Detailed Water with Coarse Grids: Combining Surface Meshes and Adaptive Discontinuous Galerkin

Essex Edwards* Robert Bridson*

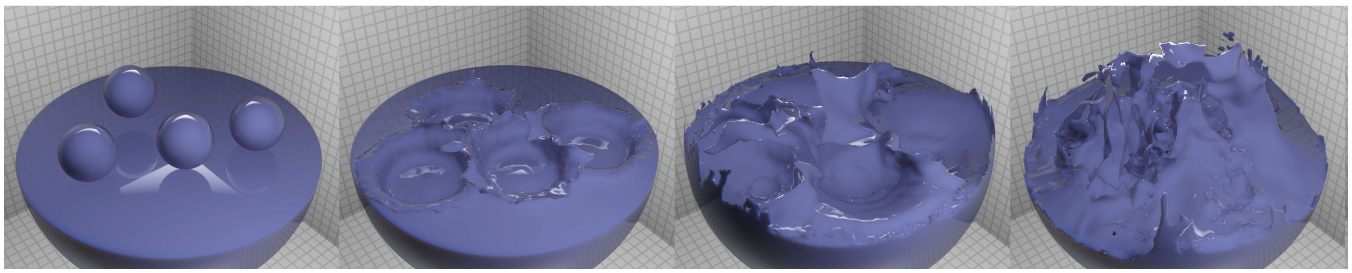

Figure 1: *A simulation in a* $25 \times 25 \times 25$ *grid generates thin splashes and sheets down to* $1/1200$ *the domain width.*

Abstract

We present a new adaptive fluid simulation method that captures a high resolution surface with precise dynamics, without an inefficient fine discretization of the entire fluid volume. Prior adaptive methods using octrees or unstructured meshes carry large overheads and implementation complexity. We instead stick with coarse regular Cartesian grids, using detailed cut cells at boundaries, and discretize the dynamics with a p-adaptive Discontinuous Galerkin (DG) method. This retains much of the data structure simplicity of regular grids, more efficiently captures smooth parts of the flow, and offers the flexibility to easily increase resolving power where needed without geometric refinement.

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: fluid simulation, liquids, meshes, adaptive, discontinuous Galerkin **Links:** DL PDF

1 Introduction

Simulating liquids for visual effects demands a high resolution surface with detailed motion, but typically not the same high resolution in the entire volume: it is only the surface that we observe, and experiment and theory often indicate that most of the interesting dynamics (e.g. strong vorticity) are generated at the surface and remain strongest near the surface. We cannot avoid all volumetric computation, but much is gained by concentrating the bulk of the computation on the surface, as surface area scales quadratically with size while volume scales cubically. We are interested in techniques that take advantage of this opportunity. For liquid simulation, this encompasses methods for the surface tracker, the volumetric velocity/pressure solver, and their interactions.

Surface tracking may use an implicit method (e.g. level-set or volume-of-fluid), explicit method (e.g. marker particles or mesh), or a hybrid (e.g. particle level set). These approaches are already capable of spending computation and memory only at the surface. In this paper we use an existing method to track the surface with an explicit triangle mesh, and concentrate on handling the dynamics. While the surface tracker may appear to be conceptually independent of the dynamics' discretization, artifacts can easily arise if the surface tracker and dynamical model are poorly coupled.

Broadly speaking, there have been two approaches to simulating a high-resolution surface without a correspondingly high-resolution mesh for the entire volume. The first category uses a simple coarse volumetric fluid model everywhere, and adds a secondary model for the missing high-resolution surface features. These methods are generally quite fast and attractively simple, but typically make simplifying assumptions that can lead to physically incorrect behavior. The second category of methods use an adaptive volumetric mesh that matches the high resolution at the surface but is low resolution in the interior of the liquid. The unified handling of all the dynamics is physically consistent and correct, but much more computationally expensive, in large part due to the complexity of using unstructured or semi-structured meshes.

We present an adaptive method that is physically consistent and correct, but still uses a simple coarse Cartesian grid. To capture high-resolution surface details we use detailed cut cells at boundaries, and adaptively use richer discrete models within grid cells near the surface, more specifically a p-adaptive Discontinuous Galerkin (DG) method.

In summary, our core technical contributions are:

- the first application of DG with exact cut cells to moving free-surface problems, and
- a novel particle advection scheme that requires fewer evaluations of the velocity field.

We also highlight some ideas that are new to fluid simulation in graphics:

- p-adaptive techniques, and
- embracing discontinuous approximations at all levels.

*(essex|rbridson)@cs.ubc.ca, University of British Columbia

2 Related Work

Fluid simulation has a long history in computer graphics, and an even longer history in scientific computing, engineering, and physics. For an overview of the major techniques in graphics, we refer the reader to Bridson's book [2008].

This paper uses the hybrid Lagrangian-Eulerian FLIP method [Zhu and Bridson 2005]. FLIP is an extension of the Eulerian velocity-pressure formulation with staggered time stepping [Foster and Metaxas 1996]. Since its introduction to graphics, FLIP has seen several developments related to this work. Adaptive FLIP particle distributions have been introduced for use in an h-adaptive simulation [Ando et al. 2013] and for detailed tracking of thin sheets [Ando et al. 2012]. Other particle methods (e.g. SPH) can also have adaptive particle distributions, including generally h-adaptive approaches [Adams et al. 2007] and special two-scale approaches [Solenthaler and Gross 2011]. Interpolation between FLIP particles and high-order grids has been seen outside of graphics with both smooth [Edwards and Bridson 2012] and non-smooth [Moresi et al. 2003] interpolants.

For tracking the liquid surface, marker particles [Foster and Metaxas 1996] and level-set methods [Osher and Fedkiw 2003] or their combination in the particle level-set [Enright et al. 2002] have been most popular. Recently, explicit surface tracking with triangle meshes has gained interest; e.g. reviewed in a recent SIGGRAPH course [Wojtan et al. 2011]. In this work we use the El Topo explicit surface tracking library of Brochu et al. [2009].

As mentioned above, some methods embed a high-resolution surface tracker in a low-resolution volumetric simulation and apply a second model to the surface to control sub-grid motion. Regularizing the surface so that its topology is representable in the grid-based physics simulation has been recognized as an important feature in these approaches: without regularization, fine scale details that are not captured in the simulation grid behave non-physically. For simulations of elastic or very viscous materials, simple persistence of fine details may be perfectly acceptable [Wojtan and Turk 2008; Wojtan et al. 2009; Wojtan et al. 2010]. For liquids, fine surface details should not persist. Müller [2009] handles this by remeshing the surface every timestep, and restricting the geometry and topology within each cell to be very simple. Thürey et al. [2010] applied surface-tension and mean-curvature flow to smooth small details in a physically-motivated way. For situations with less surface tension, good results were achieved with a surface model resembling vortex sheet equations [Bojsen-Hansen and Wojtan 2013].

