Eurographics/ ACM SIGGRAPH Symposium on Computer Animation (2011)
A. Bargteil and M. van de Panne (Editors)

Mass and Momentum Conservation for Fluid Simulation

Michael Lentine[†] Stanford University Industrial Light + Magic

Mridul Aanjaneya[†] Stanford University Nokia Research Center

Ronald Fedkiw[†] Stanford University Industrial Light + Magic

Abstract

Momentum conservation has long been used as a design principle for solid simulation (e.g. collisions between rigid bodies, mass-spring elastic and damping forces, etc.), yet it has not been widely used for fluid simulation. In fact, semi-Lagrangian advection does not conserve momentum, but is still regularly used as a bread and butter method for fluid simulation. In this paper, we propose a modification to the semi-Lagrangian method in order to make it fully conserve momentum. While methods of this type have been proposed earlier in the computational physics literature, they are not necessarily appropriate for coarse grids, large time steps or inviscid flows, all of which are common in graphics applications. In addition, we show that the commonly used vorticity confinement turbulence model can be modified to exactly conserve momentum as well. We provide a number of examples that illustrate the benefits of this new approach, both in conserving fluid momentum and passively advected scalars such as smoke density. In particular, we show that our new method is amenable to efficient smoke simulation with one time step per frame, whereas the traditional non-conservative semi-Lagrangian method experiences serious artifacts when run with these large time steps, especially when object interaction is considered.

1. Introduction

Momentum conservation is considered to be a fundamental building block for solid simulations (e.g. it is enforced during rigid body collisions, rigid body temporal evolution, elastic and damping forces for mass-spring systems, etc.), however, when simulating incompressible fluids in the graphics community, it has largely been ignored. The same can be said in the computational physics community, where many methods for simulating incompressible flow typically do not conserve momentum, albeit when simulating compressible flow, momentum is strictly conserved since it is required to get the correct shock speeds. We refer the reader to a few of the papers on compressible flow written in the graphics literature, such as [YOH00, SGTL09, KGF10]. In contrast, the semi-Lagrangian method, introduced in graphics by Stam [Sta99], does not conserve momentum, yet it has been a staple algorithm for incompressible flow for over a decade.

While many alternative advection methods exist in the computational physics literature, they often come with restrictions on the grid type, grid spacing, or boundary conditions. This makes them rather difficult to use in graph-

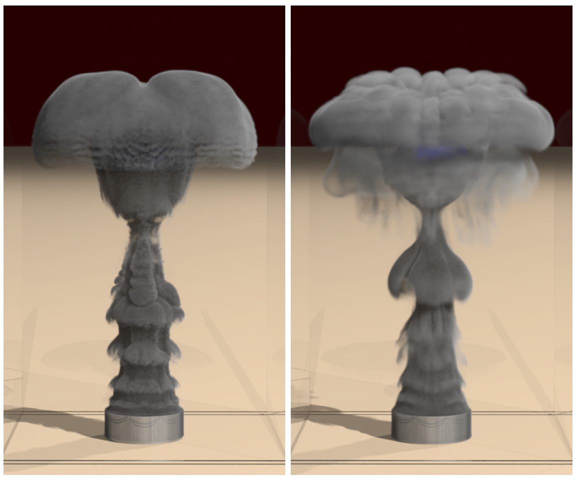

Figure 1: *A comparison between simulations of (Left) the traditional semi-Lagrangian method and (Right) our method with a very large time step at resolution $256 \times 512 \times 256$. Note how the large time steps cause alternating gaps in the smoke as seen above and below the sphere. Also note the lack of fluid structure resulting from the collision with the sphere. In contrast, our method conserves mass and momentum and produces a highly detailed flow field. Note in particular, the creation of multiple distinct vortex rings that pass through each other using our method.*

[†] e-mail: {mlentine|aanjneya|fedkiw}@cs.stanford.edu

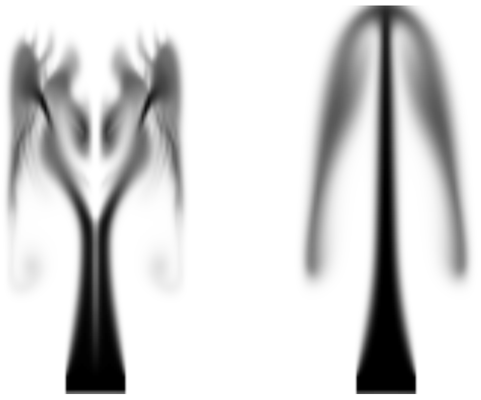

Figure 2: *(Left) using the method from [LGF11], incompressibility is not properly enforced on coarse grids with large time steps and no viscosity. Note the white line down the middle of the image where the smoke splits apart, which occurs because of a lack of incompressibility during the advection. (Right) our new method incorporates incompressibility into advection, keeping the plume from splitting apart.*

ics applications where the geometry tends to be detailed and complex. Most of these schemes also come with restrictions on the size of the time step, but semi-Lagrangian advection is unconditionally stable. In fact, this makes it appealing to build high-order methods out of first-order semi-Lagrangian building blocks, as was done in the BFECC [DL03, KLLR05] and the MacCormack [SFK*08] methods. Semi-Lagrangian methods are useful on octrees because of their hierarchical nature and varying grid cell-size [LGF04, LFO06], on tetrahedral meshes because of their lack of structure and varying element size [FOK05, FOKG05, KFCO06, CFL*07, BXH10], and for solid-fluid coupling [CMT04, CGFO06, BBB07, RMSG*08]. It is especially useful when a solid moves through the mesh causing cells to be cut in various sizes or when considering thin shells such as cloth when one does not want leaking (see [GSLF05]) across the solid surface.

Some computational physics authors have considered momentum conservation in the context of incompressible flows. However, these researchers are typically interested in obtaining algorithms that converge under refinement in both space and time, necessitating the use of fully viscous flows. In contrast, graphics applications typically use inviscid flows for efficiency (see [FSJ01]), and these are known not to converge as the grid cell size and time step size approach zero. In addition, graphics applications typically use large time steps and coarse grids. Hence, it is not clear if these algorithms in the computational physics literature would perform well for graphics applications. We have studied one such paper [LGF11] with a rather simple implementation based on the semi-Lagrangian method conducive to use in graphics applications and found that although this method converged to correct analytical solutions, it performed poorly for applications of interest to graphics. We illustrate this in Figure 2 and explain in Section 2 why it performs poorly. Furthermore, we propose a novel algorithm which aims to enforce incompressibility during advection at a discrete level, thereby alleviating the problems this scheme has for inviscid flows on coarse grids with large time steps.

