Simulating Liquids on Dynamically Warping Grids

Hikaru Ibayashi, Chris Wojtan, Nils Thuerey, Takeo Igarashi and Ryoichi Ando

Abstract—We introduce dynamically warping grids for adaptive liquid simulation. Our primary contributions are a strategy for dynamically deforming regular grids over the course of a simulation and a method for efficiently utilizing these deforming grids for liquid simulation. Prior work has shown that unstructured grids are very effective for adaptive fluid simulations. However, unstructured grids often lead to complicated implementations and a poor cache hit rate due to inconsistent memory access. Regular grids, on the other hand, provide a fast, fixed memory access pattern and straightforward implementation. Our method combines the advantages of both: we leverage the simplicity of regular grids while still achieving practical and controllable spatial adaptivity. We demonstrate that our method enables adaptive simulations that are fast, flexible, and robust to null-space issues. At the same time, our method is simple to implement and takes advantage of existing highly-tuned algorithms.

Index Terms—Computer Graphics, Physics-based Animation, Fluid Simulation, Liquid, Adaptivity, Curvilinear Grids.

1 Introduction

LIQUID simulations for computer graphics have long dazzled audiences with their ability to reproduce visually intricate physical features like vibrant water splashes and subtle ripples. Previous researchers have endeavored to reproduce these effects by designing algorithms specific to each phenomenon. Examples of this strategy include thin sheets [1], [2], water droplets [3], foams [4], [5], [6], underwater bubbles [7], [8] and capillary waves [9], [10].

An alternative to developing algorithms for specific phenomena is to use spatially adaptive simulation. Unstructured grids have been the main ingredient to achieve this goal. Examples of methods employing unstructured grids include octrees [11], [12], tetrahedral grids [13], [14], [15], [16], and Voronoi diagrams [17], [18], [19]. A central benefit of adaptive simulations is the capability of simulating large-scale phenomena at a feasible computational cost [12], [20]. The main idea of our work is to enhance existing regular grid algorithms to deliver dynamic spatial adaptivity without excessive computational overhead. Common strategies for simulating with unstructured meshes come with significant complications, namely inefficient memory access and implementation complexity due to non-trivial adaptive mesh refinement. Meanwhile, current adaptive methods based on structured grids only offer limited types of deformation. In this context, we propose a novel adaptive method with dynamically warping grids, which includes four technical contributions:

- **Deformation solver** We present a new method to adaptively warp a Cartesian grid by iteratively solving a nonlinear system. Our deformation solver quickly adapts the grids to the specific region of interest while taking into account temporal coherence (Section 4).
- **Advection Solver Compatible with existing algorithms** Our advection method is able to reuse off-the-shelf algorithms like a narrow-band FLIP solver [21], the MacCormack method [22], and sixth-order Weighted Essentially Non-Oscillatory (WENO) interpolation [23] to advect the velocity and the surface geometry (Section 5).
- **Pressure solver on deforming grids** We introduce a new pressure solver specifically designed for simulating liquid with a deforming regular grid. Our solver is straightforward to implement and avoids the null-space issues typically associated with hexahedral mesh solvers. (Section 6.1)
- **Generalized Ghost Fluid method** We propose a new formulation to accurately treat free surface boundary conditions extending the ghost fluid method [24]. We show that our approach delivers second-order accuracy on our warped grids and the technique generalizes to arbitrary discretizations (Section 6.2).

2 Related Work

Our work is based on Eulerian fluid simulations. Eulerian fluid animation has flourished since the introduction of unconditionally stable semi-Lagrangian advection by Stam [25]. Afterward, Foster and Fedkiw [26] enforced free-surface boundary conditions to simulate liquids. Since then, Marker-And-Cell (MAC) grids combined with the level-set

- *H. Ibayashi is with the Department of Computer Science, the University of Southern California, Los Angels, CA, 90007. E-mail:ibayashi@usc.edu*
- *C. Wojtan is with the Visual Computing Group, the Institute of Science and Technology Austria, Am Campus 1, 3400 Klosterneuburg, Austria. E-mail:wojtan@ist.ac.at*
- *N. Thuerey is with the the Technische Universität München, the Arcisstraße 21, 80333 München, Germany. E-mail:nils.thuerey@tum.de*
- *T. Igarashi is with the Department of Computer Science, the University of Tokyo, Hongo, Bunkyo-ku, Tokyo, 113-8654, Japan. E-mail:takeo@acm.org*
- *R. Ando is with the National Institute of Informatics, Chiyoda-ku, Tokyo 101-8430, Japan. E-mail:rand@nii.ac.jp*

Manuscript received April 19, 2005; revised August 26, 2015.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2018.2883628, IEEE Transactions on Visualization and Computer Graphics

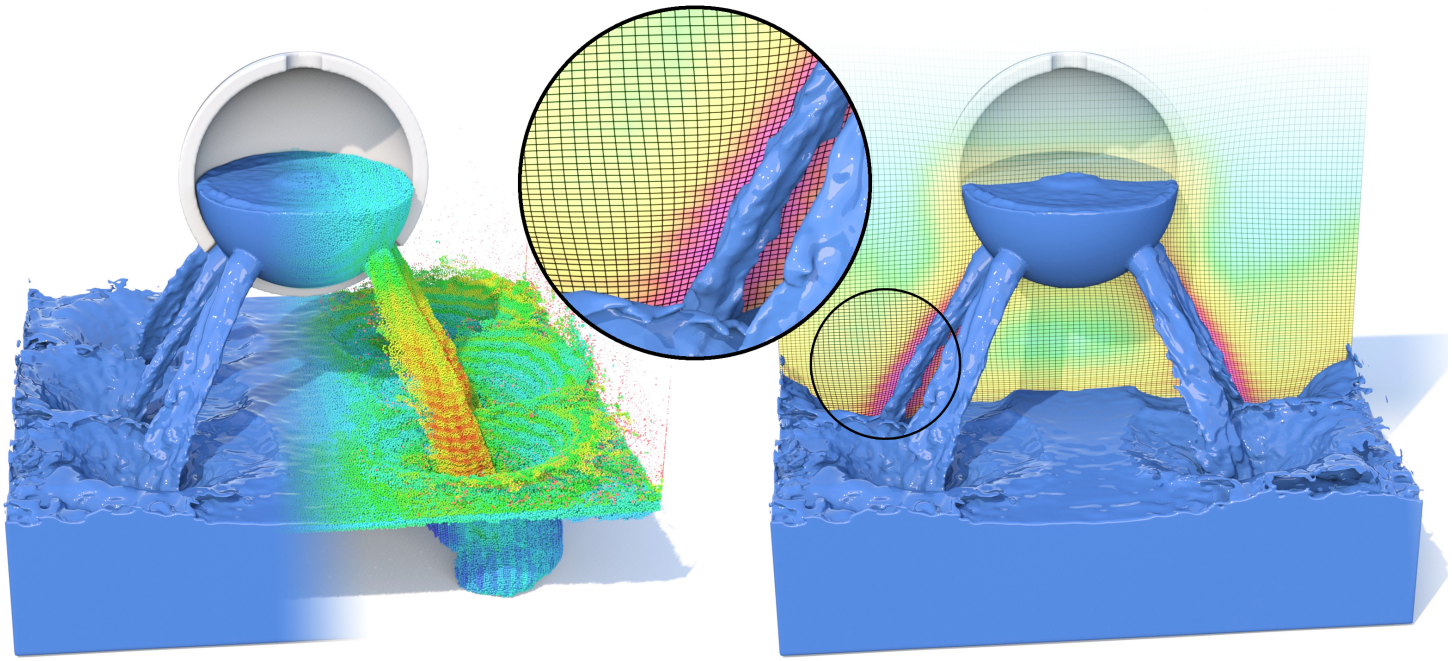


Fig. 1. Liquid leaking out of a container with holes. Our dynamically warping grids adaptively capture detailed motions without the computational expense of unstructured grids. The left image depicts the surface mesh and the corresponding FLIP particles color-coded by the grid cell size, while the right image shows a cross-section. The simulation used 128^3 cells and ran with 25 seconds per time step, and 1.8 minutes per video frame.

method have become the standard method for grid-based liquid simulation. MAC grids were originally introduced to graphics by Harlow and Welch [27] and to computer graphics by Foster and Metaxas [28]. Since then, researchers have continually extended them to support second-order accurate boundary conditions for free surfaces [24], and to handle two-way coupled rigid bodies [29]. Interested readers are referred to the book by Bridson [30] for an overview of the state-of-the-art in fluid animation. Because we target adaptivity, the remainder of this section focuses on methods related to adaptive simulation.

2.1 Arbitrary Lagrangian-Eulerian method

Our method is similar to the arbitrary Lagrangian-Eulerian (ALE) method [31]. ALE method combines a Lagrangian method with an Eulerian method in the sense that it uses grids like Eulerian methods, but those grids move with liquid like Lagrangian methods. Several works in computer graphics also make use of ALE methods, such as translating grids [32], [33] or adaptively refining grids [34]. We refer interested readers to Donea et al. [35] for a comprehensive overview of ALE methods. To the best of our knowledge, our work is the first to computer graphics that applies the ALE method for deforming uniform grids.