A second class of approaches use h-adaptive volumetric simulations to put smaller diameter (h) volumetric elements near the surface, but large elements in the bulk flow, capturing surface dynamics in the same way as the volume dynamics. Classic examples of this are octrees [Losasso et al. 2004] and unstructured tetrahedral meshes, including Eulerian approaches [Chentanez et al. 2007; Batty et al. 2010; Ando et al. 2013] and Lagrangian approaches [Misztal et al. 2012; Misztal et al. 2010; Misztal and Bærentzen 2012]. Recognizing the simplicity and efficiency of regular grids, chimera grids combine multiple regular grids of different resolution [English et al. 2013] and tall-cell grids go adaptive only in the vertical direction [Irving et al. 2006; Chentanez and Müller 2011]. Special h-adaptive schemes designed for explicit surface trackers are also possible [Brochu et al. 2010].

The techniques in this paper are not h-adaptive, but rather p-adaptive, increasing the resolution in an area by increasing the approximation degree, p, in volumetric elements rather than by geometric refinement. To our knowledge, these techniques have never been applied in graphics, but have a long history in finite element methods (FEM). When the approximation degree is allowed to be very large, one arrives at spectral [Boyd 2001] and spectral element methods [Karniadakis and Sherwin 1999]. While we use pure p-adaptivity, both h and p adaptivity can be combined into hp-adaptive methods [Babuska and Guo 1992; Schwab 1998].

Our discretization of the Poisson/projection problem uses a Discontinuous Galerkin (DG) approach, a close relative of the famous finite element method (FEM). For a review of many DG methods for elliptic problems, we recommend the unified analysis by Arnold et. al [2002]. Of these, we make use of the Locally-Discontinuous Galerkin (LDG) method, which has been applied to many problems from Poisson to non-linear Navier-Stokes [Cockburn et al. 2005; Castillo 2006]. DG has been used previously in computer graphics for elastic deformations [Kaufmann et al. 2009], where its flexibility enabled the embedding of high-resolution solids inside low resolution volumetric meshes, similar to our use.

3 The Method

We work with the velocity-pressure (u-p) form of the Navier-Stokes equations, simplified by assuming incompressible, inviscid, and uniform-density fluid with free-slip conditions at solid-fluid boundaries and free-surface conditions with no surface-tension at the air-fluid boundaries. This common model is appropriate for simulating medium to large bodies of liquid.

We apply a staggered time-stepping, with separate *advection* and *projection* steps. The fluid velocity field is stored with FLIP particles, and the boundary location is stored with an explicit triangle mesh (Figure 6). An outline of one time step is given in Algorithm 1, including references to the section where each step is described.

Algorithm 1 Outline of a Time Step

Input: particle positions, velocities, and surface mesh at time t_0
Output: particle positions, velocities, and surface mesh at time t_1

1:	*Projection Step*:	§3.1
2:	Build cut-cell volume mesh	§3.2
3:	Prepare basis functions	§3.3
4:	Assemble Poisson matrix	§3.4
5:	Interpolate velocity from particles to grid	§3.7
6:	Solve linear system	§3.4
7:	If viscosity $\neq 0$, apply viscous update	§3.10
8:	Interpolate update from grid to particles	§3.6
9:	*Advection Step*:	
10:	Add particles where necessary	§3.8
11:	Advect FLIP particles, including surface	§3.8
12:	Smooth surface	§3.9
13:	Update surface tracker:	§3.9
14:	collision detection/response	
15:	topology changes	
16:	remeshing	
17:	Remove particles where necessary	§3.8
18:	Add body forces to FLIP velocities	

3.1 Discretizing Projection

The projection step of our solver takes an intermediate velocity field $\widetilde{u}$ from advection and gravity, and applies a pressure gradient to make it divergence-free while respecting solid and air boundaries. The projected velocity field u is found by

$$u + \nabla p = \widetilde{u}, \tag{1}$$

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

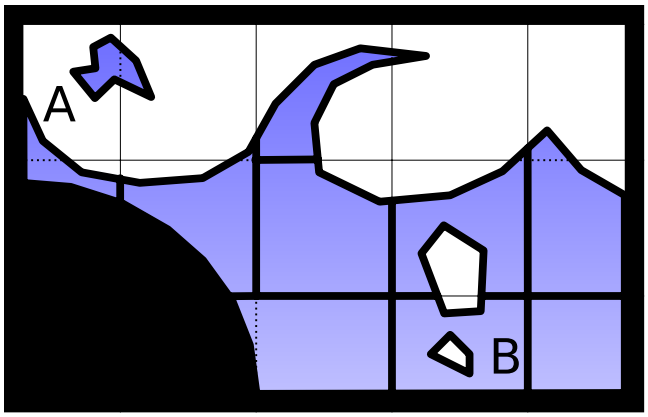

Figure 2: *A schematic diagram of the volumetric mesh (thick lines) constructed by intersecting the polyhedral liquid domain (blue fill) with the cubes of a regular grid (thin lines). Small cells are merged with neighbors (dashed lines). Note that (A) a grid cell may contain multiple connected components, and (B) a cut cell may have complex topology.*

in the liquid domain Ω, and subject to the boundary conditions.

Discretizing with DG follows the same outline as FEM. First the domain is partitioned into cells. Within each cell, some approximation space is specified for u and p. Finally, a weak form of the PDE is satisfied within this space. Unlike typical FEM, DG allows each cell to have different approximation spaces regardless of discontinuities across cell boundaries or agreement with boundary conditions. Furthermore, the cells need not be simple shapes.

3.2 The Volume Mesh

To construct our volume mesh, we begin with a regular Cartesian voxel grid with spacing h. In voxels that contain the boundary of the liquid, we intersect the liquid's volume and voxel to get a surface mesh of the liquid in just that cell. This is the detailed cut cell that we use in our discretization, replacing the cubes of the Cartesian grid, but keeping the same regular structure (Figure 2). This volume mesh conforms to the liquid boundary, exactly capturing all sub-grid features. The only simplification comes later, by projecting the PDE into the approximation space.

Each triangle of the explicit surface is assigned either the free-surface or solid boundary condition. Triangles that are close to the solid mesh and have normal oriented opposite the solid's normal are assigned solid-boundary conditions.

We need to avoid cut cells with small, skinny, or other otherwise poor geometry that can cause poor conditioning of the discretization. This issue has come up with elasticity [Kaufmann et al. 2009] and simpler fluid problems [Qin and Krivodonova 2012]. We follow a similar heuristic strategy of merging 'poor' cells with adjacent cells to create larger cells and better conditioning. It is always possible to avoid arbitrarily small volumes because our surface tracker does not produce arbitrarily thin features; see §3.9.

Deciding when and how to merge cells is a matter of heuristics, and our results were not sensitive to its details. When a cell's volume is small ($V < h^3/100$), we merge it with the adjacent cell with which it has the largest shared face. For cells with intermediate volume ($V < h^3/4$), we merge them with a neighbour only if the area (A) of their shared face is sufficiently large ($A^{3/2} > V$). The condition on shared area prevents undesired merging of thin fluid sheets into a single large thin cell.