The typical projection step which makes the fluid divergence free naturally conserves momentum. Combining this with our new momentum-conserving advection, one has a fully momentum-conserving incompressible flow solver (except for source terms). Note that body forces such as gravity and buoyancy should add momentum to a flow as potential energy is converted into kinetic energy. Conservative algorithms for both rigid and deformable bodies allow for momentum changes based on body forces (these forces would conserve momentum, but the other body, for example the earth, is not modelled). Many graphics applications use a turbulence model such as vorticity confinement (see [FSJ01,SRF05,YKH*09,JKB*10]) to increase the level of detail in the flow, and we propose a novel modification to this source term which makes it fully conserve momentum. We demonstrate the advantages of using momentum conservation for smoke simulation through several examples. Finally, we show that our method can be adapted to water simulations and present some promising results for energy conservation.

2. Advection

A momentum-conserving incompressible flow solver requires a conservative method for advection. Consider a passively advected scalar ϕ in a divergence-free velocity field $\vec{u}$ which is evolved using the equation

$$\phi_t + \vec{u}\cdot\nabla\phi = 0. \tag{1}$$

Throughout the paper, we denote the value of ϕ at position $\vec{x}$ and at time t by $\phi(\vec{x},t)$. We use $\vec{x}_k$ to denote the location of the center of cell k. When updating ϕ to time t^{n+1}, the traditional semi-Lagrangian method [Sta99] traces a characteristic ray from cell j backwards in time to some position $\vec{x}^\star$ and interpolates ϕ values from the surrounding cells to obtain a ϕ value at $\vec{x}^\star$. This value is then used as the new value of ϕ at cell j, i.e., $\phi(\vec{x}_j,t^{n+1})$. In other words, the semi-Lagrangian method updates ϕ at $\vec{x}_j$ as

$$\phi(\vec{x}_j,t^{n+1}) = \phi(\vec{x}^\star,t^n) = \sum_{i\in N(\vec{x}^\star)} w_{ij}\phi(\vec{x}_i,t^n), \tag{2}$$

where $N(\vec{x}^\star)$ denotes the set of cells i near position $\vec{x}^\star$ and w_{ij} are the interpolation weights from cell centers $\vec{x}_i$ to the point $\vec{x}^\star$. Note that cell i could be in the set $N(\vec{x}^\star)$ for several points $\vec{x}^\star$ and several cells j. Also note that the sum $\beta_i = \sum_j w_{ij}$ represents the total amount of ϕ removed from cell i and is typically not equal to 1.

2.1. Conservation

[LGF11] proposes two additional steps beyond Equation 2 to make this scheme fully conserve ϕ. They note that when

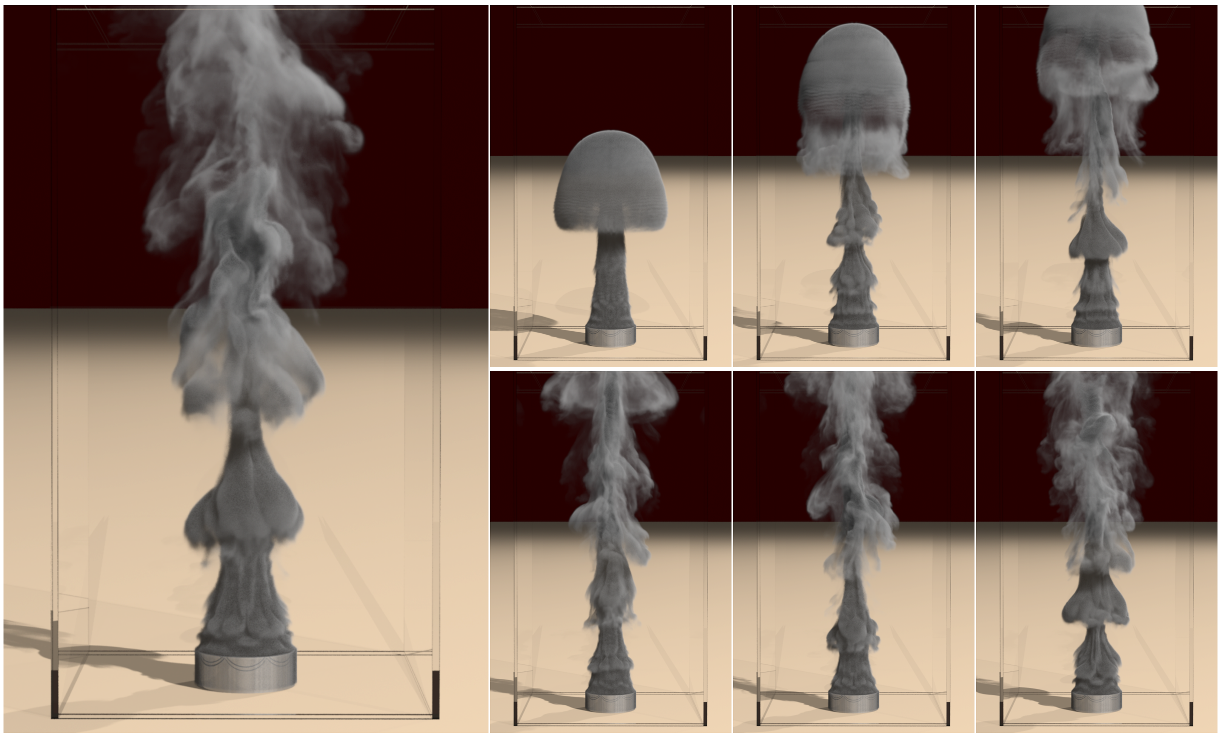

Figure 3: *An example using our conservative advection method with smoke injected from below simulated at one time step per frame using a high CFL number (approximately 40) at resolution* $256 \times 512 \times 256$.

$\beta_i > 1$, more ϕ is removed from cell i than exists at time t^n and modify the weights to be $\hat{w}_{ij} = w_{ij}/\beta_i$, for these cells. In addition, when $\beta_i < 1$, some of the ϕ in cell i is not advected with the flow. In this case, they perform a second forward advection step tracing a characteristic ray forward in time from cell i to some new position $\vec{x}^{\star\star}$, and distribute $(1-\beta_i)\phi(\vec{x}_i, t^n)$ to the cells in $N(\vec{x}^{\star\star})$ using an interpolation stencil. Therefore, if the interpolation weights from cell j to the point $\vec{x}^{\star\star}$ are α_{ij}, they distribute $(1-\beta_i)\alpha_{ij}\phi(x_i, t^n)$ to each cell j. This means that they modify Equation 2 to increment weights w_{ij} by an amount $(1-\beta_i)\alpha_{ij}$ to get $\hat{w}_{ij}$ for these cells as well as changing $N(\vec{x}^\star)$ to include all cells from the forward advection step that were not already accounted for during the backward advection step. If we redefine w_{ij} to be $\hat{w}_{ij}$ we note that Equation 2 still holds, except that now the weights w_{ij} have been first clamped (to guarantee that no excess amount of ϕ is removed from cell i) and then incremented (to account for any ϕ in cell i that had not yet been advected). These modifications guarantee that $\beta_i = 1$, which implies that for each cell i, the amount of ϕ advected to other cells is exactly equal to the amount of ϕ originally present in cell i, making the scheme fully conservative.