2.2 Unstructured meshes

Tetrahedral meshes have been widely used for adaptive simulations [13], [15], [16], [20], [36]. Working with tetrahedral meshes requires high-quality mesh generation, an active area of research [37], [38]. The cost of accessing an element typically requires a tree traversal or hash table lookup, which substantially lowers the cache hit rate. Researchers have also used octree grids. Losasso et al. [11] and Setaluri et al. [39] employed the Finite Volume Method on an octree grid to discretize pressure and velocity similarly to the MAC discretization. Nielsen and Bridson [40] and Ferstl et al. [12] discretized them with the Finite Element Method, where the pressure and the velocity are collocated. Some have also investigated adaptive simplicial complexes [41], [42] and Voronoi-based approaches with a particle-based pressure solver [18], [19]. Sparse grid methods [39], [43] can overcome many of these problems with slow access times. Recently, Aanjaneya et al. [44] demonstrated that liquid simulations are possible with a sparse grid discretization based on power-diagrams. However, the necessity of such specialized discretizations at the free surface illustrates the difficulties that commonly arise for adaptive methods. Our generalized boundary conditions in conjunction with the warped regular grids circumvent such problems.

2.3 Structured grids

Curvilinear coordinates also bring adaptivity to structured grids. The use of curvilinear coordinates dates back to the original work of FLIP [45] and was introduced to graphics by Azevedo and Oliveira [46]. Aside from curvilinear grids, Zhu et al. [47] proposed to use a new extended grid structure where grids were stretched horizontally or vertically. Both of the methods used the Finite Volume Method to discretize the pressure. Our work is related to their methods in the sense that we also use stretched regular grids. However, we further allow such grids to dynamically warp over time, which gives more flexible adaptivity. In the context of structured grids, Irving et al. [48] proposed *tall cells* for liquid simulations, which vertically coarsened the

grids away from the free surface, and Chentanez and Müller [49] extended them to real-time applications.

2.4 Image warping

Image and video processing algorithms have similarly investigated deforming regular grids to adapt the image content to various constraints. For still images, researchers have demonstrated that feature-aware deformations can help to preserve salient regions of an image [50]. Non-linear deformations of video streams were likewise proposed to robustly re-scale video content to different devices [51], or to change the disparity of stereoscopic videos [52]. While these approaches share our goal to achieve flexible deformations with regular grids, the target applications impose very different constraints. Image deformation algorithms usually assign a desired shape and size to each pixel. Because the target deformation is fixed to each pixel, the problem can often be reduced to a linear system. In contrast, adaptive refinement algorithms are usually given a desired volume *as a function of space*, so moving an element will, in fact, change its desired volume. Because this is inherently a moving-target problem, it inevitably results in a *non-linear* system. To cope with this added difficulty, we propose a solution which reduces the solution space in a problem-specific manner and propose problem-specific speed-ups.

2.5 Hybrid approaches

Several researchers focused on combining Cartesian grids with unstructured grids. For example, the methods of Dobashi et al. [53] and Azevedo and Oliveira [46] developed overlapping grids to cover complex regions. Both methods used Cartesian grids of different resolutions that are translated and rotated. English et al. [54] introduced *Chimera grids*, which decompose the simulation domain into multiple regions of interest. These regions are individually discretized with different Cartesian grids and later combined into a single linear system. Some researchers exploited irregular meshes to capture detailed boundaries. Feldman et al. [14] adapted irregular tetrahedra to the curved boundaries of solids, while Brochu et al. [17] devised Voronoi-based meshes to accurately capture the geometry and the topology of free surfaces. These hybrid approaches improve the average cache hit rate, and our method likewise benefits from fast memory accesses provided by regular grids. However, in contrast to many of these approaches, we do not employ any cells with irregular connectivity. Our grids purely consist of warped cubes, so many tasks like the implementation and stability analyses are simpler.

2.6 Mesh-free methods

Adaptive Smoothed-Particle Hydrodynamics (SPH) methods are also studied in the engineering literature [55], [56] as well as interactive applications [57]. Adams et al. [58] proposed to dynamically subdivide SPH particles, while Solenthaler and Gross [59] used a hierarchy model to circumvent the numerical issues introduced by particle splitting and merging. Particle representations are especially well suited for splashes, and our FLIP-based solver also employs particles to represent small structures.

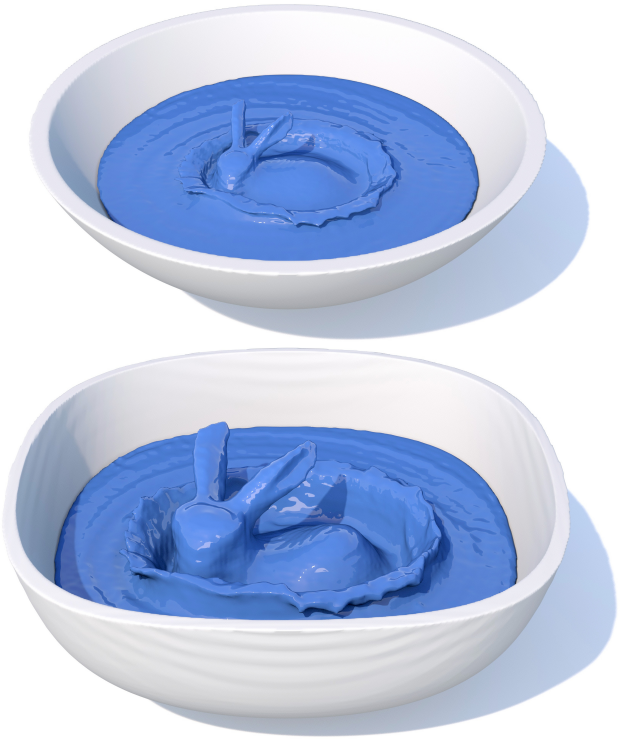


Fig. 2. Merging bunny: the top image shows the spatial (physical) coordinate and the bottom corresponding reference coordinate at the same timings, illustrating the magnification of local feature. $128\times64\times128$ resolution, 10 seconds per time step and 29 seconds per video frame.

2.7 Polynomial and spectral adaptivity

Some researchers designed specific basis functions for increasing efficiency. Treuille et al. [60] extracted a representative basis by performing Principal Component Analysis (PCA) on a set of training examples. De Witt et al. [61] used a Laplacian eigenfunction basis and derived how the basis coefficients change over time. Edwards and Bridson [62] used the Discontinuous Galerkin method to simulate detailed liquids on very coarse grids by representing pressure with a high-order polynomial basis. This class of methods is known as *p*-adaptivity, and our warping grids framework could benefit from combining it with such high-order bases.

3 Method Overview and Definitions

We solve the Navier-Stokes (NS) equations by an operator splitting approach [30]. Our method consists of three independent building blocks: a grid deformation solver for spatial adaptivity (Section 4), the advection of velocity and surface geometry (Section 5), and a pressure solver for warped grids (Section 6). We note that the majority of our results use a narrowband FLIP solver [21] for additional efficiency. We outline our approach in Algorithm 1, and we describe each of the components in more detail in the following sections.

Before we start discussing our algorithm, we would like to define our terminology, which follows ALE practice [35]. We refer to coordinates in the physical space as "spatial coordinates". Quantities defined in this coordinate can be directly visualized (Figure 2 top). "Reference coordinate",

Algorithm 1: Simulation Loop

input : Velocity field and the level-set $\boldsymbol{u}_t$ and ϕ_t.
output: New velocity field and the level-set $\boldsymbol{u}_{t+\Delta t}$ and $\phi_{t+\Delta t}$.

1 Solve the grid deformation and get the mesh velocity $\boldsymbol{u}_D$ (Section 4)
2 Convert $\boldsymbol{u}_t$ and $\boldsymbol{u}_D$ to the reference coordinate $\bar{\boldsymbol{u}}_t$ and $\bar{\boldsymbol{u}}_D$
3 Advect $\boldsymbol{u}_t^* \leftarrow \mathrm{Adv}(\boldsymbol{u}_t)$ through $\bar{\boldsymbol{u}}_t - \bar{\boldsymbol{u}}_D$ on the reference coordinate (Section 5)
4 Advect $\phi_t^* \leftarrow \mathrm{Adv}(\phi_t)$ through $\bar{\boldsymbol{u}}_t - \bar{\boldsymbol{u}}_D$ on the reference coordinate (Section 5)
5 Add force to $\boldsymbol{u}_t^*$
6 Project $\boldsymbol{u}_{t+\Delta t} \leftarrow \mathrm{Proj}(\boldsymbol{u}_t^*)$ (Section 6)
7 Re-distance $\phi_{t+\Delta t} \leftarrow \mathrm{Redist}(\phi_t)$

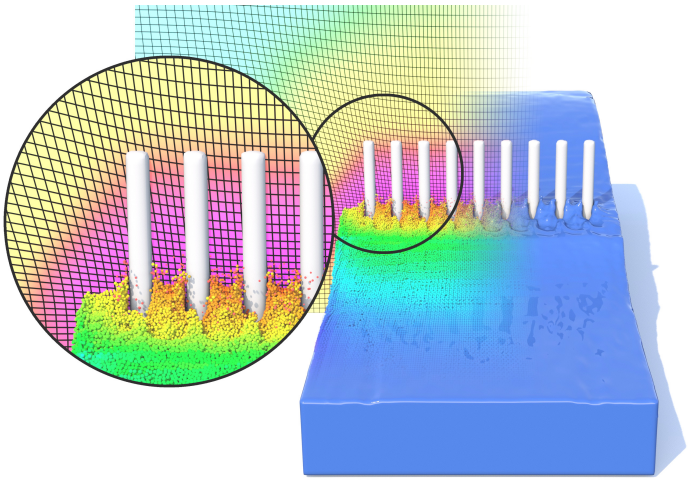


Fig. 3. Combing a liquid: the left side of the zoomed image highlights our warped grids adapting near the solid boundaries, making the accurate interaction with cylinders possible.$128 \times 64 \times 128$ resolution, 12 seconds per time step and 30 seconds per video frame.

on the other hand, denotes the coordinate on uniform grids (Figure 2 bottom). We exploit reference coordinates for the convenience of calculation, but quantities in this coordinate have no direct physical correspondences. *Deformation solver* is the operation that produces a vector field that we use to deform a uniform grid. "Mesh velocity" denotes the change of this vector field over time.