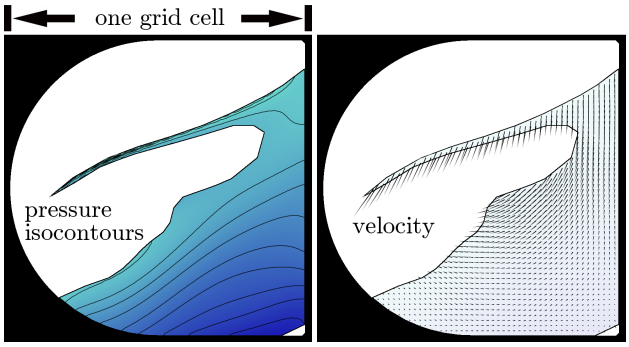

Figure 3: *A simulation entirely within one grid cell. Note the detailed pressure variation within the thin sheet and detailed sub-grid velocities. This example uses degree 6 polynomials for pressure: 28 variables.*

The location and size of the grid each frame is arbitrary, as it stores no data between time-steps. In our 3D implementation, we perturb its location each time-step to avoid degenerate cut-cell geometry.

3.3 The Approximation Space

DG requires an approximation space for u and p within each cell. For pressure, we use P_k, i.e. polynomials of degree no more than k. We could simply use the same space P_k for each component of velocity, but we use P_{k+1} instead because it provides a more accurate velocity for the same size pressure solve. Larger disparities, such as P_{k+2} for velocity, may be unstable. Figure 3 illustrates how these high-order approximation spaces can provide lots of sub-grid detail. If a cell has multiple connected components, each component uses a separate DG approximation space: we simply enrich the basis accordingly.

When applying high-order FEM on Cartesian grids, tensor product polynomials (i.e. Q_k, including terms like $x^k y^k$) are normally used, as these make continuity simple while achieving k^{th} order accuracy. However, Q^k has dimension $k^3 + \mathcal{O}(k^2)$ whereas P_k only has dimension $k^3/6 + \mathcal{O}(k^2)$ but achieves the same order of accuracy. Since DG allows discontinuities, we can use the P_k spaces, so the pressure solve has only one sixth the number of variables of an equivalent continuous discretization.

The solution is independent of the choice of basis, but it has large effects on the conditioning of the linear system. We use a nodal basis for P_k defined by points in a simplex, as commonly used for FEM. To adapt the basis to the irregularly-shaped cut cells, we approximately fit this simplex to each cell by fitting an oriented bounding box aligned with the cell's inertia tensor and another axis-aligned bounding box, taking the smaller of these boxes, and using the largest tetrahedron that fits in the box. This provides anisotropically stretched basis functions that match the anisotropy of thin sheets or tendrils of liquid.

The cornerstone of p-adaptivity is using different approximation spaces in different cells. For cells touching the free surface we use high-degree polynomials (typically cubic or quartic pressure fields). All other cells use one degree less than their neighbor of highest degree, but no less than linear pressure, as shown in Figure 4. Using linear pressure as the minimum ensures that the hydrostatic case is solved exactly. This p-adaptive approach allows us to use a coarse regular grid for the whole domain, while still achieving fine-scale details where desired.

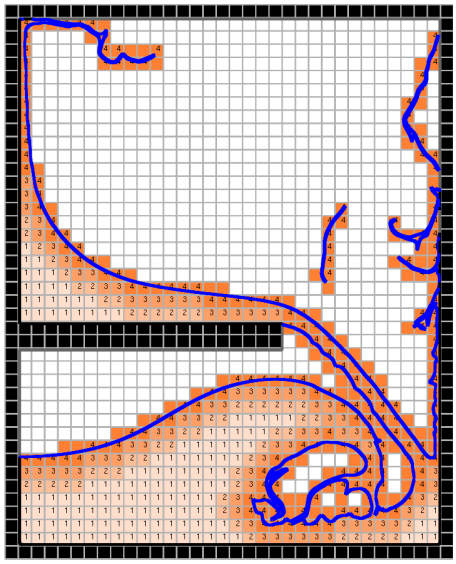

Figure 4: *This simulation uses quartic polynomials for pressure at the surface (red cells), and linear polynomials for pressure in the interior (light cells), with smooth grading of the polynomial degree in intermediate cells.*

For smooth functions, p-refinement is more efficient than h-refinement. Error estimates generally bound the error by $\mathcal{O}(h^p)$, while the number of degrees of freedom is $\mathcal{O}((p/h)^3)$. Dividing those expressions, the error per degree of freedom decreases exponentially with p but only geometrically with h. For non-smooth functions, this is false, but fortunately liquid simulations typically have smooth velocity and pressure, even when the surface geometry is non-smooth.

3.4 The LDG Equations

We use the Local Discontinuous Galerkin (LDG) method, which is well-studied, flexible, and has simple parameters [Cockburn et al. 2005]. Using LDG, when the solution is exactly representable in the approximation space, LDG finds that solution exactly, independent of geometry. Consequently quiescent free-fall and hydrostatic liquids (linear p, constant u) are exactly reconstructed. In general, the boundary conditions, continuity, and the PDE are all satisfied in a weak sense.

LDG for the Poisson problem was introduced by Castillo et al. [2000]. This section briefly follows their construction of the equations. We refer the reader to their paper for full details and alternative forms of the equations.

We look first at the divergence-free condition. Multiplying by an arbitrary smooth test-function q and integrating over an arbitrary domain $K \subseteq \Omega$ reveals $\int_K q \nabla \cdot u = 0$. Integrating by parts,

$$\int_K \nabla q \cdot u \, dV = \int_{\partial K} q \hat{u} \cdot n_K \, dA \tag{3}$$

where n_K is the unit outward normal on K.

For continuous functions, $u = \hat{u}$, but to apply this to discontinuous velocity fields, DG methods introduce a *numerical flux*, $\hat{u} \neq u$ at discontinuities. The discrete solution finds a velocity field in the approximation space that satisfies (3) for all cells K in the mesh and all test functions q in the pressure approximation space.

Different DG methods define different numerical fluxes. Let K^+ and K^- be two cells with a shared face, with normals $n^\pm$ and $u = u^\pm$ in $K^\pm$. Then, letting $\theta = \frac{1}{2} - C_{12} \cdot n^+$, LDG defines

$$\hat{u} = \theta u^+ + (1-\theta)u^- - C_{11}(n^+ p^+ + n^- p^-). \tag{4}$$

where C_{12} and C_{11} are parameters.

On the domain boundary, the flux is defined using the boundary conditions. At solid-boundaries, $\hat{u}$ is taken to be the velocity of the solid. At the free-surface $\hat{u} = u^+ - C_{11} n^+ p^+$. Both of these are equivalent to a particular definition of u^-, p^-, C_{11}, and C_{12} on the boundary.

We choose the parameters to be as simple as possible while achieving optimal convergence. The vector field C_{12} may be an arbitrary $\mathcal{O}(1)$ function. We bypass defining it, and set $\theta = \frac{1}{2}$ directly. The parameter C_{11} acts as a penalty parameter on discontinuities in the pressure. By using $C_{11} = \mathcal{O}(1)$, LDG requires no mesh-dependent parameters, but using $C_{11} = \mathcal{O}(h^{-1})$ achieves a better order of accuracy. We use $C_{11} = h^{-1}$.