Although the authors show convergence for viscous flows under refinement in space and time, when experimenting with this scheme on coarse grids, with large time steps and inviscid flows, we observed that vortices were able to tear the flow apart, producing gaps such as those illustrated in Figure 2 (left) where a white spacing runs down the center line of the flow. We observed that the absence of smoke along the center line was caused because the clamping limits the amount of density that can reach these cells. The quantity $\gamma_j = \sum_i w_{ij}$ gives one an indication of how much of ϕ reaches each cell. For the traditional semi-Lagrangian scheme, $\gamma_j = 1$, meaning that every cell is filled, even though cells are oversampled or undersampled in order to do this. [LGF11] enforces $\beta_i = 1$ but typically produces $\gamma_j \neq 1$. For conservative incompressible flow, all of ϕ should be advected ($\beta_i = 1$) and every destination cell should be exactly filled ($\gamma_j = 1$). This can be viewed as making the weight matrix W doubly-stochastic.

We propose the following new scheme. First, we make the observation that the clamping and forward advection steps can be flipped, and thus, after the first semi-Lagrangian step we forward advect for all cells with $\beta_i < 1$ before clamping. This advection step ensures that $\beta_i \geq 1$ and $\gamma_j \geq 1$ for all cells. Next, we clamp γ_j to 1 by redefining all w_{ij} to be w_{ij}/γ_j, guaranteeing incompressibility. Unfortunately, to guarantee conservation of ϕ, one still has to rescale β_i to 1, and in general, β_i is not equal to 1 at this point. Simply clamping β_i would guarantee conservation and give improved results over the original scheme but would still give γ_j unequal to 1. To alleviate this problem one could iterate by alternately clamping γ_i and β_i, always ensuring to clamp β_i last to enforce conservation. Doing this will converge such that both γ_i and β_i are 1 but applying a diffusion-based opera-

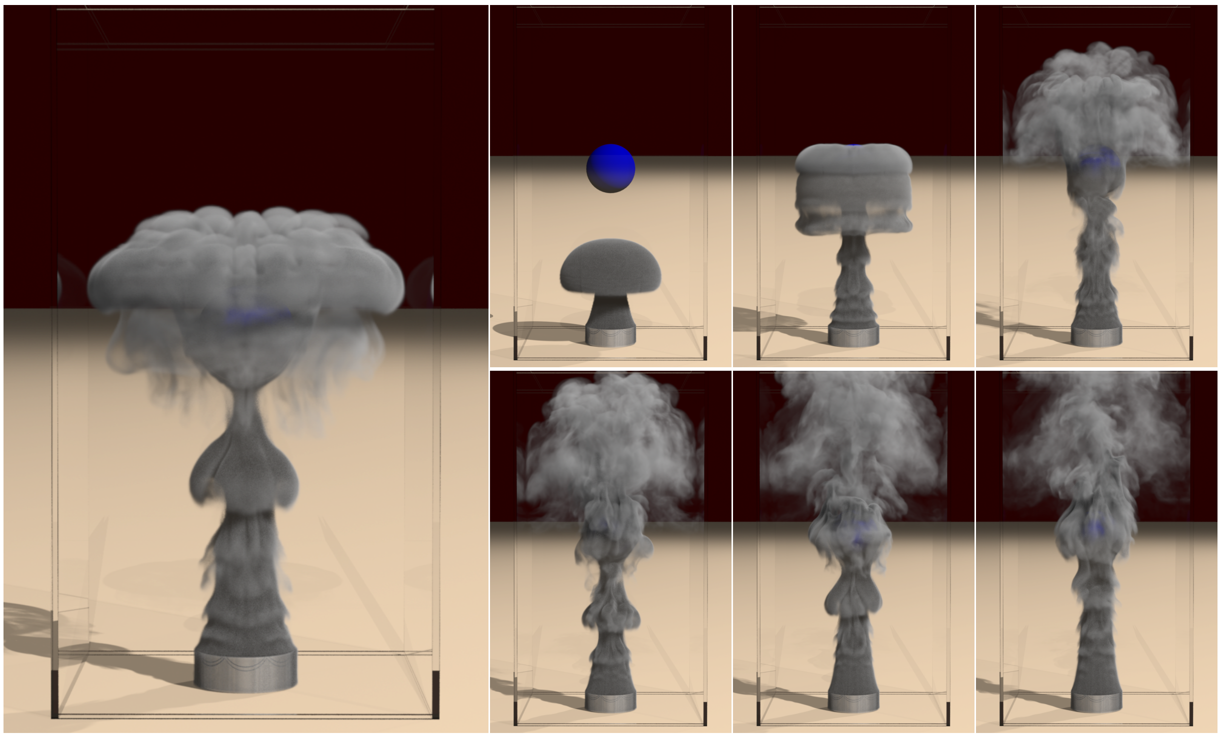

Figure 4: *An example using our conservative advection method with smoke injected from below and a static sphere simulated at one time step per frame using a high CFL number (approximately 40) at resolution* $256 \times 512 \times 256$.

tor is more efficient. For our examples, we use one clamping iteration and then apply our diffusion method (see below).

Diffusing the quantity ϕ would destroy the details of the flow, and so we diffuse the sum γ_j instead. Consider the heat equation $(\gamma_j)_t - \Delta\gamma_j = 0$ discretized with forward Euler in time and central differencing in space, which has an explicit time step restriction of $\Delta t = (\Delta x)^2/2d$, where d is the dimension. This means that between any two cells j and $j+1$ in three dimensions, the *flux* looks like $(\gamma_{j+1} - \gamma_j)\Delta t/(\Delta x)^2$ or $(\gamma_{j+1} - \gamma_j)/6$ (by substitution). We can then apply this flux to each of the six pieces in the seven point stencil independently. However, this process can be accelerated by considering the heat equation one dimension at a time, where $\Delta t = (\Delta x)^2/2$ and the flux looks like $(\gamma_{j+1} - \gamma_j)/2$. Thus, we sweep through the grid in a dimension by dimension manner considering every flux between adjacent grid points j and $j+1$, updating their values by the amount $(\gamma_{j+1} - \gamma_j)/2$. This is done using Gauss-Jacobi iterations within a dimension, but Gauss-Seidel iterations between dimensional sweeps.

Assuming $\gamma_{j+1} > \gamma_j$, we subtract off the difference $(\gamma_{j+1} - \gamma_j)/2$ from γ_{j+1} and add it to γ_j, making them both equal to $(\gamma_{j+1} + \gamma_j)/2$. This can be viewed as updating the weights $w_{k,j+1}$ and $w_{k,j}$ to $(w_{k,j+1} + w_{k,j})/2$, for all rows k. Observe that the sum $w_{k,j+1} + w_{k,j}$ does not change in the process implying that the sums β_j remain invariant under diffusion. We also advect ϕ during this update by moving the amount $\phi_{j+1}(\gamma_{j+1} - \gamma_j)/2\gamma_{j+1}$ from cell $j+1$ to cell j. Note that moving ϕ this way does not diffuse ϕ itself as evidenced by the fact that when all γ_j values are equal the ϕ values remain unchanged.