4 Deformation Solver

The problem of deforming a grid has been revisited many times throughout the image- and geometry-processing literature. The most efficient of these algorithms address the problem by assigning a desired deformation to each element and then deforming each element to optimally match the target. Because the target deformation is *fixed to each element*, the problems can often be reduced to a linear system. However, in our application of adaptive refinement, the desired deformation depends on interesting physical behaviors, which will inevitably be a function of *physical space*. Thus, the target deformations of the elements will change as the grid deforms, and the system is fundamentally non-linear. To cope with this difficulty, we exploit the structure of our particular problem in order to both reduce the dimension of our solution space (Section 4.1) and devise numerical acceleration strategies (Section 4.4).

4.1 Defining the problem

We first note that our deformation aims to control *element volumes* instead of completely general deformations. We can exploit this to reduce the solution space of our deformations from a 3-dimensional displacement vector field down to a scalar field. To do this, we first note that the volume of an element can be computed as the cell's rest volume added to divergence of the deformation field integrated over the cell, according to the divergence theorem. Then we note that the divergent part of a vector field is uniquely described by the gradient of a scalar field, according to the Helmholtz decomposition. Thus, we can remove all deformations that are irrelevant to our sizing function by restricting each element's deformation to $\boldsymbol{x}(\xi, \varphi) = \boldsymbol{\xi} + \nabla_\xi \varphi$, where $\boldsymbol{\xi}$ is the reference coordinate and φ is a scalar field containing all of the degrees of freedom in the deformation. In addition, to encode the region of interest, i.e. the region a user wants to emphasize, our solver is fed a user-defined scalar field, which we call *sizing function* or $f_{\text{size}}(\boldsymbol{x})$. *Sizing function* can be interpreted as a field of sinks in the spatial coordinate. By putting a large sink at a specific area, a user can make nodes concentrated and obtain a fine grids there. Therefore, we obtain φ via the following non-linear Poisson equation:

$$\nabla_\xi^2 \varphi(\boldsymbol{\xi}) = f_{\text{size}}\Big(\boldsymbol{x}(\boldsymbol{\xi}, \varphi)\Big). \tag{1}$$

Each time step, we solve for the scalar field φ and then deform the grid according to $\boldsymbol{x}(\xi, \varphi)$. We store φ at cell centers, and discretize the Laplacian operator by the seven-point Laplacian matrix. We evaluate $\nabla_\xi \varphi$ on vertices and use it to displace the vertices. This nodal staggered discretization naturally avoids the null-space problems that can occur on collocated discretizations. We solve Eq. (1) by iteratively solving a linearized version of the equation with φ on the right hand side fixed to the value produced by the previous iteration. Each iteration imposes Neumann boundary conditions $\boldsymbol{n}_\Omega \cdot \nabla_\xi \varphi = 0$, where $\boldsymbol{n}_\Omega$ is the boundary of our grid (not the boundary of our simulation domain). To make sure a solution exists in each iteration, we re-normalize the right hand side by first offsetting it with a constant γ such that $\int_\Omega (f_{\text{size}}(\boldsymbol{x}(\boldsymbol{\xi}, \varphi)) + \gamma)\, dV_\xi = 0$. Then, to ensure a numerically stable scaling, we divide the vector on the right hand side by its minimum value.

4.2 Sizing Function

As we described in Section 4.1, we propose a sizing function that is defined by users to emphasize the region of interest. We would like to note that although it gives us the intuition of the sink field, our sizing function is heuristically designed, and is not intended to mimic any particular physical quantity in the real world. We adopted a criteria of using the liquid surface and the region of large velocity as the region of interest. There could be various ways to define a sizing

function but we found that it becomes quite controllable when formulated as a sum of exponential functions:

$$f_{\text{size}}(\boldsymbol{x}) = -\exp\Big(-\alpha_\phi\,\hat{\phi}\Big) + \exp\Big(-\beta_u\,||\boldsymbol{u}||\Big), \tag{2}$$

where $\hat{\phi}$ denotes the distance from the free surface and $||\boldsymbol{u}||$ denotes the magnitude of the velocity at point $\boldsymbol{x}$. The relative importance of these two terms is weighted by user-weighted parameters α_ϕ and β_u. All of our examples use the values $\alpha_\phi = 7$ and $\beta_u = 0.5$, except for Figure 6 where we have used a static warped grid generated from the initial fluid condition with the same parameters. In case a user wishes to emphasize another region of interest, any additional term can be similarly added to Eq. (2). For example, for a scene with detailed boundary interactions (Figure 3), we add an additional term $-\exp\Big(-\gamma_\psi\,\hat{\psi}\Big)$, where $\hat{\psi}$ is the distance from the solid boundary, and we set $\gamma_\psi = 7$. We evaluate the sizing function at cell centers and apply a small amount of blurring to ensure smoothness.

4.3 Temporal Deformation Penalty

Depending on the sizing function f_{size}, the solution to Eq. (1) can produce temporally flickering deformations. We introduce the following energy function to penalize the drastic motion or acceleration of our displacement field:

$$P_{\text{temporal}} = \int_\Omega \frac{1}{2}\rho_1 \Big|\Big|\frac{\partial \nabla_\xi \varphi}{\partial t}\Big|\Big|^2 + \frac{1}{2}\rho_2 \Big|\Big|\frac{\partial^2 \nabla_\xi \varphi}{\partial t^2}\Big|\Big|^2 dV_\xi. \tag{3}$$

where ρ_1 and ρ_2 are tunable control parameters. Discretizing the time derivatives with implicit Euler and minimizing Eq. (3) with respect to φ_{t+1} gives us the following equation:

$$\Big(\frac{\rho_1}{\Delta t^2} + \frac{\rho_2}{\Delta t^4}\Big)\nabla_\xi^2 \varphi_{t+1} = \Big(\frac{\rho_1}{\Delta t^2} + \frac{2\rho_2}{\Delta t^4}\Big)\nabla_\xi^2 \varphi_t - \frac{\rho_2}{\Delta t^4}\nabla_\xi^2 \varphi_{t-1}, \tag{4}$$

where φ_{t-1}, φ_t, and φ_{t+1} denote the value of φ at the previous, current, and next time step. Finally, we add Eq. (4) to Eq. (1) to get an equation for φ that trades off between temporal coherence and respecting the sizing function:

$$\begin{aligned}\Big(1 + \frac{\rho_1}{\Delta t^2} + \frac{\rho_2}{\Delta t^4}\Big)\nabla_\xi^2 \varphi = f_{\text{size}}\Big(\boldsymbol{x}(\boldsymbol{\xi}, \varphi)\Big) + \\ \Big(\frac{\rho_1}{\Delta t^2} + \frac{2\rho_2}{\Delta t^4}\Big)\nabla_\xi^2 \varphi_t - \frac{\rho_2}{\Delta t^4}\nabla_\xi^2 \varphi_{t-1}.\end{aligned} \tag{5}$$

We set $\rho_1 = 2\Delta t^2$ and $\rho_2 = \Delta t^4$ throughout our examples. Once again, we re-normalize the right-hand side of the equation to ensure stability.

4.4 Speeding-up the Solver

Repeatedly solving the linear system in Eq. (5) can be expensive. We accelerate our solver by exploiting the fact that the left-hand side of our linear system remains the same throughout our simulation, allowing us to pre-compute a preconditioner at the beginning of a simulation. In our work, we have employed an Algebraic Multigrid (AMG) method provided by Demidov [63] as a preconditioner. For coarsening and smoothing operators for AMG, we used the Smoothed Aggregation technique and a Gauss-Seidel smoother provided with the library. Our method benefits from such a multigrid method since the domain of our deformation grids does not involve any irregular boundaries.

Unlike the pressure solver, our deformation solver does not need to reach an accurate solution in practice. In our case, we stop the iteration when the maximal change per iteration is less than the half of a grid cell size. Furthermore, at each iteration we only perform three steps of the Biconjugate gradient stabilized method (BiCGSTAB), and re-evaluate the right-hand side of Eq. (5), which we found to converge quickly and stably.

We solve the deformation by first constructing a multi-resolution pyramid by repeatedly down-sampling grids by a factor of two. We start with performing our deformation solver from the coarsest resolution, upsample the solution and continue to the next higher resolution. In our examples, we solve only up to one level coarser resolution than the finest resolution. The deformation at the finest resolution is computed simply by subdividing the grids from the previous level. To prevent each solver iteration from overshooting the solution, and thus leading to a poor convergence rate, we damp each update by fifty percent .