Writing u and p in terms of coefficient vectors $\mathbf{u}$ and $\mathbf{p}$ for the velocity and pressure basis functions, and substituting into the integral equations (3), we arrive at an equivalent linear system $\mathbf{G}^\mathrm{T}\mathbf{u} - \mathbf{S}\mathbf{p} = \mathbf{0}$. The weak form of the momentum equation is discretized similarly and combined into the symmetric indefinite linear system

$$\begin{bmatrix} \mathbf{M} & \mathbf{G} \\ \mathbf{G}^\mathrm{T} & -\mathbf{S} \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} \mathbf{M}\tilde{\mathbf{u}} \\ \mathbf{0} \end{bmatrix} \tag{5}$$

where: $\frac{1}{2}\mathbf{u}^\mathrm{T}\mathbf{M}\mathbf{u}$ exactly equals the kinetic energy of the fluid, the right-hand-side $\tilde{\mathbf{u}}$ is found by interpolation from the FLIP particles (see §3.7), and any non-stationary solids would add another term on the right-hand-side.

3.5 Assembly and Linear Algebra

To assemble the system, we evaluate the integrals with exact high-order quadrature. In the interior of the domain, efficient Gaussian quadrature is available, and the grid structure allows a table of local integrals to be precomputed once, then added into the global matrix as needed. However, integration over cut cells must be performed every time step as their shapes change.

To integrate over a cut cell's volume, we clip the water mesh to the Cartesian grid cell, triangulate its boundary, then inexpensively tetrahedralize the cell by connecting each face to an arbitrary central point. By taking sign into account for the final exact integral, we allow overlapping and inverted tetrahedra. The final volume integral is a sum of exact quadratures applied to each tetrahedron, after which the tetrahedra are discarded and used for nothing else. The boundary integrals are similarly computed by a summation of quadratures over faces.

The volume integrals in $\mathbf{G}$ ($\mathbf{S}$ has no volume integral) can be easily derived from $\mathbf{M}$ by $\mathbf{MB}$, where $\mathbf{B}$ is a sum of several precomputed matrices that take the derivative of the pressure basis functions and apply a change of basis into the velocity basis. This is computed on a cell-by-cell basis, with small dense linear algebra operations.

To solve system (5), we eliminate $\mathbf{u}$ to get a symmetric positive definite system for the pressure only.

$$\mathbf{L}\mathbf{p} = (\mathbf{S} + \mathbf{G}^\mathrm{T}\mathbf{M}^{-1}\mathbf{G})\mathbf{p} = \mathbf{G}^\mathrm{T}\tilde{\mathbf{u}} \tag{6}$$

Because the basis functions in separate cells do not overlap, $\mathbf{M}$ is block diagonal and easy to invert directly. We construct $\mathbf{L}$ with

several dense matrix operations per cell and solve this linear system with conjugate gradient, preconditioned with block-wise zero-fill incomplete Cholesky. Substitution after solving system (6) gives $\mathbf{u} = \widetilde{\mathbf{u}} - \mathbf{M}^{-1}\mathbf{G}\mathbf{p}$.

3.6 Interpolating to the Particles

Once $\mathbf{u}$ and $\widetilde{\mathbf{u}}$ are known, we interpolate an update from the grid to the FLIP particles. The DG basis functions naturally define the interpolants $u(x)$ and $\widetilde{u}(x)$, and a linear operator $\mathbf{P}$ that evaluates these at all the particle locations. To update the particle velocities $\mathbf{v}_{\text{old}}$, we use Zhu and Bridson's PIC/FLIP mixing strategy [2005]:

$$\mathbf{v}_{\text{PIC}} = \mathbf{P}\mathbf{u} \tag{7}$$

$$\mathbf{v}_{\text{FLIP}} = \mathbf{v}_{\text{old}} + \mathbf{P}\mathbf{u} - \mathbf{P}\widetilde{\mathbf{u}} \tag{8}$$

$$\mathbf{v}_{\text{new}} = \theta\mathbf{v}_{\text{FLIP}} + (1-\theta)\mathbf{v}_{\text{PIC}} \tag{9}$$

We interpret this mixing of $\mathbf{v}_{\text{PIC}}$ and $\mathbf{v}_{\text{FLIP}}$ as the solution to a modified ODE that includes a decay of the (potentially) noisy particle velocities towards the noise-free grid velocities. For a decay with half-life λ, setting $\theta = 2^{-\Delta t/\lambda}$ achieves this decay in a stable fashion. We choose the half-life in terms of some characteristic time for the simulation, generally $\lambda = 0.5\,\text{s}$. This is not equivalent to an approximate viscosity, because the grid's velocity field is not a simple low-order approximation.

3.7 Interpolating from the Particles

In previous FLIP methods for computer graphics, the interpolation/approximation of the intermediate particle velocity $\widetilde{\mathbf{v}}$ to the grid is done by evaluating a weighted-average at the nodal points of the grid. This is inappropriate in this discretization because the nodal points have no particular meaning: they may be oddly distributed in space, outside the fluid, or with another basis construction nonexistent.

Instead, we find the coefficients $\widetilde{\mathbf{u}}$ that minimize the error when interpolating back to the particles, $\min \|\mathbf{P}\widetilde{\mathbf{u}} - \widetilde{\mathbf{v}}\|$. In cut cells that contain few particles, many solutions have zero residual. To make the problem well-posed, we use a regularization term to select smooth velocity fields.

$$\widetilde{\mathbf{u}} = \underset{\mathbf{u}}{\operatorname{argmin}} \left(\|\mathbf{P}\mathbf{u} - \widetilde{\mathbf{v}}\|_2^2 + \eta \sum_K \int_K \|\nabla u(x)\|_2^2 \right) \tag{10}$$

where $\eta = 0.1\,\text{m}^{-1}$ in all our simulations. The regularization also adds robustness against small amounts of noise in the particle data.

This global optimization problem separates into a small independent least-squares problem in each cell, since we use discontinuous approximation spaces which are independent in each cell, and the smoothness term is designed for separability by ignoring discontinuities across cell faces.

3.8 Particles and Advection

The fluid state consists of the FLIP particles (positions $\mathbf{r}$ and velocities $\mathbf{v}$), as well as the surface triangle mesh (positions, velocities, and connectivity). All the surface vertices are also FLIP particles, which is vital for capturing thin sheets and threads where the fluid volume is so small that there may be no particles in the interior. Every time step, we add and remove particles as necessary so that each cell has approximately twice as many particles as it has basis functions for each velocity component, scaled down by volume-fraction in cut cells, ensuring that particles and grid represent a similar level of detail.

When the liquid surface is advected, vertices that were touching the solid boundary before being advected are projected back onto the solid surface at the end of the timestep, consistent with the free-slip solid boundary conditions. To reduce collisions between the solid and liquid mesh, the liquid mesh is kept away from the solid by moving a small amount in the normal direction after this projection.