Diffusing the sums γ_j does not affect the sums β_j, so they remain 1. If the heat equation is solved to steady state, then γ_j equals 1 for all cells j. However, this turns out to be expensive and unnecessary. Hence, we actually clamp and diffuse the *cumulative weights*. We denote the cumulative weight at time t^n by γ_j^n, and initialize $\gamma_j^0 = 1$. These weights are advected forward in time in the same manner as ϕ to get $\tilde{\gamma}_j^{n+1}$. These advected weights are then used to clamp ϕ as described above, after which we apply a few iterations of our diffusion scheme to get γ_j^{n+1} (updating ϕ values in the process). Note that only a few iterations (between 1 and 7) are required in a time step as $\tilde{\gamma}_j^{n+1}$ incorporates the errors in incompressibility from previous iterations. One could alternatively use other existing methods for making the matrix W doubly-stochastic but we found that this method works well in our examples. It is important to note that although we are looking for a doubly-stochastic matrix we want a matrix that is as close as possible to the results from the conservative semi-Lagrangian method prior to adding diffusion.

2.2. Collisions

Although this method works well in the absence of solid objects we would also like to simulate examples such as the one shown in Figure 4. To conserve ϕ in the presence of objects, we modify both the forward and backward ray casting for interpolation to stop when it hits a solid object. We then use this surface point as $\vec{x}^{\star}$ (or $\vec{x}^{\star\star}$) for our interpolation weights. Unfortunately, this would still give interpolation weights coming from cells along the surface of the object. In this case, one can simply set those weights to 0, rescaling the remaining weights to sum to 1.

3. Incompressible Flow

Having developed a method for conservatively advecting a quantity, we can now apply it to incompressible flow in order to obtain a simulation that conserves both mass and momentum. We use our new advection method to passively advect necessary quantities such as smoke density $\hat{\rho}$ using the equation

$$\hat{\rho}_t + \vec{u}\cdot\nabla\hat{\rho} = 0, \tag{3}$$

which conserves the total mass throughout the simulation.

3.1. Navier-Stokes Equations

In order to conserve momentum, we solve the inviscid, incompressible Navier-Stokes equations, which are given by

$$\vec{u}_t + (\vec{u}\cdot\nabla)\vec{u} = -\frac{1}{\rho}\nabla p + \vec{f} \tag{4}$$

$$\nabla\cdot\vec{u} = 0, \tag{5}$$

where $\vec{u}$ is the velocity field of the fluid, ρ is the density of the fluid, $\vec{f}$ are any external forces (such as gravity) scaled by ρ, and p is the fluid pressure. We solve these equations by first calculating an intermediate velocity field $\vec{u}^{\star}$ via

$$\frac{\vec{u}^{\star} - \vec{u}^{n}}{\Delta t} + (\vec{u}^{n}\cdot\nabla)\vec{u}^{n} = \vec{f} \tag{6}$$

using our conservative advection method on the MAC grid. Since ρ is constant, this conserves momentum as well. We then subsequently add in the pressure forces via the equation

$$\frac{\vec{u}^{n+1} - \vec{u}^{\star}}{\Delta t} = -\frac{1}{\rho}\nabla p, \tag{7}$$

where the pressure is calculated by solving the Poisson equation

$$\nabla\cdot\frac{1}{\rho}\nabla\hat{p} = \nabla\cdot\vec{u}^{\star}, \tag{8}$$

where $\hat{p} = p\Delta t$.

We note that the pressure solve conserves momentum because pressure is applied in an equal and opposite manner to each neighboring $\vec{u}^{\star}$. This means that solving this system in the absence of external forces is fully mass and momentum conserving. We also note that since projection naturally conserves momentum and our method only modifies the advection step, other faster projection methods such as [LZF10] and [MST10] can be used as well.

3.2. Vorticity Confinement

Some external forces are meant to add momentum to the system. For example, gravity adds momentum to the system, which is allowed because in reality momentum would be conserved if the earth was simulated. However, some forces such as vorticity confinement [FSJ01, SRF05, YKH*09, JKB*10], which are necessary to make interesting flow fields, are not momentum conserving. Heuristically, vorticity confinement should conserve momentum globally since spinning things in a circle tends to produce equal amounts of linear momentum in each direction. It turns out that it is straightforward to make vorticity confinement conservative; we do this as follows. For each spatial dimension, sum all the forces in that dimension, divide the resultant cumulative force by the number of cells, and decrement the force at every cell by the net force divided by the number of cells. We denote this momentum conserving vorticity confinement force as $\vec{F}^{\star}$. Finally, add the momentum conserving force $\varepsilon\vec{F}^{\star}$, where ε is a scale parameter.

4. Smoke Simulation

Although the traditional semi-Lagrangian method allows us to take bigger time steps than conditionally stable methods, these large time steps result in noticeable errors in the simulation. For conditionally stable simulations, one typically uses the CFL number in order to define stability. A CFL number of 1 means that semi-Lagrangian rays have a max-

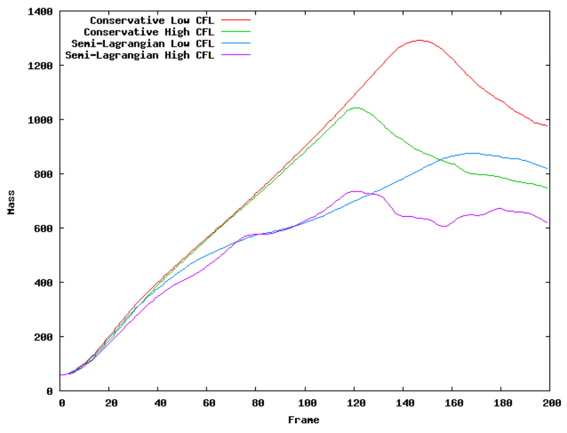

Figure 5: *A comparison between four simulations at resolution $128\times256\times128$. The red and green lines are simulations using our conservative scheme. Note that the difference in these at later frames is due to different amounts of smoke exiting the domain as the simulations are different with largely different time steps - but we stress that smoke is fully conserved in both cases. Comparing the blue to the red, or similarly the purple to the green shows the amount of mass loss suffered by the traditional semi-Lagrangian method. Note that an appreciable amount of mass is lost even before large amounts of smoke starts exiting the domain.*