5 Advection

We follow an ALE formulation of the form [35]:

$$\frac{\partial \boldsymbol{u}(\boldsymbol{\xi})}{\partial t} + \bar{\boldsymbol{c}}(\boldsymbol{\xi})\nabla_\xi \boldsymbol{u}(\boldsymbol{\xi}) = 0. \tag{6}$$

where $\bar{\boldsymbol{c}} = \bar{\boldsymbol{u}}(\boldsymbol{\xi}_i) - \bar{\boldsymbol{u}}_D(\boldsymbol{\xi}_i)$ and we denote vector qualntity $\boldsymbol{a}$ in the spatial coordinate as $\bar{\boldsymbol{a}}$ when in the reference coordinate. This can be interpreted as advecting the material velocity $\boldsymbol{u}$ along convective velocity $\bar{\boldsymbol{c}}$, which is defined as the difference of motion between fluid and the background mesh. The mesh velocity $\boldsymbol{u}_D$ is defined as:

$$\boldsymbol{u}_D = \frac{\partial \nabla_\xi \varphi}{\partial t}. \tag{7}$$

One may approximate Eq. (6) with a semi-Lagrangian advection in the spatial coordinate, but in our method, we use reference coordinates to perform semi-Lagrangian advection. Finally, the convective velocity is obtained as follows,

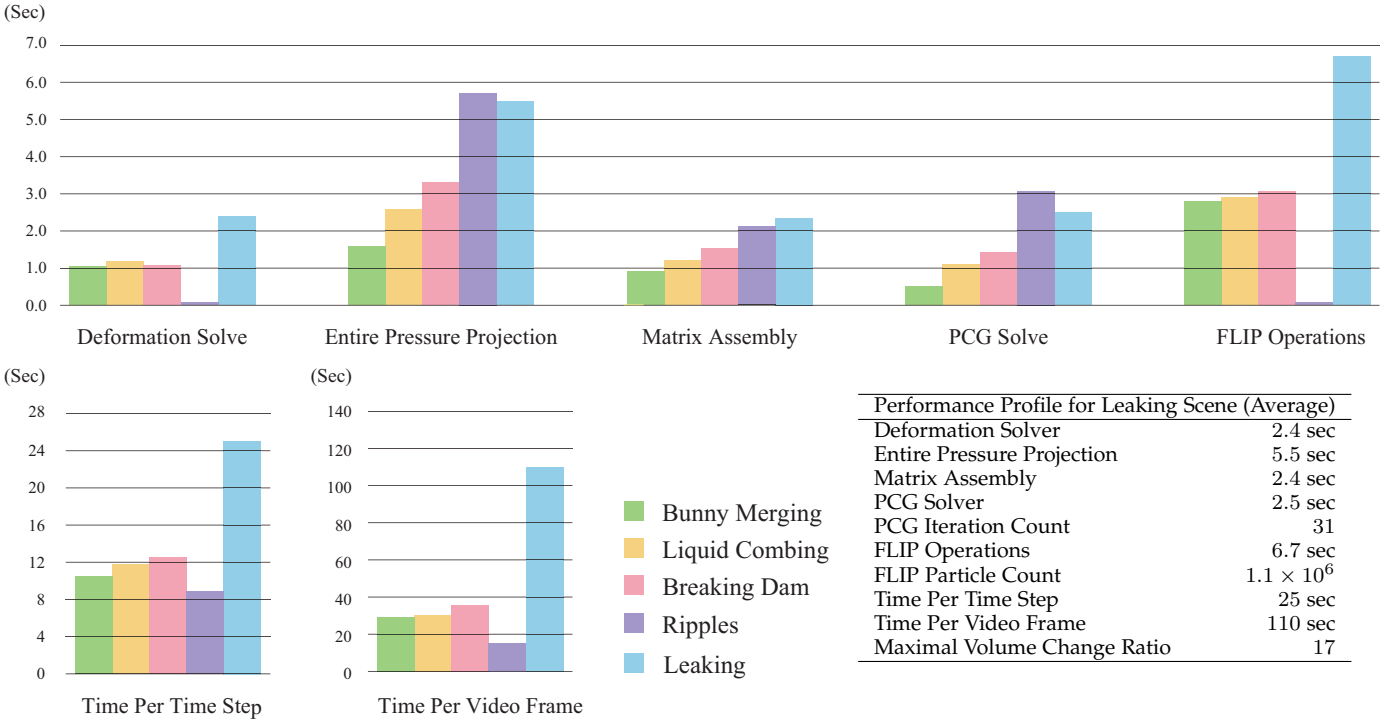
$$\bar{\boldsymbol{c}}(\boldsymbol{\xi}_i) = J^{-1}\boldsymbol{u}(\boldsymbol{x}_i) - J^{-1}\boldsymbol{u}_D(\boldsymbol{x}_i), \tag{8}$$

where $\boldsymbol{\xi}_i$ and $\boldsymbol{x}_i$ denote the corresponding cell centers on respective coordinates, and J is the grid Jacobian, which will be precisely defined in Section 6.1. One remarkable benefit of our advection scheme is that it automatically incorporates the effect of dynamically warping grids using starndard advection schemes. For example, we have employed the MacCormack method [22] combined with the WENO interpolation [23] to advect both the level-set and the velocity, as well as the second-order Runge-Kutta scheme for advecting FLIP particles. In addition, unlike conventional unstructured adaptive methods, we do not need to access a grid element by an $O(\log N)$ tree traversal.

6 Pressure Solver on Warped Grids

At the heart of our method's adaptivity is a pressure solver that runs on a warped grid while being numerically robust and straightforward to implement. Our pressure solver also includes a novel technique to realize second-order accurate Dirichlet boundary conditions on a warped grid, which is formulated via the generalization of the well-known ghost fluid technique [24].

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2018.2883628, IEEE Transactions on Visualization and Computer Graphics



Performance Profile for Leaking Scene (Average)	
Deformation Solver	2.4 sec
Entire Pressure Projection	5.5 sec
Matrix Assembly	2.4 sec
PCG Solver	2.5 sec
PCG Iteration Count	31
FLIP Operations	6.7 sec
FLIP Particle Count	1.1×10^6
Time Per Time Step	25 sec
Time Per Video Frame	110 sec
Maximal Volume Change Ratio	17

Fig. 4. Timings of our examples

6.1 Kinetic Energy Minimization

Our discretization uses piece-wise constant velocity vectors stored at cell centers and piecewise trilinear pressure samples stored on vertices. We derive our pressure solver through kinetic energy minimization [29]:

$$\underset{\boldsymbol{p}}{\text{minimize}} \int_{\Omega} \frac{1}{2}\rho \left|\left| \overset{*}{\boldsymbol{u}} - \frac{\Delta t}{\rho}\nabla p \right|\right|^2 dV, \tag{9}$$

where Δt, $\overset{*}{\boldsymbol{u}}$, Ω, ρ denote the time step size, the intermediate velocity after the advection, the liquid domain, and the fluid density, respectively. We solve Eq. (9) based on the Finite Element Method (FEM). We refer to the reference coordinates with $\boldsymbol{\xi} = (\xi, \eta, \tau)$ and the spatial (physical) coordinates with $\boldsymbol{x} = (x, y, z)$. We express the trilinear pressure function by a sum of shape functions weighted by the pressure samples on vertices:

$$p(\boldsymbol{\xi}) = \sum_{i=1}^{8} N_i(\boldsymbol{\xi}) p_i, \tag{10}$$

where N_i is the trilinear shape function defined on vertex i. Taking the gradient of both sides of Eq. (10) with respect to $\boldsymbol{x}$ (using the chain rule), and then substituting the result into Eq. (9) gives:

$$\underset{\boldsymbol{p}}{\text{minimize}} \sum_{i=1}^{n} \int_{\Omega_i} \frac{1}{2}\rho \left|\left| \overset{*}{\boldsymbol{u}} - \frac{\Delta t}{\rho} J^{-1} \nabla_{\xi} \sum_{j=1}^{8} N_j p_j \right|\right|^2 |J| dV_{\xi}, \tag{11}$$

where n and Ω_i denote the total cell count and a cubic cell domain in the reference coordinates, $\nabla_{\xi} = (\frac{\partial}{\partial \xi}\frac{\partial}{\partial \eta}\frac{\partial}{\partial \tau})^T$, and $dV_{\xi} = d\xi\, d\eta\, d\tau$. $|J|$ is the determinant of a Jacobian matrix $J \in \mathbb{R}^{3\times 3}$ with respect to $\boldsymbol{\xi}$ with each entry given by:

$$J_{ij} = \frac{\partial x_j}{\partial \xi_i} = \frac{\partial}{\partial \xi_i}\Big\{ \sum_{k=1}^{8} N_k(\boldsymbol{\xi}) \boldsymbol{v}_k \Big\}_j, \tag{12}$$

where $\{\}_j$ denotes the j^{th} component of a vector, and $\boldsymbol{v}_k$ is the position vector of vertex k of the warped cube element in spatial coordinates. This Jacobian determinant encodes the relative volume change by a change of coordinate $|J| = dV_x/dV_{\xi}$ where $dV_x = dx\, dy\, dz$. Taking the derivative of Eq. (11) with respect to pressure and setting it to zero yields a symmetric positive-definite linear system:

$$\sum_{i=1}^{n} \int_{\Omega_i} (\nabla)^T D(\nabla) \left[p\right] dV_{\xi} = \sum_{i=1}^{n} \int_{\Omega_i} (\nabla)^T |J| \overset{*}{\boldsymbol{u}}\, dV_{\xi}, \tag{13}$$

where $[p] \in \mathbb{R}^{8\times 1}$ is the discretized pressure on the eight vertices of a grid cell. Note that we reserve $[\ ]$ for discretized variables. $(\nabla) \in \mathbb{R}^{3\times 8}$ is the matrix encoding the gradient, which maps nodal values to a cell-centered vector. Each entry is given by:

$$(\nabla)_{ij} = \left\{ J^{-1} \nabla_{\xi} N_j(\boldsymbol{\xi}) \right\}_i. \tag{14}$$

$D \in \text{diag}(\mathbb{R}^3)$ is the diagonal matrix:

$$D = \frac{|J|\Delta t}{\rho} \boldsymbol{I}, \tag{15}$$

where $\boldsymbol{I}$ is the identity matrix. In our results, we set $\rho = 1$ for convenience. We discretize Eq. (13) with an eight-point Gaussian quadrature integration scheme (single-point Gaussian quadrature is provably unstable, as we discuss in Section 9). Boundary conditions of the system in the absence of surface tension are $p = 0$ on free surfaces and $\boldsymbol{n} \cdot \nabla p = 0$ on static solid boundaries [30], where $\boldsymbol{n}$ is the normal vector of a solid. First-order accurate boundary conditions are given by setting $p_i = 0$ for vacuum vertices, and skipping the evaluation of Eq. (11) for solid cells. We solve the linear system using a preconditioned Conjugate Gradient method

(PCG) with the modified incomplete Cholesky factorization[1] provided by Bridson [30]. Once we have solved for the pressure, we update the velocity at cell centers by:

$$\boldsymbol{u}_{\text{new}} = \overset{*}{\boldsymbol{u}} - \frac{\Delta t}{\rho} J^{-1} \nabla_{\xi} \sum_{k=1}^{8} N_k(\boldsymbol{\xi}_{\text{c}}) p_k, \tag{16}$$

where $\boldsymbol{\xi}_c$ denotes the cell center. Like other fluid simulation methods, we extrapolate the new velocity outside of the fluid domain after the velocity update.