As in other FLIP solvers, the particles are advected using an ODE integrator through the static velocity field u computed on the grid. The simplest classical integrator that has stable behavior around vortices is RK3. Unfortunately, this requires 3 evaluations of the velocity field per step, which are expensive high-order interpolations from the grid. Furthermore, these evaluation points may be outside the fluid volume, requiring some sort of extrapolation.

We present a new technique that uses only a single evaluation of the velocity field. With a single lookup into the grid, we build the linear Taylor approximation to u at a particle's location r, using $u(r)$ and $\nabla u(r)$. The velocity field in each cell is at least a quadratic polynomial, so ∇u is always available. (In conventional solvers, a local finite difference could be used to estimate the gradient instead, typically using values that are already in cache.) This affine velocity field is enough to describe vortices and shear flow, so we use RK3 to advect through this approximation, and achieve the same linear stability properties as RK3 applied directly to u.

To demonstrate the advection scheme, we uniformly distribute particles in a 2D box, and advect them through the divergence-free velocity field with stream-function $\sin(x)\sin(y)$. This velocity field is a single vortex with pure rotation at its center, and pure sheer in the corners of the $[0,\pi]^2$ domain. Figure 5 shows the result of 100 timesteps ($\Delta t = 1.5$) with RK2, RK3, our approximate RK3, and the correct solution. Each integrator uses a different number of substeps, such that all approaches require the same number of evaluations of u. We experimented with other advection schemes using integrators of various accuracy applied to velocity approximations of various order, and found this combination to be an appropriate trade-off between speed and accuracy.

Tracing through the velocity field is usually also a significant expense for existing FLIP and semi-Lagrangian simulators, since each velocity evaluation requires an unstructured memory look-up and mixing of integer and floating-point pipelines: we believe our new technique should be of interest in accelerating other solvers too.

3.9 Surface Tracking

Surface tracking is handled by the El Topo library [Brochu and Bridson 2009], which produces quality triangulations as shown in Figure 6. We made some small some modifications to adapt El Topo to our use case.

Our discretization cannot handle arbitrarily thin sheets of liquid, so we prevented the surface tracker from creating them. We first tried deleting sheets of liquid that became too thin (e.g. $< 0.02h$), but this results in flicker at the tips of thin sheets as they are deleted. This is visible in our 2D videos, which use this strategy. We found better results in 3D by adding a repulsion force that tries to thicken sheets that are thinner than $0.05h$. Thickening a thin sheet can cause an increase in fluid volume and momentum, but our results indicate it is an acceptable trade-off. Either choice helps ensure good conditioning and behavior of the DG problem.

Some high-frequency surface features are handled poorly by our discretization and the surface tracker, so we use a small amount of Laplacian smoothing on the surface every step. As with PIC/FLIP smoothing, we can interpret this as a decay of each vertex towards the average position of its neighbours, with half-life between $0.1\,\text{s}$

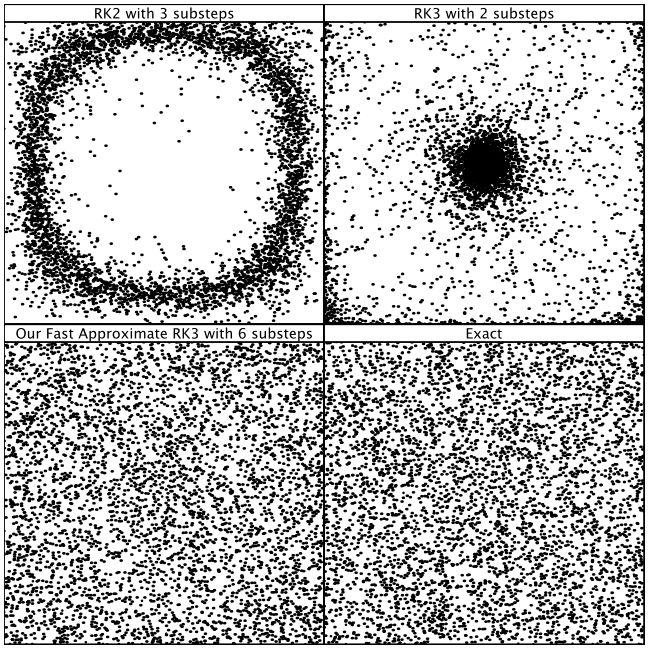

Figure 5: *Uniformly-distributed tracer particles are advected in a stationary vortex. Each integrator takes substeps such that all approaches evaluate the velocity field 6 times per timestep. After 100 timesteps, RK2 and RK3 both show significant clumping and divergence of particles.*

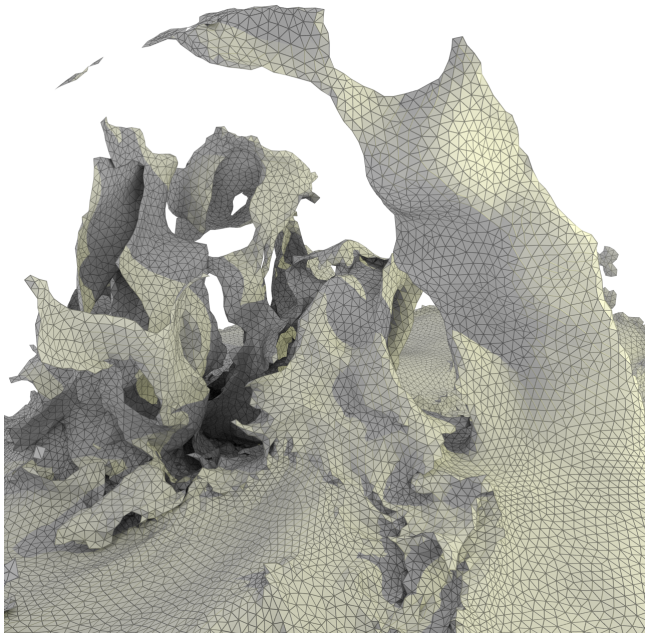

Figure 6: *A still from the simulation in Figure 1, showing the complex splashing geometry, and high-quality triangles from the explicit surface tracker.*

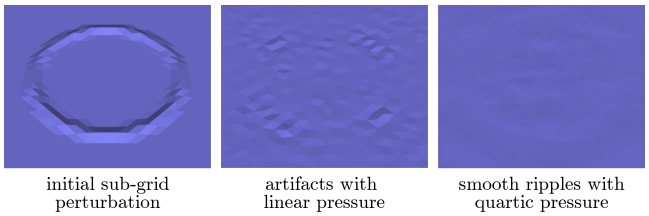

Figure 7: *Sub-grid features persist as non-physical bumps when using linear pressure, but are properly captured by physics when using quartic pressure, leaving only ripples in the surface. This view is approximately five grid cells wide.*

and $0.5\,\mathrm{s}$ for different examples. Smoothing is not applied at sharp convex features, determined by dihedral angles between triangles.