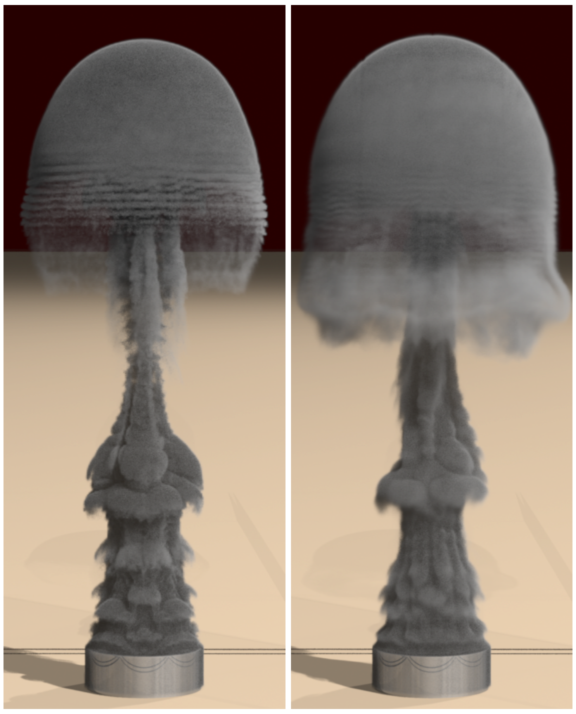

Figure 6: *A comparison between simulations of (Left) the traditional semi-Lagrangian method and (Right) our method with a very large time step at resolution $256 \times 512 \times 256$. Note how the large time steps yield poor interpolation resulting in alternating gaps in the smoke; this is especially apparent slightly above the ground plane and in the large plume. Conserving the amount of smoke, as done by our method, does not produce these artifacts.*

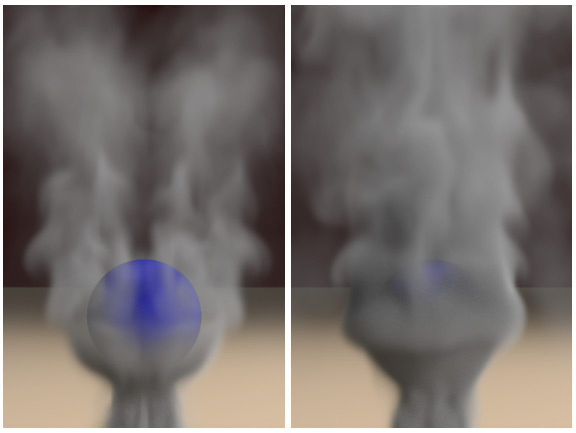

Figure 7: *A comparison between simulations of (Left) the traditional semi-Lagrangian method and (Right) our method using a typical CFL of 1 at resolution $128 \times 256 \times 128$. Note the large amount of mass lost when the smoke interacts with the sphere as illustrated in Figure 5.*

imum length of 1 grid cell, whereas a CFL number of 20 would allow them to trace back as many as 20 grid cells.

We demonstrate our method on several smoke simulation examples ran at 24 frames per second, as shown in Figures 3 and 4. We simulated two smoke examples, one with a ball and the other without it. Both examples have a density source at the bottom of the domain. For each example, we ran four simulations at resolution $128 \times 256 \times 128$: traditional semi-Lagrangian method with a CFL number of 1, our new method with the same CFL number, traditional semi-Lagrangian method at frame rate (1 time step per frame), and our method at frame rate. We also ran simulations at resolution $256 \times 512 \times 256$. Running these simulations using a low CFL number was infeasible, so we ran both examples at frame rate. In the lower resolution case, running at frame rate is equivalent to using a CFL number of 20. For the higher resolution ones the CFL number was around 40. This corresponds to taking 1/40 as many time steps as a CFL 1 simulation.

We compared our method to the traditional semi-Lagrangian method for all these examples. Figure 6 shows a comparison of the high resolution examples. Note the large amount of dissipation and artifacts that traditional semi-Lagrangian examples have compared to our method. We also quantitatively compared the two methods, as shown in Figure 5. For the lower resolution simulations we also compared our method with the traditional semi-Lagrangian method using a low CFL number (see Figure 7), demonstrating that even with low CFL numbers we achieve an increase in visual fidelity. In particular, note the large amount of mass lost both near and above the sphere using the traditional semi-Lagrangian method.

It is important to note that as the grid resolution increases by a factor of 2, the cost increases by a factor of 8 in space and a factor of 2 in time. However, using our method we can reduce this to only a factor of 8 in space as we can generate good results at very large time step sizes. We also note that although our advection method is about three times slower than the traditional semi-Lagrangian method, the projection step dominates the cost for smoke simulations (typically around 90% of the simulation time) and thus our method achieves performance comparable to the traditional semi-Lagrangian method for each time step.

5. Water Simulation

In addition to smoke, we applied momentum conservation to water. To do this we used the particle level set method [FF01, EFFM02, EMF02], although one could also use methods such as coupled level set volume of fluid (CLSVOF) [MUM*06], marker level set [MMS09] or front tracking [BBB10, WTGT10]. In order to make water conservative, momentum needs to be conserved during the velocity advection. However, our algorithm, as described in Section 2, cannot be directly applied because this would allow

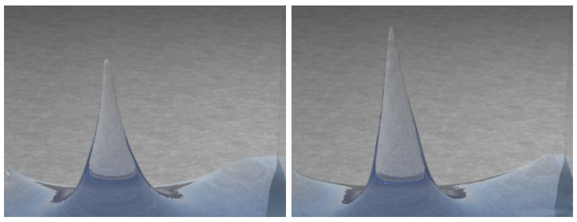

Figure 8: *A comparison between water simulations using (Left) the traditional semi-Lagrangian method and (Right) our method at resolution* $128 \times 256 \times 128$*. Note the improvement in momentum seen using our conservative method. In particular, the height of the splash is higher using our method. Note that in this example our method has 25% more momentum than the standard method.*

momentum to be advected from cells outside the level set (resulting in a gain in momentum), or allow momentum to be advected to cells outside the level set (resulting in a loss in momentum). To deal with these issues we modify both the backward and forward advection steps.

First, the standard semi-Lagrangian method is used to advect the level set from time t^n to t^{n+1}. Next, we consider the velocity advection step of our algorithm. We modify the algorithm described in Section 2.1 to use the time t^n level set as a collision body when doing backwards interpolation (see Section 2.2), ensuring that we only advect momentum that was part of the water volume. Note that some of the characteristic rays may not land in the level set at time t^n due to numerical errors. With reference to Figure 9 as an example, this would mean that faces $2,3,4,5$ would all be outside the level set at time t^n. However, at least one of the two adjacent cells A or B at time t^{n+1} must have had a valid characteristic which landed inside the level set (say, at G), otherwise that particular velocity degree of freedom would not be inside the level set at time t^{n+1}. We then use A and/or B to find valid velocities. However, since these rays end up half of a grid cell away from the velocity value, we find a new velocity by tracing a ray half of a grid cell in the appropriate direction from the base of these characteristics. This would change the position from 6 to 7 which we can then use to update 1. In the case of both A and B being inside the levelset we use the average of the two approximations of 1. Note that when going half of a grid cell in one direction or the other from the base of the characteristic, we trace a ray colliding with the level set at time t^n just as before.

For forward advection we treat the time t^{n+1} level set as a collision body. However, we also need to treat forward advection a bit differently since there is no guarantee that time t^n values of the level set land within the time t^{n+1} level set using the particle level set method. Therefore, in the case when a velocity characteristic does not land within the time t^{n+1} level set, we simply find the nearest point on the surface and allocate the velocity to that point. Our intention here is not to provide an in-depth analysis for water simulation, as the case for our new method is adequately made via smoke simulation. However, we did wish to discuss some of the issues involved in adapting it for water and show some preliminary simulations. Figure 8 shows a comparison between our method and the traditional semi-Lagrangian method. In this example, momentum conservation produces a higher splash as expected (see also the video).