6.2 Accurate Boundary Conditions

Accurate boundary conditions are essential for producing visually pleasant liquid simulations [30]. For solid boundaries, We employ the method of Batty et al. [29] and scale the diagonal entries of D and the intermediate velocity $\boldsymbol{u}^*$ by the fraction of a cell not occupied by solid. Free surface boundary conditions are less straightforward. Unfortunately, first-order accurate Dirichlet boundary conditions exhibit grid-aligned artifacts on the free surface. Enright et al. [24] introduced the *ghost fluid* method to achieve second-order accuracy for Dirichlet boundary conditions on a regular grid. Afterward, the method was extended to irregular meshes where pressure was discretized by the Finite Volume Method (FVM) [17], [36]. Inspired by their methods, we generalize the ghost fluid technique to accurately handle boundary conditions for any grid cell shape. Thus, our new approach works not only for regular grids or tetrahedral meshes, but also for our warped grid discretization. Furthermore, wherever our proposed method is applicable, there is no longer a need to independently develop free surface boundary conditions for each discretization.

We start by reviewing the ghost fluid method for the standard MAC discretization on a regular grid. Consider two nodes across the interface in one dimension. The velocity update at the midpoint of the nodes is:

$$u_{\text{new}} = u^* - \frac{\Delta t}{\rho}\Big(\frac{p_G - p_L}{\Delta x}\Big), \tag{17}$$

which is derived from the discretization of the pressure gradient:

$$(\nabla)\Big[p\Big] = \Big(\frac{p_G - p_L}{\Delta x}\Big), \tag{18}$$

where p_L denotes the pressure on the "liquid" side of the interface, p_G denotes the *ghost pressure* on the "air" side of the interface, and Δx denotes the distance between the nodes. According to Enright et al. [24], the ghost pressure is given by $p_G = (\phi_G/\phi_L)p_L$, where ϕ_G and ϕ_L are the level-set values at the nodes where p_G and p_L are evaluated. Substituting this relation into Eq. (18) gives:

$$(\nabla)\Big[p\Big] = \overbrace{\Big(\frac{\phi_L - \phi_G}{\phi_L}\Big)}^{\triangle}\overbrace{\Big(\frac{0 - p_L}{\Delta x}\Big)}^{\odot}. \tag{19}$$

Note that the $\odot$ term is exactly the first-order accurate free-surface pressure boundary condition, and the $\triangle$ term is a carefully-chosen multiplier that is used in the pressure solver. Our insight here is that the ghost fluid method can be regarded as an "upgrade" operation: The $\triangle$ term explicitly converts the first-order accurate pressure gradient discretization into a second-order accurate one.

1. With MIC(0) parameters τ = 0.97 and γ = 1.0

6.2.1 Generalized Ghost Fluid

We generalize this idea to higher dimensions with a novel linear conversion operator M_ϕ:

$$(\nabla)_2\Big[p\Big] = \overbrace{M_\phi}^{\triangle}\overbrace{(\nabla)_1\Big[p\Big]}^{\odot}. \tag{20}$$

Here, M_ϕ is a 3×3 matrix that takes a first-order-accurate discretization of the pressure gradient vector $(\nabla)_1[p]$ and converts it into a second-order-accurate one $(\nabla)_2[p]$. M_ϕ is then integrated into the larger pressure solver matrix. The exact forms of $(\nabla)_1$ and $(\nabla)_2$ will depend on the particular mesh discretization (regular grid, tetrahedral mesh, etc.), and our approach requires that the pressure gradient is piecewise constant with unknown magnitude and a direction determined by the free surface geometry. We explain how to compute $(\nabla)_1$ and $(\nabla)_2$ for our particular warped grid discretization in Section 6.2.2.

We view the derivation of M_ϕ as a constraint-satisfaction problem. We begin with a 3×3 matrix, consisting of 9 degrees of freedom. Mapping an arbitrary 3-dimensional input vector to the given output vector $(\nabla)_2[p]$ will only pin down three of these degrees of freedom. Because we will eventually integrate M_ϕ into our symmetric-positive definite linear system, we also impose symmetry and positive-definiteness onto M_ϕ. We utilize the remaining degrees of freedom to optimize the numerical stability of M_ϕ. We work out these details and provide the analytical form of M_ϕ in Appendix A.

Once M_ϕ is computed, we can enjoy second-order accurate free surface boundary conditions by simply replacing D in Eq. (13) with DM_ϕ in all cells that overlap the free surface boundary. We also use the second order pressure gradient in Eq. (20) to update the velocity in Eq. (16). We would like to emphasize that our new approach not only enforces second-order accurate boundary conditions, but it also preserves the symmetry and positive-definiteness of the system matrix and exhibits provably optimum numerical conditioning. The resulting linear solver is remarkably stable and converges quickly.

6.2.2 Ghost Fluid Applied to our Discretization

Here, we derive the forms of $(\nabla)_1$ and $(\nabla)_2$ in our particular discretization, which stores both pressures and level-set samples at grid vertices. We can analytically enforce an accurate $p = 0$ condition at the free surface by constraining the pressure to be a multiple of the level set function $[p] = [\phi]P$, as pointed out by Ando et al. [64]. To compute the first order pressure gradient $(\nabla)_1[p]$, instead of satisfying the $p = 0$ condition exactly at the interface, we satisfy it at each node outside of the liquid. We do this by again expressing the pressure as a multiple of the distance function, but this time we replace all level set values outside of the liquid by zero: $(\nabla)_1[p] = (\nabla)[\mathring{\phi}]P$, where $[\mathring{\phi}] = [\max(0, \phi)]$ is a vector of level-set values, with all entries outside of the liquid set to zero. By expressing the pressure in this way, we only allow one degree of freedom per cell and introduce trilinear error terms ($O(\Delta x \Delta y \Delta z)$ in three dimensions), which conveniently does not affect the first-order accuracy of this pressure gradient discretization. Plugging these pressure gradient discretizations into Eq. (20) gives:

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2018.2883628, IEEE Transactions on Visualization and Computer Graphics

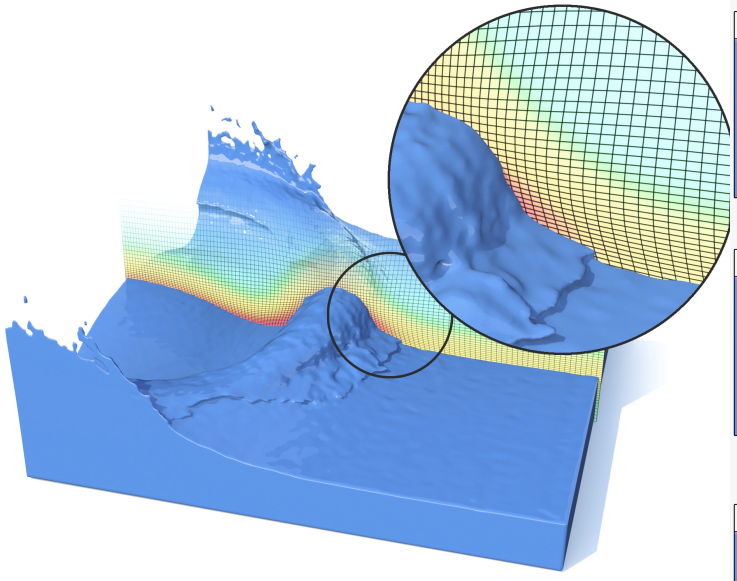


Fig. 5. Breaking dam creating splahes and thin sheets by our dynamically warping grids adapting to the surfaces: $128 \times 64 \times 128$ resolution, 12 seconds per time step and 36 seconds per video frame.

$$(\nabla)[\phi]P = \overbrace{M_\phi}^{\triangle}\overbrace{(\nabla)[\mathring{\phi}]P}^{\odot} \tag{21}$$

and we can divide by the unknown P to prescribe a relationship for M_ϕ for each boundary cell:

$$(\nabla)[\phi] = M_\phi(\nabla)[\mathring{\phi}] \tag{22}$$

Thus, for the purposes of solving for M_ϕ in Section 6.2.1, we can use $(\nabla)_1 = (\nabla)[\phi]$ and $(\nabla)_2 = (\nabla)[\mathring{\phi}]$. Note that the level-set value ϕ must be the actual signed distance in spatial coordinates. To calculate ϕ, we first compute the level-set $\bar{\phi}$ in reference coordinates, and convert to ϕ in spatial coordinates by:

$$\phi = \bar{\phi} \Big/ \left|\left|\nabla\bar{\phi}\right|\right| = \bar{\phi} \Big/ \left|\left|J^{-1}\nabla_\xi\bar{\phi}\right|\right|. \tag{23}$$

This conversion operation is only valid for cells near the boundary, but accurate distances in spatial coordinates are only needed for boundary conditions anyway. Thus, we perform this conversion only near the free surfaces before the projection step.

