Scalable fluid simulation using anisotropic turbulence particles. In *ACM SIGGRAPH Asia 2010 papers*, ACM, New York, NY, USA, SIGGRAPH ASIA '10, 174:1–174:8.

ROBINSON-MOSHER, A., SHINAR, T., GRETARSSON, J., SU, J., AND FEDKIW, R. 2008. Two-way coupling of fluids to rigid and deformable solids and shells. *ACM Trans. Graph. 27* (August), 46:1–46:9.

SCHECHTER, H., AND BRIDSON, R. 2008. Evolving sub-grid turbulence for smoke animation. In *Proceedings of the 2008 ACM/Eurographics Symposium on Computer Animation*.

SELLE, A., RASMUSSEN, N., AND FEDKIW, R. 2005. A vortex particle method for smoke, water and explosions. In *ACM SIGGRAPH 2005 Papers*, ACM, SIGGRAPH '05, 910–914.

SELLE, A., FEDKIW, R., KIM, B., LIU, Y., AND ROSSIGNAC, J. 2008. An unconditionally stable maccormack method. *J. Sci. Comput. 35* (June), 350–371.

SIN, F., BARGTEIL, A. W., AND HODGINS, J. K. 2009. A point-based method for animating incompressible flow. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*.

SOLENTHALER, B., AND PAJAROLA, R. 2009. Predictive-corrective incompressible sph. *ACM Trans. Graph. 28* (July), 40:1–40:6.

STAM, J. 1999. Stable fluids. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '99, 121–128.

TOSELLI, A., AND WIDLUND, O. 2004. *Domain Decomposition Methods - Algorithms and Theory*, vol. 34 of *Springer Series in Computational Mathematics*. Springer.

TREUILLE, A., LEWIS, A., AND POPOVIĆ, Z. 2006. Model reduction for real-time fluids. In *ACM SIGGRAPH 2006 Papers*, ACM, SIGGRAPH '06, 826–834.

WEISSMANN, S., AND PINKALL, U. 2009. Real-time interactive simulation of smoke using discrete integrable vortex filaments. In *Workshop in VRIPS 2009*, Eurographics Association.

WEISSMANN, S., AND PINKALL, U. 2010. Filament-based smoke with vortex shedding and variational reconnection. In *ACM SIGGRAPH 2010 papers*, ACM, SIGGRAPH '10, 115:1–115:12.

WENDT, J. D., BAXTER, W., OGUZ, I., AND LIN, M. C. 2007. Finite volume flow simulations on arbitrary domains. *Graph. Models 69* (January), 19–32.

WICKE, M., STANTON, M., AND TREUILLE, A. 2009. Modular bases for fluid dynamics. In *ACM SIGGRAPH 2009 papers*, ACM, SIGGRAPH '09, 39:1–39:8.

ZHU, Y., AND BRIDSON, R. 2005. Animating sand as a fluid. In *ACM SIGGRAPH 2005 Papers*, ACM, New York, NY, USA, SIGGRAPH '05, 965–972.