6. Energy

In addition to mass and momentum conservation, we can also conserve the energy of the simulation. Energy conservation has been researched recently as a way to reduce dissipation (see e.g. [SB08, MCP*09, WP10, PTC*10]). In order to conserve the overall energy, we need to make sure energy is conserved in every step of the simulation. To do this in a visually appealing manner, we choose to add the energy that was lost at every time step with vorticity confinement. The role of vorticity in spinning up or slowing down a flow works to our advantage for adding or removing energy. Moreover, it can be difficult to add energy via a force-based method as projection can remove any force added at this step, but vorticity confinement is by nature incompressible (not exactly, but approximately) and therefore mostly survives the projection step.

Consider adding a momentum conserving vorticity confinement force $\vec{F}$

$$\vec{u}_c = \vec{u}^\star + \varepsilon\vec{F} \tag{9}$$

to the incremental velocity from Equation 6. The increase in energy due to this force can be calculated as

$$E = \frac{1}{2}\left(\sum_i m\vec{u}_c^2 - \sum_i m\vec{u}^{\star 2}\right) \tag{10}$$

where m is the mass of the cell. Equation 10 simplifies to

$$2E = \varepsilon^2 \sum_i m|\vec{F}|^2 + 2\varepsilon \sum_i m\vec{u}^\star \cdot \vec{F} \tag{11}$$

Solving this quadratic, we get

$$\varepsilon = \frac{-\sum_i m\vec{u}^\star \cdot \vec{F} \pm \sqrt{(\sum_i m\vec{u}^\star \cdot \vec{F})^2 + 2E\sum_i m|\vec{F}|^2}}{\sum_i m|\vec{F}|^2} \tag{12}$$

We choose the root such that $\varepsilon = 0$ when $E = 0$. This means

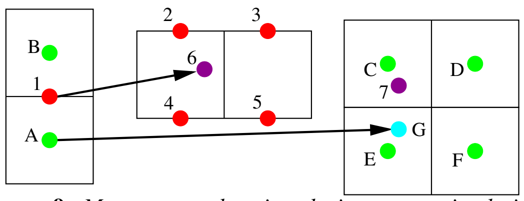

Figure 9: *Momentum advection during water simulation: shown are the semi-Lagrangian rays used to advect phi values (green) and velocity values (red). Note that when advecting the interpolated velocity value, 6 is used if valid, otherwise 7 may be used.*

that the $\pm$ is chosen to be the same sign as $\sum_i m\vec{u}^\star \cdot \vec{F}$. Note that one root is equivalent to vorticity spinning the flow in one direction to add energy and in the other direction to minimize energy, whereas the other root corresponds to vorticity confinement looking to “invert" the entire flow field, which is non-physical.

6.1. Tracking Energy

E is updated after every step in order to track how much energy is gained or lost. For advection, we simply advect $KE = m|\vec{u}^n|^2/2$ conservatively using our newly modified version of Equation 2 and compare the advected value of $KE^\star$ with $m|\vec{u}^\star|^2/2$ for each cell, where $\vec{u}^\star$ is obtained via conservative momentum advection of $\vec{u}^n$.

The change in energy due to projection can be calculated at each 1-dimensional MAC grid cell as

$$\begin{aligned}\Delta K &= \frac{\rho}{2}\left((u^{n+1})^2 - (u^\star)^2\right)\\ &= \frac{\rho}{2}\left(u^{n+1}+u^\star\right)\left(u^{n+1}-u^\star\right)\\ &= \frac{\rho}{2}\left(u^{n+1}+u^\star\right)\left(-\frac{\Delta t}{\rho}p_x\right)\\ &= -\Delta t\bar{u}\,(p_x)\end{aligned}$$

where $\bar{u} = (u^{n+1}+u^\star)/2$. ΔKV, where V is the volume of the cell, is added to E. Note that this is corrected during the vorticity confinement application for the next time step.

Figure 10 illustrates an example of a closed box with an initial flow field. The flow is then allowed to evolve, and we add fish tracer particles to visualize the flow field. Note that the flow never dies down and always conserves the energy in the system, as shown in Figure 11.

If external forces are divergence-free, then this method is fully conservative. Non-divergence free external forces modify the pressure field, and therefore ΔK in some sense includes energy added due to these external forces. For example, if we add gravity to the system (with a ground) and project the resulting flow, the projection will lose a large portion of the energy that was added with gravity, and we do not wish to treat this as energy lost. If we did, the energy would constantly increase even for an isolated system. One way to solve this is to simply do two projections, one to make the flow field divergence free before forces are added and another after the forces are added. However, this is a significant increase in cost.

Another way to find the energy lost is to figure out the total energy of the system at the start, keep track of the amount sourced in and out and store that as A. This allows us to calculate $E = A - PE - KE$ for vorticity confinement. For kinetic energy we simply calculate $KE = \sum_i m|\vec{u}^\star|^2/2$ for all cells i. For potential energy it depends on whether we are simulating smoke or water. For water we can calculate $PE = \sum_i mgh$ where g is the gravitational constant and h is the height. For smoke we use buoyancy instead of gravity. We calculate the potential energy as $PE = \sum_i m\hat{\rho}b(h_o - h)$ where b is the buoyancy constant, $\hat{\rho}$ is the smoke density and h_o is a constant. Using this equation, we see that as smoke rises from buoyancy the potential energy decreases and the kinetic energy increases to account for this loss.

Using this formulation we can simulate examples with gravity, such as the one shown in Figure 10 right, where we drop a ball of water into a pool of shallow water, which re-

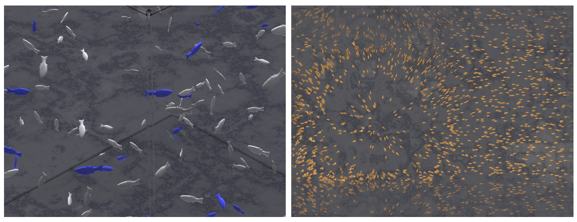
Figure 10: *(Left) A simulation of energy conserving incompressible flow at resolution $64\times64\times64$. The initial flow field is created by starting with upwards velocities in the center of the domain and zero elsewhere. The flow is then made divergence free and simulated forward in time, and we note that conserving energy provides a sustainable flow field for long-time simulation as seen in Figure 11 (also see the video). Fish models are passively advected to visualize the flow field. (Right) A simulation of energy conserving free surface flow at resolution $64\times64\times128$. The initial flow field is created by dropping a ball of water into a pool of shallow water (viewed from top down).*