3.10 Viscosity

While our focus is inviscid flow, we sometimes found adding a small amount of viscosity improved the stability and visual appeal of the results. Viscosity is applied as a separate time-splitting step, after projection, to the grid's velocity field. We use an approximate implicit formulation of viscosity in which we apply the Galerkin discretization of the implicit backward-Euler viscous step in each cell, but solve each cell independently. This can increase the size of the discontinuities at cell boundaries, but not visibly so for the levels of viscosity in our examples. Figures 1 and 8 (bottom) used kinematic viscosity of $2 \cdot 10^{-2}$ and $2 \cdot 10^{-4}$ respectively, in non-dimensional units relative to h (length) and 1 second (time). Other examples use no viscosity.

4 Results

The discretization uses an extended model for pressure and velocity in a single cell. To experiment with the capabilities of this model, we ran a simulation in a single grid cell. This required no modifications to the method, just shorter edge lengths and more particles per cell than we typically use. Figure 3 shows a 1×1 simulation using sextic polynomials for pressure. Within the single cell, the method captures thin sheets, sloshing, and interactions with non-trivial solid boundaries. This example is, in essence, a very coarse spectral method, and emphasizes that our model is solving for full sub-grid physics, not applying a simple smoothing or more approximate fluid model to the surface.

The polynomial degree used in cut cells must be chosen in concert with mesh resolution. We examined the behavior as the polynomial degree changes by reproducing an experiment from [Brochu et al. 2010]. We simulated a still pool, with a subgrid disturbance $h/2$ wide and $h/10$ high, with surface mesh edge lengths of $\approx h/4$. Figure 7 shows that with low-order polynomials there are persistent artifacts, but for quartic pressure and higher, the bump is captured by the physics and ripples across the domain. Furthermore, simulation quality increases gradually as the polynomial degree is increased through intermediate degrees. This example uses no unphysical Laplacian smoothing in the normal direction.

Two controlled scenarios demonstrate the handling of thin sheets (Figure 8). Duplicating the experiment from [Thürey et al. 2010], we drop a ball of liquid onto a flat-topped pillar whence it expands into a thin sheet. As the sheet moves nearly ballistically through the air, small perturbations in its initial velocity amplify into a wavy shape. Second, we run a dam-break with a smooth obstacle less than $3h$ tall. The liquid forms a thin sheet as it spreads over and around the obstacle, following its curves and generating a detailed

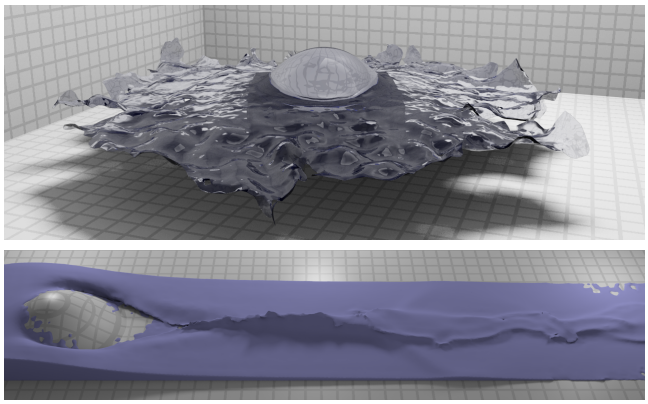

Figure 8: *Top: sheets move nearly ballistically after a splash. Bottom: thin sheets flow over and around a solid obstacle causing a complex wake. Grid resolution is shown on the solid floor. The obstacle is less than $3h$ tall.*

wake. In both scenarios, the thin sheets are still represented exactly in the discretization, due to the use of detailed cut cells.

To gain a sense of how practical the new approach is, we compared our code to a commercial FLIP solver, running the complex scenario from Figures 1 and 6. Figure 9 shows the results. Running on a 4-core 2.3GHz Intel Core i7 laptop with 4GB memory, the new DG solver took an average of 72 seconds per time step, with a 25^3 grid and sheets as thin as $1/48^{\text{th}}$ of a grid cell. Timings for different steps are given in Table 1. We ran the commercial FLIP solver (using a standard discretization and sparse, tiled voxels for efficiency) at two different resolutions, effectively 49^3 and 121^3. Both methods took up to three time steps per frame.

At 49^3 the commercial FLIP solver used approximately the same number of pressure variables as the DG code and was $14\times$ faster at 5.25s per time step, but of course gave far less detailed results: the solver could not resolve sheets thinner than a grid cell. At 121^3 the commercial FLIP solver took a similar compute time, at 43s per time step, and had a qualitatively similar perceived level of detail. However, the character of the detail is quite different. The DG simulation produces smooth, structured, and very thin features, with sheets approximately $1/500^{\text{th}}$ of the domain width in thickness - and even thinner in places. In contrast, the commercial FLIP code cannot reliably represent anything below a grid cell, $1/121^{\text{th}}$ of the domain width. The commercial FLIP simulation produces rough and splashy results in which the sheets break up into droplets, and cannot produce the smooth and extremely thin sheets of the DG results without drastically higher grid resolution.

In contrast to our research prototype, the commercial code is thoroughly optimized and multithreaded. Given the scope for continued performance and parallelism improvements in the new code, the ease of p-adaptivity, and the fully dynamic thin features that other solvers cannot capture nearly as efficiently, we believe this represents a very practical way forward for liquid animation.

5 Discussion

Interpreting the Method One simple interpretation of this method looks at it as another spatially-adaptive alternative to octrees or unstructured meshes. Relative to those methods, this approach has several advantages. First, it keeps the structure of the regular grid, significantly simplifying many parts of the algorithm. This approach also naturally produces anisotropic elements around

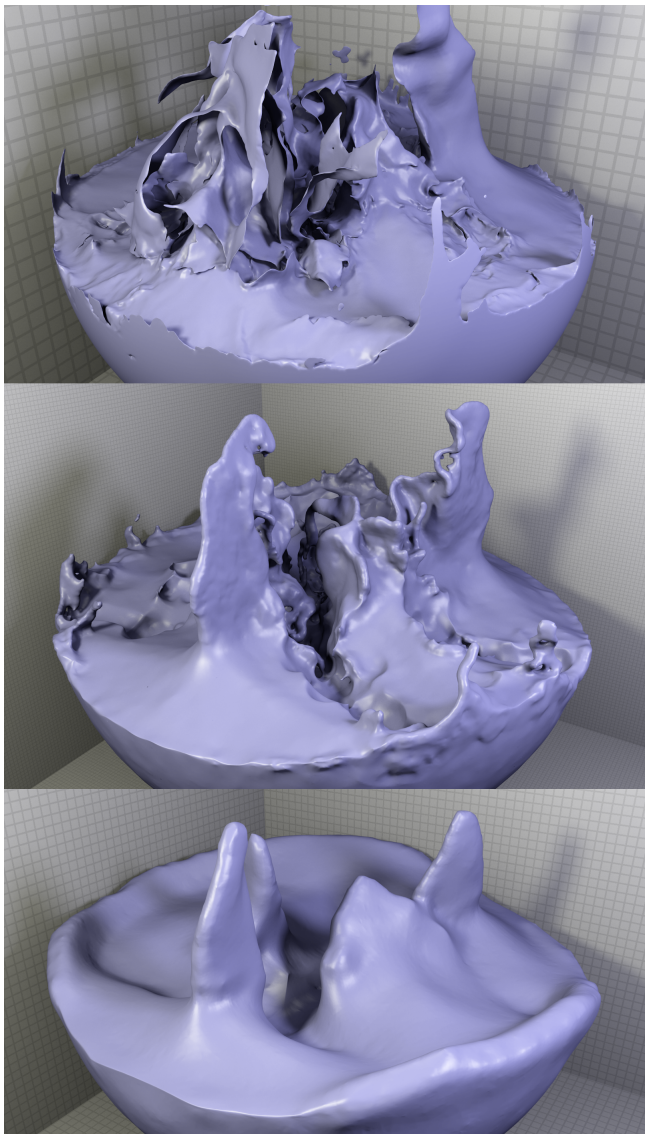

Figure 9: *The same simulation run with our method on a 25^3 grid (top), a commercial FLIP code on a 121^3 grid (middle), and the same commercial code on a 49^3 grid (bottom).*