7 Visualization

To visualize our simulation, we first construct a surface mesh of FLIP particles on the Cartesian grids. Next, we displace the vertices of the mesh as well as ballistic FLIP particles through the deformation field. In our implementation, we use OpenVDB [43] to create this mesh.

8 Results

We ran all of our examples on a Linux machine with 2.7GHz Intel Xeon E5-2697, using a target L_∞ norm of 10^{-3} for the relative residual of the pressure solver. The breaking dam set up of Figure 5 and the example of a bunny falling into a basin in Figure 2 illustrate that our method simulates detailed liquids. Our dynamically warping grids

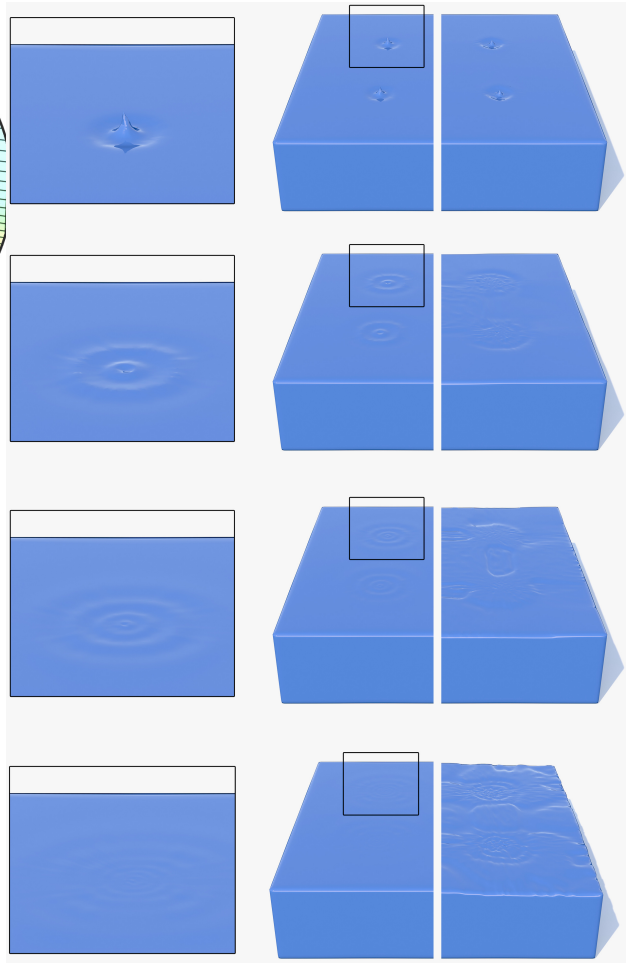
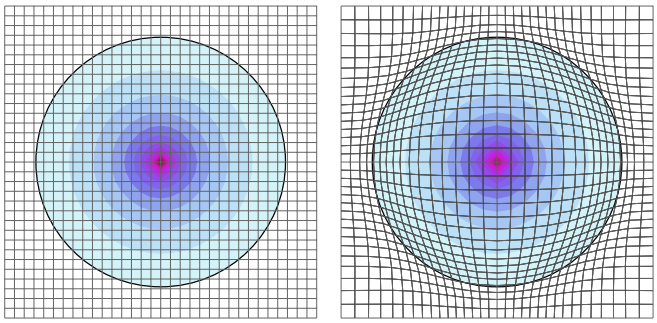


Fig. 6. Expanding ripples with high adaptivity. Left: our second-order accurate free surface boundary conditions capturing the subtle motion of expanding ripples in a nearly open-space static pool. $128 \times 64 \times 128$ resolution, 9 seconds per time step, 15 seconds per video frame and the maximal volume ratio 200. Right: first-order accurate boundary conditions with the same setup exhibit grid aligned artifacts. This example employs a level-set surface tracker to demonstrate the accuracy of our method, and the same scenario using FLIP is provided in the supplemental video.

act to exaggerate the visually important geometric features, such as splashes and thin liquid sheets. The cross-sections shown in each figure illustrates the warped grid cells and are color-coded by the local volume change induced by our warping. A reference coordinate view of the latter example is shown in the second row of Figure 2. It highlights that our solver elegantly handles refining and coarsening obstacles in the flow. Each time step took 12.5 and 10.5 seconds for this example, of which the entire projection required 3.3 seconds and 1.6 seconds. The grid deformation solver took 1.1 seconds and 1.0 seconds, and the maximal volume differences were 7.4 and 15, respectively.

The liquid combing set up of Figure 3 highlights our ability to handle the complex interaction with solid boundaries more accurately than a MAC solver at the same resolution. A direct comparison with a MAC solver is shown in Figure 10. In this example, the MAC solver is not able to capture the small-scale splashes due to the lack of resolution between the cylinders. In this case, we have

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2018.2883628, IEEE Transactions on Visualization and Computer Graphics



	First-order Accuracy		Ours	
Resolution	L_1 error	Order	L_1 error	Order
32	4.9×10^{-3}	N/A	2.3×10^{-4}	N/A
64	3.0×10^{-3}	0.73	5.2×10^{-5}	2.0
128	1.5×10^{-3}	0.95	1.3×10^{-5}	2.0
256	7.1×10^{-4}	0.95	3.4×10^{-6}	2.0
512	3.6×10^{-4}	1.1	9.1×10^{-7}	1.9
1024	1.9×10^{-4}	1.0	2.4×10^{-7}	1.9
	First-order Accuracy		Ours	
Resolution	L_1 error	Order	L_1 error	Order
32	4.4×10^{-3}	N/A	1.3×10^{-4}	N/A
64	1.5×10^{-3}	1.6	2.8×10^{-5}	2.1
128	4.5×10^{-4}	1.7	5.6×10^{-6}	2.4
256	1.7×10^{-4}	1.4	1.3×10^{-6}	2.1
512	6.0×10^{-5}	1.5	3.3×10^{-7}	1.9
1024	2.3×10^{-5}	1.4	9.8×10^{-8}	1.8

Fig. 7. Numerical verification of our second-order accuracy for a Poisson's problem: we solve $\nabla^2 p(\boldsymbol{x}) = \delta(\boldsymbol{x})$ with $p = 0$ boundary conditions on the lines of solid circles. The colored contour plots show the magnitude of our actual numerical solution. The top table refers to the experiment with the regular setting (top left) and the bottom our warped setting (top right). Both numerical experiments show the second-order convergence.

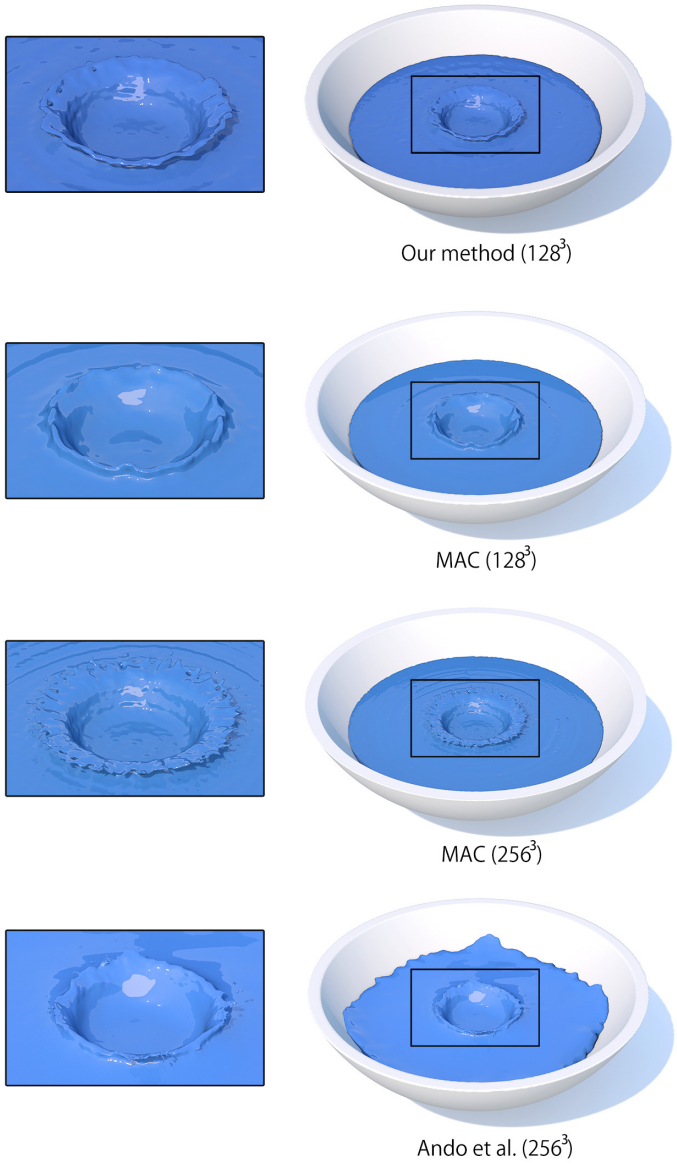


Fig. 8. Comparison of a drop falling into a pool using three different solvers. Our method successfully captures the thin sheets of a crown splash at 128^3 resolution, while the MAC method fails to simulate such detail without doubling the resolution. The adaptive method of Ando et al. [20] successfully recovers small-scale details but leads to significantly longer runtimes. Timing details are given in Table 1.

increased the effective resolution around the obstacles by using the distance to the cylinders as a region of interest in Eq. (2). Each time step of this simulation took 12 seconds on average. Our dynamic grid deformation solver took 1.2 seconds whereas the pressure projection took 2.6 seconds. The maximal volume ratio was 18 on average.