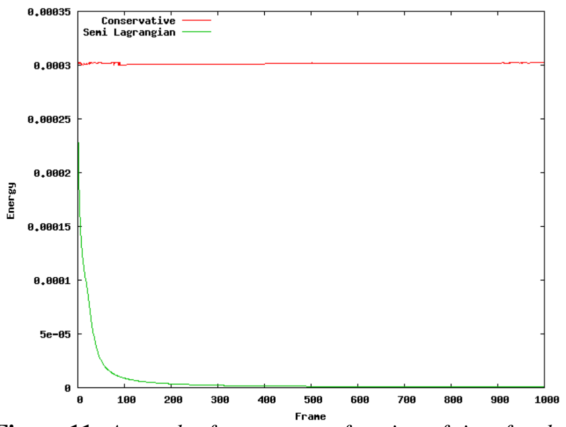

Figure 11: *A graph of energy as a function of time for the simulation in Figure 10. The red line shows that our method conserves energy almost exactly for 1000 frames. Note that we do exactly conserve energy when comparing two velocity fields before projection. However, projection removes or adds a very small amount of energy as can be seen by the wiggles in the red line. For comparison we also plot the results for the same simulation using the traditional semi-Lagrangian method as a green line, and note that the energy quickly dies out.*

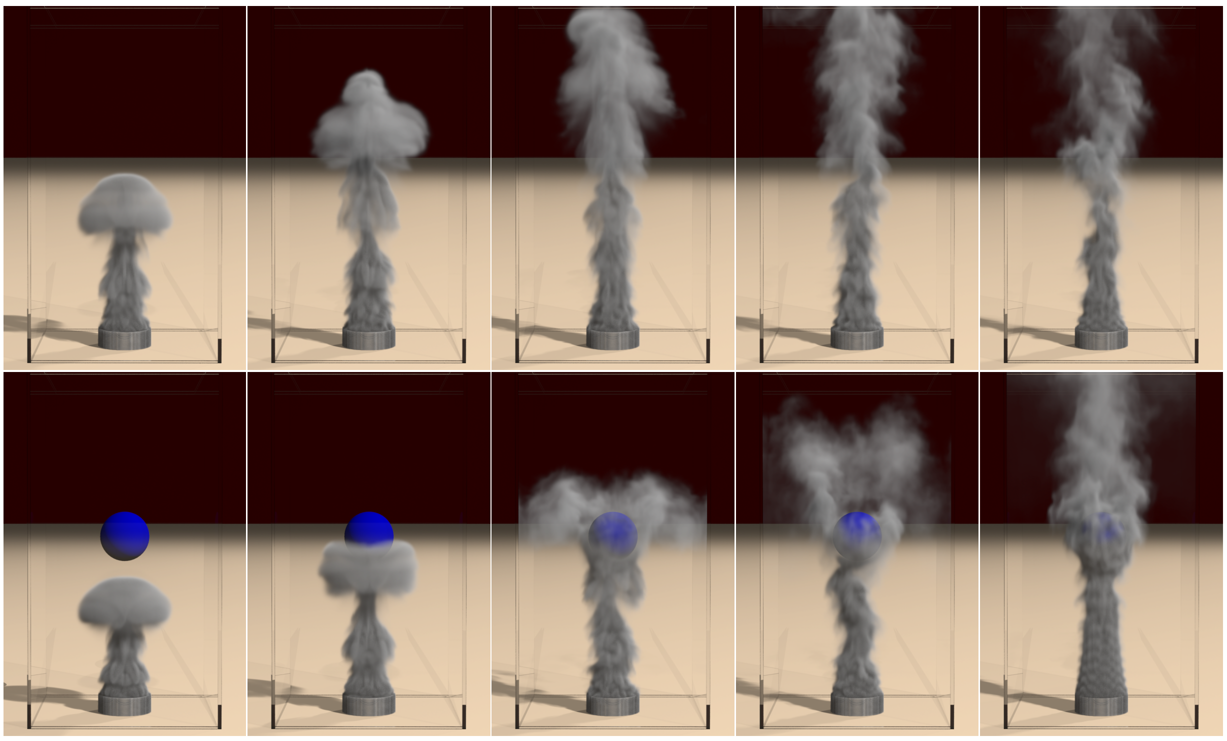

Figure 12: *Two simulations using our method with energy conservation with smoke injected from below at resolution* $128 \times 256 \times 128$. *Note that no vorticity confinement was added other than that used to conserve energy. Also note that the resulting density field appears significantly less viscous than the traditional semi-Lagrangian method which explicitly adds vorticity confinement.*

duces the potential energy and increases the kinetic energy. We can also simulate energy conserving smoke as shown in Figure 12. In this case, we did not add any vorticity confinement to the system other than what is added for energy conservation. Note the increased amount of detail obtained using this method. This allows us to provide a method for automatically and dynamically determining how much vorticity should be added into the simulation. However, if energy is lost and there are few areas of vorticity to add energy to, adding in energy can cause undesirable noise. To alleviate this problem we do not add vorticity for these steps, instead accumulating this energy to be added back once sufficient vorticity has developed. Alternatively, one could target the correct energy and add the energy lost slowly over time. However, for our examples we found that there was insufficient vorticity only near the very beginning where relatively little energy was lost. Thus, we could simply add in the total energy lost as soon as vortices started appearing.

7. Conclusion

We introduced a novel algorithm for mass and momentum conservation in incompressible flow. We designed a new advection method using the basic building blocks used in semi-Lagrangian advection which is known to work well for inviscid flows, coarse grids and large time steps, a scenario common in computer graphics. We also proposed a modification to the vorticity confinement model which conserves momentum. We have shown that by conserving mass and momentum we are able to run high quality simulations while taking very large time steps (at frame rate). Our technique can also be adapted to conserve momentum for water simulation. Finally, we showed a modification using vorticity confinement that preserves energy as well. By being able to run energy conserving fluid simulations with large time steps, we can create fast, interesting fluid flows that can be potentially used for other applications such as reduced order models or crowd simulations.

Acknowledgements

Research supported in part by ONR N00014-09-1-0101, ONR N00014-11-1-0027, ONR N00014-06-1-0505, ONR N00014-05-1-0479, for a computing cluster, NSF IIS-1048573 and King Abdullah University of Science and Technology (KAUST) 42959. M.L. was supported in part by an Intel Ph.D. Fellowship. M.A. was supported in part by the Nokia Research Center. We would like to thank Christos Kozyrakis for additional computing resources as well as Jacob Leverich for helping us use those resources. We would also like to thank Craig Schroeder for his fish model.

References

[BBB07] BATTY C., BERTAILS F., BRIDSON R.: A fast variational framework for accurate solid-fluid coupling. *ACM Trans. Graph. (SIGGRAPH Proc.) 26*, 3 (2007), 100.

[BBB10] BROCHU T., BATTY C., BRIDSON R.: Matching fluid simulation elements to surface geometry and topology. *ACM Trans. Graph.* (2010).

[BXH10] BATTY C., XENOS S., HOUSTON B.: Tetrahedral embedded boundary methods for accurate and flexible adaptive fluids. In *Proceedings of Eurographics* (2010).

[CFL*07] CHENTANEZ N., FELDMAN B. E., LABELLE F., O'BRIEN J. F., SHEWCHUK J. R.: Liquid simulation on lattice-based tetrahedral meshes. In *ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.* (2007), pp. 219–228.