Median Component Runtimes		(seconds/timestep)
Projection:	build volume mesh and u-p basis	2
	quadrature + assembly	15
	linear solver	6
Advection:	RK3, smoothing, collisions	2
Surface:	collision handling	38
	topology changes and remeshing	7
	rebuilding data-structures	5
Entire Step:		82
Median Variable Counts		**(number/timestep)**
Grid:	cut cells	1600
	regular cells	2800
	pressure variables	50800
Mesh:	surface vertices	205000

Table 1: *Detailed timings of the algorithm's components during the simulation from Figure 1, run on a laptop with an Intel i7-3610QM and 8GB of memory.*

thin fluid ligaments and sheets, providing significant reductions in problem size relative to the isotropic h-adaptation that is nearly universal. Furthermore, increasing p is more efficient for problems with smooth solutions. However, it is difficult to achieve large resolution differences between two areas of the simulation with just p-adaptivity, precisely because of imposed grid structure. Our p-adaptive approach would also work on general h-adapted meshes, complementing the h-adaptive techniques, and indeed would simplify octree-like methods with T-junctions since continuity between elements is not required. We are interested in seeing their combination into an hp-adaptive method.

Another over-simplified way to look at this method is as an approach for regularizing sub-grid features that arise when embedding a high-resolution surface into a low-resolution grid. Unlike the two-model approaches in §2, this approach resolves the velocity-pressure field to sub-grid features in a unified way and a single solve. Nevertheless, it cannot handle arbitrary amounts of sub-grid detail without extremely expensive and high-order bases. Combining these techniques may improve the result quality. The small amount of Laplacian smoothing we apply to the mesh can be interpreted in this way.

Both of the above interpretations try to cast this method in terms of the familiar, and consequently miss its novel behaviors and particular strengths. This approach is adaptive in a new way. In a sense, the velocity field has no limit to how small a feature can be, just as the distance between two extrema in a polynomial can be arbitrarily small. There is no free lunch, of course. The velocity field in a cell is limited to some finite dimensional space, and there is a limit on the total number and type of 'features' per grid cell. Still, the velocity field in a cell has the full approximating power of degree k polynomials, regardless of how small the liquid volume is within that cell. As a result, sub-grid thin sheets (and other features) can stretch, curl, and generally behave physically.

Thin Sheets This algorithm handles thin sheets remarkably well. We used sheets down to $0.02h$ thick, giving extremely anisotropic elements. We contrast our behavior here with the behavior of typical grid solvers. To represent the sheet using a level set on a standard MAC grid would require at least fifty times the grid resolution, i.e. greater than 2500 active degrees of freedom for a sheet that only uses 35 pressure variables in our approach. Methods using explicit surface tracking within a simple regular MAC grid could track the same geometry as our approach, but will still have difficulty producing an accurate velocity field in areas where sheets collide and produce complex topology within a single grid cell.

We chose a common physical model without surface tension, and without any effects from the 'air' outside the liquid domain. Physical experiments and theory identify that it is the velocity difference at the liquid-air interface that drives instability of sheets, and surface tension that drives retraction at their rims [Eggers 2011]. Without either of these effects, sheets and jets spread to be very thin without breaking up. This is in stark contrast to typical fluid simulations, in which discretization artifacts break up or delete fluid sheets, despite the fact that they have no physical model for that behavior. It's also in stark contrast to our every-day experience with splashes at small scale, which atomize almost instantly. Capturing instability-driven break-up in a physical way remains an interesting outstanding problem.

Discontinuities This algorithm embraces discontinuities at every opportunity, granting more flexibility and efficiency to the algorithms. Advection, interpolation between the grid and the particles, and the volume integrals all involve purely local data. The relevant data can be stored contiguously in memory, allowing efficient computation on modern architectures. Similarly, large parts of the computation can be expressed as dense matrix operations on data from a single cell (or pair of cells), and computed with highly-tuned dense BLAS and LAPACK routines.

Limitations Our method has some weaknesses when the velocity field is not very smooth. For example, a small collapsing air bubble requires a velocity field that quickly switches directions from one side of the bubble to the other, and these sometimes appear to have a 'numerical pressure' slowing their collapse. Another important case is when topological merges occur, introduce a nearly-discontinuous intermediate velocity, and cause large discontinuities in the discrete solution. The only time we saw visible discontinuities was in the first few frames after a droplet impacts a pool. They are imperceptible at regular playback speed. Despite these limitations, we still find the p-adaptive approach works well. These issues could, potentially, be resolved by using a different approximation space that includes velocity fields like these.

6 Conclusion

The combination of explicit surface tracking, detailed cut cells, and p-adaptive DG is an effective method for liquid simulation. They provide a qualitatively different set of performance characteristics than more common discretizations. In particular, they scale much better for smooth fields, and accurately capture thin sheets and features well below the grid scale. These techniques, applied together or independently, have lots of room for improvement and application in this and other areas.

Acknowledgments

We thank the anonymous reviewers for their useful suggestions, and the Natural Sciences and Engineering Research Council of Canada for supporting this research.

References

ADAMS, B., PAULY, M., KEISER, R., AND GUIBAS, L. J. 2007. Adaptively sampled particle fluids. *ACM Trans. Graph. 26*, 3 (July).

ANDO, R., THÜREY, N., AND TSURUNO, R. 2012. Preserving fluid sheets with adaptively sampled anisotropic particles. *IEEE Transactions on Visualization and Computer Graphics 18*, 8, 1202–1214.

ANDO, R., THÜREY, N., AND WOJTAN, C. 2013. Highly adaptive liquid simulations on tetrahedral meshes. *ACM Trans. Graph. (Proc. SIGGRAPH 2013)* (July).

ARNOLD, D., BREZZI, F., COCKBURN, B., AND MARINI, L. 2002. Unified analysis of discontinuous Galerkin methods for elliptic problems. *SIAM journal on numerical analysis 39*, 5, 1749–1779.

BABUSKA, I., AND GUO, B. 1992. The h, p and hp version of the finite element method basic theory and applications. *NASA STI/Recon Technical Report N 93*, 22550.