The example of Figure 6 verifies that our method yields the expected gain in visual quality when using second-order accurate boundary conditions. We set up the scene with four small droplets on a static pool with a high degree of spatial adaptivity of a maximal volume ratio of 200. In this example, our second-order accurate boundary conditions shown on the left can capture both sharper splashes and propagating ripples more accurately than the first-order accurate boundary conditions shown on the right. Each time step of this simulation took 8.9 seconds, and the pressure projection took 5.7 seconds on average. In this specific example, we have used a static warped grid (instead of deforming it over time), and we used the level-set method for tracking liquid surfaces to demonstrate our visual accuracy. For clarification, we also provide an example of the same setup using FLIP in the supplemental video. We note that when FLIP is employed, a subtle noise on the surface persists once the particles are perturbed. However, these artifacts are orthogonal to our boundary conditions and could be alleviated by post-processing the liquid surface.

The simulation of liquid leaking from a container with holes in the bottom is shown in Figure 1. Our method can reproduce the visually interesting streams of turbulent water. A MAC solver, on the other hand, needs to double the resolution to reproduce similar apparent detail. The comparison with the MAC solver for this example at the same resolution is shown in Figure 10, as well as in the supplemental video. Each time step took 25 seconds, grid deformation 2.4 seconds, and the pressure projection 5.5 seconds on average. The maximal volume ratio of this example was 17 on average.

Method	Resolution	Time Per Step [s]
MAC	128^3	4.28
Ours	128^3	10.2
MAC	256^3	26.4
[20]	256^3	29.0

TABLE 1
Timings of Figure 8.

8.1 Numerical Verification

We additionally performed a numerical verification for our free surface boundary conditions, and observe that it yields second-order convergence into the analytical solution. We set up our numerical test in 2D with a single point source of divergence at the center, and solve $\nabla^2 p(\boldsymbol{x}) = \delta(\boldsymbol{x})$ with the $p = 0$ boundary conditions enforced on the surface of a circular domain. The analytical solution to this configuration is given by $p(\boldsymbol{x}) = G(||\boldsymbol{x}||) - G(r)$, where G and r denote the Green's function solution of Laplace's equation, and the radius of a circle.

We measured the average of the L_1 norm of the error on the surface, and examined the convergence factor by doubling the resolution, similar to Enright et al. [65] and Batty [66]. Our results in Figure 7 show that both with and without the grid deformation the convergence rate stays second-order. Interestingly, we also observed that the convergence rate improves even more when our sizing function is applied. In this case, our method (with parameters $\alpha_\phi = 7$, $\beta_\psi = 0$ and $\gamma_u = 0$) further improves the solution by increasing the effective resolution around the surface.

8.2 Timings

A detailed performance breakdown for our method is shown in Figure 4. In all of our examples, the cost of the grid deformation solver was less than 10 percent of the whole simulation time, and the timings for the PCG solver were close to the timings for assembling the linear system. For the leaking scene example, the average total cost spent on these two operations (deformation solver and the matrix assembly) was 4.8 seconds. Hence, the computational overhead of our method compared to a standard liquid solver is approximately 19 percent. Note that since we highly depend on the performance of a linear system solver for the grid deformation as well as the pressure solver, we expect that our solver can be sped up with more sophisticated methods, such as a Schur complement solver [67]. Note that the timings for the grid deformation of Figure 6 are omitted since we did not dynamically warp the grid in this example.

8.3 Comparisons

Figure 8 shows a comparison of a drop falling into a pool of liquid using three different solvers: our method, a regular MAC solver and the adaptive method of Ando et al. [20]. Animations for all solvers are provided in the supplemental video. For this setup, our method with 128^3 resolution can capture fine details, e.g., for the crown splash after the drop impact. Such features are noticeably less detailed using a regular MAC grid at the same resolution. Doubling the resolution of the MAC grid would resolve the features, but leads to very significant increases in runtime. In this case the MAC grid runtime is 2.58 times higher than our 128^3 warped grid. The method of Ando et al. [20] enables large-scale simulations with aggressive adaptivity, but the overhead arising from the unstructured BCC-meshes can counteract gains in performance at moderate resolutions. The resulting simulation recovers small-scale details but leads to an almost three times higher runtime than our method. The timings of this comparison can be found in Table 1.

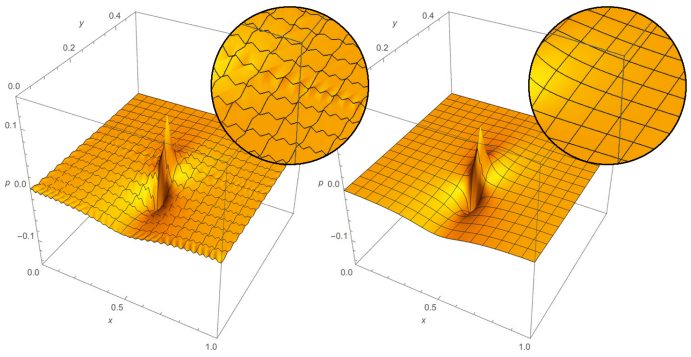


Fig. 9. Pressure surface plot with a single strong velocity initiated at the center pointing upward. Left: a single-point Gaussian quadrature integration rule applied to integrate Eq. (13) exhibits oscillation due to null-space issues. Right: our eight-point (four-point in 2D) Gaussian integration rule removes the artifacts.

9 Discussion

We observe ringing artifacts when a naive single-point Gaussian quadrature integration rule is employed to integrate Eq. (13). This issue arises from a null-space in the system. An example of our null-space is the pressure gradient evaluated at the cell center with pressure values of -1, 1, -1, 1 assigned on four vertices in counter-clockwise. In this specific configuration, the pressure gradient unphysically evaluates to zero. Such a null-space is known to induce numerical instabilities when solving a linear system, and researchers have taken care to eliminate such instabilities [68], [69]. We circumvent this issue by switching to the eight-point Gaussian quadrature integration rule. Figure 9 illustrates the comparison of the effect of our integration scheme.

Like other fluid simulation frameworks, our method also undergoes slight volume changes depending on the accuracy of the employed velocity advection scheme and surface tracker. Hence, we include the method introduced by Kim et. [6] in our solver. We distribute the correction term to the right-hand side of our pressure solver weighted by the Jacobian of the cells. We found this simple technique effectively recovers the original volume. We compute the volume on the Cartesian coordinate by summing up the Jacobians weighted by a cell fraction of fluid.

We note that an alternative to our discretization, aside from the boundary conditions, is the method by Azevedo and Oliveira [46] where FVM is employed. We prefer our FEM-based discretization since the FVM is known to introduce accuracy errors when computing pressure gradients

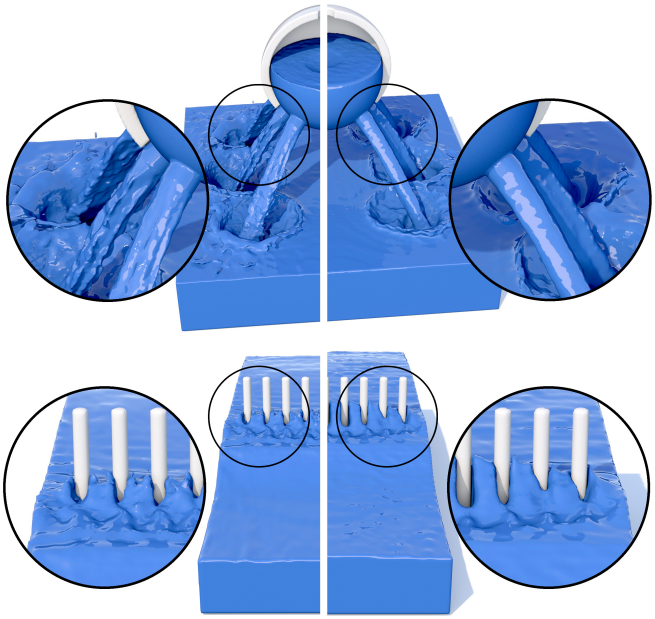


Fig. 10. Comparison with the MAC method. Left side shows our method, right side the MAC solver at the same resolution.

unless the circumcenter of an element is chosen as a sampling location of pressure. As pointed out by Batty et al. [36] the pressure differentiation degrades to the first-order accuracy if it is not evaluated at the circumcenter. We also note that the method of Ferstl et al. [12] can be extended to use our pressure solver. Our work introduces novel generalized second-order accurate boundary conditions, which could also be beneficial for these other discretizations.

Using a FLIP surface tracker adds persistent surface noise to any discretization, and subtle surface bumps are visible in our FLIP results as well. We create our surfaces by extracting the surface in reference coordinates and then warping it to spatial coordinates, so our warped grids will warp the existing FLIP artifacts along with it. This warping essentially makes isotropic FLIP artifacts smaller and more anisotropic, depending on the sizing function. In the end, these artifacts are orthogonal to our boundary conditions, and they can be alleviated by post-processing the liquid surface.

Currently, our deformation tends to produce thin elements on liquid surfaces, which has a side effect of robustly capturing thin sheets of liquid. On the other hand, we see that these thin features can introduce visible artifacts along the elongated directions. In future work, we would like to utilize methods developed for image warping [51], [52] to address the issue. Such vector-valued solvers could enable additional rotational displacements, allowing us to increase the expressiveness of the achievable deformation, at the expense of more computational degrees of freedom. Similar to Zhu et al. [47], our warped grid could be used to produce an effect similar to non-reflecting boundary conditions.

Our current implementation limits the scale of adaptivity, primarily because of the increased cost of deformation solver and the limited CFL number. This issue is common among spatially adaptive simulation methods, where a time step size must be adjusted according to the ratio of the maximum velocity and the minimum grid cell size. This restriction could be reduced by advecting grid points through the velocity field, as demonstrated by Fan et al. [70] for solids.