[CGFO06] CHENTANEZ N., GOKTEKIN T. G., FELDMAN B., O'BRIEN J.: Simultaneous coupling of fluids and deformable bodies. In *ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.* (2006), pp. 325–333.

[CMT04] CARLSON M., MUCHA P. J., TURK G.: Rigid fluid: Animating the interplay between rigid bodies and fluid. *ACM Trans. Graph. (SIGGRAPH Proc.) 23* (2004), 377–384.

[DL03] DUPONT T., LIU Y.: Back and forth error compensation and correction methods for removing errors induced by uneven gradients of the level set function. *J. Comput. Phys. 190/1* (2003), 311–324.

[EFFM02] ENRIGHT D., FEDKIW R., FERZIGER J., MITCHELL I.: A hybrid particle level set method for improved interface capturing. *J. Comput. Phys. 183* (2002), 83–116.

[EMF02] ENRIGHT D., MARSCHNER S., FEDKIW R.: Animation and rendering of complex water surfaces. *ACM Trans. Graph. (SIGGRAPH Proc.) 21*, 3 (2002), 736–744.

[FF01] FOSTER N., FEDKIW R.: Practical animation of liquids. In *Proc. of ACM SIGGRAPH 2001* (2001), pp. 23–30.

[FOK05] FELDMAN B., O'BRIEN J., KLINGNER B.: Animating gases with hybrid meshes. *ACM Trans. Graph. (SIGGRAPH Proc.) 24*, 3 (2005), 904–909.

[FOKG05] FELDMAN B., O'BRIEN J., KLINGNER B., GOKTEKIN T.: Fluids in deforming meshes. In *Proc. of the ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.* (2005).

[FSJ01] FEDKIW R., STAM J., JENSEN H.: Visual simulation of smoke. In *Proc. of ACM SIGGRAPH 2001* (2001), pp. 15–22.

[GSLF05] GUENDELMAN E., SELLE A., LOSASSO F., FEDKIW R.: Coupling water and smoke to thin deformable and rigid shells. *ACM Trans. Graph. (SIGGRAPH Proc.) 24*, 3 (2005), 973–981.

[JKB*10] JANG T., KIM H., BAE J., SEO J., NOH J.: Multilevel vorticity confinement for water turbulence simulation. *Vis. Comput. 26* (2010), 873–881.

[KFCO06] KLINGNER B. M., FELDMAN B. E., CHENTANEZ N., O'BRIEN J. F.: Fluid animation with dynamic meshes. *ACM Trans. Graph. 25*, 3 (2006), 820–825.

[KGF10] KWATRA N., GRÉTARSSON J. T., FEDKIW R.: Practical animation of compressible flow for shock waves and related phenomena. In *ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.* (2010), pp. 207–215.

[KLLR05] KIM B.-M., LIU Y., LLAMAS I., ROSSIGNAC J.: Using BFECC for fluid simulation. In *Eurographics Workshop on Natural Phenomena 2005* (2005).

[LFO06] LOSASSO F., FEDKIW R., OSHER S.: Spatially adaptive techniques for level set methods and incompressible flow. *Computers and Fluids 35* (2006), 995–1010.

[LGF04] LOSASSO F., GIBOU F., FEDKIW R.: Simulating water and smoke with an octree data structure. *ACM Trans. Graph. (SIGGRAPH Proc.) 23* (2004), 457–462.

[LGF11] LENTINE M., GRÉTARSSON J., FEDKIW R.: An unconditionally stable fully conservative semi-lagrangian method. *J. Comput. Phys. (to appear)* (2011).

[LZF10] LENTINE M., ZHENG W., FEDKIW R.: A novel algorithm for incompressible flow using only a coarse grid projection. *ACM Transactions on Graphics* (July 2010).

[MCP*09] MULLEN P., CRANE K., PAVLOV D., TONG Y., DESBRUN M.: Energy-preserving integrators for fluid animation. In *SIGGRAPH '09: ACM SIGGRAPH 2009 papers* (2009), pp. 1–8.

[MMS09] MIHALEF V., METAXAS D. N., SUSSMAN M.: Simulation of two-phase flow with sub-scale droplet and bubble effects. *Comput. Graph. Forum* (2009).

[MST10] MCADAMS A., SIFAKIS E., TERAN J.: A parallel multigrid poisson solver for fluids simulation on large grids. In *SCA/Eurographics Symp. on Comput. Anim.* (2010), ACM Press, pp. 1–10.

[MUM*06] MIHALEF V., UNLUSU B., METAXAS D., SUSSMAN M., HUSSAINI M.: Physics based boiling simulation. In *SCA '06: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2006), pp. 317–324.

[PTC*10] PFAFF T., THUEREY N., COHEN J., TARIQ S., GROSS M.: Scalable fluid simulation using anisotropic turbulence particles. In *ACM SIGGRAPH Asia 2010 papers* (2010), SIGGRAPH ASIA '10, pp. 174:1–174:8.

[RMSG*08] ROBINSON-MOSHER A., SHINAR T., GRÉTARSSON J., SU J., FEDKIW R.: Two-way coupling of fluids to rigid and deformable solids and shells. *ACM Trans. on Graphics 27*, 3 (Aug. 2008), 46:1–46:9.

[SB08] SCHECHTER H., BRIDSON R.: Evolving sub-grid turbulence for smoke animation. In *SCA '08: Proc. of the 2008 ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.* (2008), pp. 1–7.

[SFK*08] SELLE A., FEDKIW R., KIM B., LIU Y., ROSSIGNAC J.: An Unconditionally Stable MacCormack Method. *J. of Sci. Comp. 35*, 2 (2008), 350–371.

[SGTL09] SEWALL J., GALOPPO N., TSANKOV G., LIN M.: Visual Simulation of Shockwaves. *Graphical Models* (2009).

[SRF05] SELLE A., RASMUSSEN N., FEDKIW R.: A vortex particle method for smoke, water and explosions. *ACM Trans. Graph. (SIGGRAPH Proc.) 24*, 3 (2005), 910–914.

[Sta99] STAM J.: Stable fluids. In *Proc. of SIGGRAPH 99* (1999), pp. 121–128.

[WP10] WEISSMANN S., PINKALL U.: Filament-based smoke with vortex shedding and variational reconnection. *ACM Trans. Graph.* (2010).

[WTGT10] WOJTAN C., THÜREY N., GROSS M. H., TURK G.: Physics-inspired topology changes for thin fluid features. *ACM Trans. Graph.* (2010).

[YKH*09] YOON J.-C., KAM H. R., HONG J.-M., KANG S.-J., KIM C.-H.: Procedural synthesis using vortex particle method for fluid simulation. *Comput. Graph. Forum 28*, 7 (2009), 1853–1859.

[YOH00] YNGVE G., O'BRIEN J., HODGINS J.: Animating explosions. In *Proc. of ACM SIGGRAPH 2000* (2000), pp. 29–36.