BATTY, C., XENOS, S., AND HOUSTON, B. 2010. Tetrahedral embedded boundary methods for accurate and flexible adaptive fluids. In *Proceedings of Eurographics*.

BOJSEN-HANSEN, M., AND WOJTAN, C. 2013. Liquid surface tracking with error compensation. *ACM Transactions on Graphics (SIGGRAPH 2013) 32*, 4, 79:1–79:10.

BOYD, J. P. 2001. *Chebyshev and Fourier spectral methods*. Courier Dover Publications.

BRIDSON, R. 2008. *Fluid Simulation for Computer Graphics*. A K Peters/CRC Press, Sept.

BROCHU, T., AND BRIDSON, R. 2009. Robust topological operations for dynamic explicit surfaces. *SIAM J. Sci. Comput. 31*, 4, 2472–2493.

BROCHU, T., BATTY, C., AND BRIDSON, R. 2010. Matching fluid simulation elements to surface geometry and topology. *ACM Trans. Graph. 29*, 4, 1–9.

CASTILLO, P., COCKBURN, B., PERUGIA, I., AND SCHÖTZAU, D. 2000. An a priori error analysis of the local discontinuous Galerkin method for elliptic problems. *SIAM Journal on Numerical Analysis 38*, 5, 1676–1706.

CASTILLO, P. 2006. A review of the local discontinuous Galerkin (LDG) method applied to elliptic problems. *Applied numerical mathematics 56*, 10, 1307–1313.

CHENTANEZ, N., AND MÜLLER, M. 2011. Real-time eulerian water simulation using a restricted tall cell grid. In *ACM SIGGRAPH 2011 Papers*, ACM, New York, NY, USA, SIGGRAPH '11, 82:1–82:10.

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, Aire-la-Ville, Switzerland, Switzerland, SCA '07, 219–228.

COCKBURN, B., KANSCHAT, G., AND SCHÖTZAU, D. 2005. The local discontinuous Galerkin method for linearized incompressible fluid flow: a review. *Computers & fluids 34*, 4, 491–506.

EDWARDS, E., AND BRIDSON, R. 2012. A high-order accurate particle-in-cell method. *International Journal for Numerical Methods in Engineering 90*, 9, 1073–1088.

EGGERS, J. 2011. The subtle dynamics of liquid sheets. *Journal of Fluid Mechanics 672*, 1.

ENGLISH, R. E., QIU, L., YU, Y., AND FEDKIW, R. 2013. Chimera grids for water simulation. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ACM, New York, NY, USA, SCA '13, 85–94.

ENRIGHT, D., FEDKIW, R., FERZIGER, J., AND MITCHELL, I. 2002. A hybrid particle level set method for improved interface capturing. *Journal of Computational Physics 183*, 1, 83–116.

FOSTER, N., AND METAXAS, D. 1996. Realistic animation of liquids. *Graphical models and image processing 58*, 5, 471–483.

IRVING, G., GUENDELMAN, E., LOSASSO, F., AND FEDKIW, R. 2006. Efficient simulation of large bodies of water by coupling two and three dimensional techniques. *ACM Trans. Graph. (Proc. SIGGRAPH) 25*, 3, 805–811.

KARNIADAKIS, G. E., AND SHERWIN, S. J. 1999. *Spectral/hp element methods for CFD*. Oxford University Press.

KAUFMANN, P., MARTIN, S., BOTSCH, M., AND GROSS, M. 2009. Flexible simulation of deformable models using Discontinuous Galerkin FEM. *Graphical Models 71*, 4, 153–167. Special Issue of ACM SIGGRAPH / Eurographics Symp. Comp. Anim. 2008.

LOSASSO, F., GIBOU, F., AND FEDKIW, R. 2004. Simulating water and smoke with an octree data structure. In *ACM Transactions on Graphics (TOG)*, vol. 23, ACM, 457–462.

MISZTAL, M., AND BÆRENTZEN, J. 2012. Topology adaptive interface tracking using the deformable simplicial complex. *ACM Transactions on Graphics 31*, 3.

MISZTAL, M., BRIDSON, R., ERLEBEN, K., BÆRENTZEN, J., AND ANTON, F. 2010. Optimization-based fluid simulation on unstructured meshes. In *Proceedings of the 7th Workshop on Virtual Reality Interaction and Physical Simulation (VRIPHYS 2010)*, SIGGRAPH.

MISZTAL, M., ERLEBEN, K., BARGTEIL, A., FURSUND, J., CHRISTENSEN, B., BÆRENTZEN, J., AND BRIDSON, R. 2012. Multiphase flow of immiscible fluids on unstructured moving meshes. In *Eurographics/ACM SIGGRAPH Symposium on Computer Animation*, The Eurographics Association, 97–106.

MORESI, L., DUFOUR, F., AND MHLHAUS, H.-B. 2003. A lagrangian integration point finite element method for large deformation modeling of viscoelastic geomaterials. *Journal of Computational Physics 184*, 2, 476 – 497.

MÜLLER, M. 2009. Fast and robust tracking of fluid surfaces. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ACM, New York, NY, USA, SCA '09, 237–245.

OSHER, S., AND FEDKIW, R. 2003. *Level set methods and dynamic implicit surfaces*, vol. 153. Springer.

QIN, R., AND KRIVODONOVA, L. 2012. A discontinuous Galerkin method for solutions of the Euler equations on Cartesian grids with embedded geometries. *Journal of Computational Science*.

SCHWAB, C. 1998. *p-and hp-finite element methods: Theory and applications in solid and fluid mechanics*. Clarendon Press Oxford.

SOLENTHALER, B., AND GROSS, M. 2011. Two-scale particle simulation. *ACM Trans. Graph. 30*, 4 (July), 81:1–81:8.

THÜREY, N., WOJTAN, C., GROSS, M., AND TURK, G. 2010. A multiscale approach to mesh-based surface tension flows. In *ACM SIGGRAPH 2010 Papers*, ACM, New York, NY, USA, SIGGRAPH '10, SIGGRAPH, 48:1–48:10.

WOJTAN, C., AND TURK, G. 2008. Fast viscoelastic behavior with thin features. *ACM Trans. Graph. 27*, 3, 1–8.

WOJTAN, C., THÜREY, N., GROSS, M., AND TURK, G. 2009. Deforming meshes that split and merge. In *ACM Transactions on Graphics (TOG)*, vol. 28, ACM, 76.

WOJTAN, C., THÜREY, N., GROSS, M., AND TURK, G. 2010. Physics-inspired topology changes for thin fluid features. *ACM Trans. Graph. 29*, 4, 1–8.

WOJTAN, C., MÜLLER-FISCHER, M., AND BROCHU, T. 2011. Liquid simulation with mesh-based surface tracking. In *ACM SIGGRAPH 2011 Courses*, ACM, New York, NY, USA, SIGGRAPH '11, ACM, 8:1–8:84.

ZHU, Y., AND BRIDSON, R. 2005. Animating sand as a fluid. In *ACM SIGGRAPH 2005 Papers*, ACM, SIGGRAPH, 965–972.