The types of grids generated by our method are naturally limited to those with the connectivity of a regular grid. Consequently, our method cannot create some grid configurations like uniformly tiny cells all around the boundary of a closed region but uniformly large cells in the interior. On the other hand, we do not restrict the boundary of our grid to the boundary of the simulation, so our method can certainly create a region of fine cells surrounded by coarse cells. We note that this topic only affects the calculation of the deformation itself, and does not influence our generalized boundary conditions, pressure solver, or advection strategy.

The numerical conditioning depends on element quality in the same way as other curvilinear grid methods, and the warping often results in an increased number of active cells compared to a standard MAC solver at the same resolution.

Although our method benefits from the memory access patterns of regular grids, our implementation is not yet optimized to its full extent. Techniques such as the efficient sparse grid methods [39], [43], [71], [72] were shown to yield very high performance, but we point out that the fundamental ideas we introduce are orthogonal to the aforementioned works. Our method for warping grids could be used in conjunction with these techniques to further increase the amount of resolved detail.

9.1 Momentum Preservation

We want to point out that our velocity field is not exactly momentum preserving because of the moving mesh. Suppose that we define momentum of an element i as:

$$\int_{\Omega_i} \boldsymbol{u}(\boldsymbol{x}) dV, \tag{24}$$

where $\boldsymbol{u}(\boldsymbol{x})$ and Ω_i denote the continuous function of velocity and the volumetric region of an element respectively. For simplicity, we set $\rho = 1$ in this exposition, because density is constant in our case. The change of momentum over time is given as:

$$\frac{d}{dt}\int_{\Omega_i} \boldsymbol{u}(\boldsymbol{x}) dV = \int_{\Omega_i} \frac{\partial \boldsymbol{u}(\boldsymbol{x})}{\partial t} dV + \oint_{\partial\Omega_i} \boldsymbol{u}(\boldsymbol{x})(\boldsymbol{u}_D(\boldsymbol{x}) \cdot \boldsymbol{n}) dS, \tag{25}$$

where $\partial\Omega_i$ denotes the boundary of the region Ω_i. Notice that the second term of Eq. (25) on the right considers a flux of momentum on the region boundary. We note that our advection scheme, although it takes into account mesh velocity, does not compute momentum flux on element boundaries. Although this violation does not seem to be a serious cause of visual artifacts so far, further investigation is needed to achieve more accurate dynamics.

10 Conclusion

This paper presented dynamically warping grids for liquid simulations. We devised a method to allow flexible spatial adaptivity on regular grids. Consequently, our method can capture complex and diverse liquid motions while retaining the advantages of structured grids. We demonstrated that

our method runs in harmony with off-the-shelf algorithms for velocity advection and surface tracking.

We also combined novel generalized second-order accurate boundary conditions with our FEM-based pressure solver to enable subtle liquid surface dynamics. We proved that this scheme has optimal convergence properties (symmetry, positive definiteness, and condition number, see Appendix A), and we have verified its accuracy through numerical experiments. We also presented a new grid deformation solver with temporal coherence, which requires less than 10% of the overall calculations.

Altogether, we demonstrated that our method improves the visual quality of liquid simulations through the use of adaptivity, and without excessive computational overhead or complicated implementations. In the future, we would like to extend our method to handle moving solid boundaries, two-way coupled rigid bodies and surface tension forces. We would also like to combine our approach with sparse grid methods [39], [43], [44], to merge the fast lookup times of sparse grids with our method's ability to maintain temporal coherence and adapt to subtle changes in boundary conditions.

APPENDIX A
COMPUTATION OF M_ϕ

We can view Eq. (20) as a conversion between an arbitrary source vector $\boldsymbol{b}$ to the target vector $\boldsymbol{a}$:

$$\boldsymbol{a} = M_\phi \boldsymbol{b}. \tag{26}$$

We introduce a rotation matrix R such that:

$$\begin{aligned} R\boldsymbol{a} &= \hat{\boldsymbol{a}} = (\hat{a}_x\ \hat{a}_y\ 0)^T \\ R\boldsymbol{b} &= \hat{\boldsymbol{b}} = (\hat{b}_x\ 0\ 0)^T. \end{aligned} \tag{27}$$

We then apply this change of variables to simplify the problem

$$\hat{\boldsymbol{a}} = M_R \hat{\boldsymbol{b}} \tag{28}$$

such that $M_R = RM_\phi R^T$ and $M_\phi = R^T M_R R$. We ensure M_R is symmetric and positive-definite by factoring it with a Cholesky decomposition: $M_R = L_R L_R^T$, where L_R is the lower triangular matrix

$$L_R = \begin{pmatrix} c_{11} & 0 & 0 \\ c_{21} & c_{22} & 0 \\ c_{31} & c_{32} & c_{33} \end{pmatrix}. \tag{29}$$

This decomposition constrains M_ϕ to be symmetric too, because

$$\begin{aligned} M_\phi &= R^T M_R R \\ &= R^T L_R L_R^T R \\ &= \left(R^T L_R\right)\left(R^T L_R\right)^T. \end{aligned} \tag{30}$$

This also makes M_ϕ positive definite, because for all nonzero vectors $\boldsymbol{x}$ and $\boldsymbol{y} = R^T\boldsymbol{x}$,

$$\begin{aligned} \boldsymbol{y}^T M_\phi \boldsymbol{y} &= \boldsymbol{x}^T R M_\phi R^T \boldsymbol{x} \\ &= \boldsymbol{x}^T M_R \boldsymbol{x} \\ &> 0, \quad \text{by positive-definiteness of } M_R \end{aligned} \tag{31}$$

The positive definite property is extremely helpful for numerical algorithms, but it imposes a constraint on the input vectors $\boldsymbol{a}$ and $\boldsymbol{b}$, namely that $\hat{\boldsymbol{b}}^T M_R \hat{\boldsymbol{b}} = \hat{\boldsymbol{b}}^T \hat{\boldsymbol{a}} > 0$. Physically, this means that the angle between the first- and second-order pressure gradients must be less than $\pi/2$ (they must not point in opposite directions). In the rare event that this condition is violated we resort back to the first order accuracy for safety. This is analogous to the strategy of clamping small level-set values to prevent instability in the traditional ghost fluid method. Using the Cholesky decomposition and the zero entries of Eq. (27) to constrain M_R gives us

$$M_R = \begin{pmatrix} \hat{a}_x/\hat{b}_x & \hat{a}_y/\hat{b}_x & 0 \\ \hat{a}_y/\hat{b}_x & c_{22}^2 + \hat{a}_y^2/(\hat{a}_x\hat{b}_x) & c_{22}c_{32} \\ 0 & c_{22}c_{32} & c_{32}^2 + c_{33}^2 \end{pmatrix}. \tag{32}$$

This matrix still has three degrees of freedom, c_{22}, c_{32}, and c_{33}, which we use to optimize numerical stability of the boundary conditions, by minimizing the condition number:

$$\kappa(M_R) = \frac{|\lambda_{\max}|}{|\lambda_{\min}|} \tag{33}$$

where $\lambda_{\max}$ and $\lambda_{\min}$ are the maximum and minimum eigenvalues of M_R. According to Marshall and Olkin [73], the following inequality holds for any positive-definite matrix:

$$\kappa \begin{pmatrix} U_{11} & U_{12} \\ U_{21} & U_{22} \end{pmatrix} \geq \kappa(U_{11}). \tag{34}$$

Applying this to our case,

$$\kappa(M_R) \geq \kappa \begin{pmatrix} \hat{a}_x/\hat{b}_x & \hat{a}_y/\hat{b}_x \\ \hat{a}_y/\hat{b}_x & c_{22}^2 + \hat{a}_y^2/(\hat{a}_x\hat{b}_x) \end{pmatrix}. \tag{35}$$

Our strategy will be to first manipulate our degree of freedom c_{22} on the right hand side to minimize the lower bound on $\kappa(M_R)$. Then, we will try to optimize M_R's eigenvalues such that $\kappa(M_R)$ is exactly equal to that lowest possible condition number. We first analytically minimize the right hand side of Eq. (35) by setting $c_{22}^2 = (\hat{a}_x^2 + \hat{a}_y^2)/(\hat{a}_x\hat{b}_x)$. Next, we set $c_{32} = 0$, which conveniently factors the characteristic polynomial of M_R into a scalar times the upper left 2×2 sub-problem that we already minimized. Finally, we ensure that this third eigenvalue is neither a minimum nor a maximum (it does not affect the condition number) by setting it to the average of the other two eigenvalues, which are guaranteed to be positive due to the positive-definiteness property. At this point, we have constrained all the degrees of freedom in M_R, and its condition number is equal to that of its optimized upper left sub-matrix (the inequality in Eq. (35) becomes an equality). Therefore, it has the minimum possible condition number. The final matrix is given by:

$$M_R = \begin{pmatrix} \hat{a}_x/\hat{b}_x & \hat{a}_y/\hat{b}_x & 0 \\ \hat{a}_y/\hat{b}_x & (\hat{a}_x^2 + 2\hat{a}_y^2)/(\hat{a}_x\hat{b}_x) & 0 \\ 0 & 0 & (\hat{a}_x^2 + \hat{b}_y^2)/(\hat{a}_x\hat{b}_x) \end{pmatrix}. \tag{36}$$

This matrix M_R achieves second-order accurate boundary conditions, symmetric positive-definiteness, and provably optimum numerical conditioning. The resulting linear solver is remarkably stable and converges quickly in practice.

APPENDIX B
NUMERICAL VERIFICATION OF PRESSURE SOLVER

We ran our pressure solver on a Taylor Green vortex velocity field, of which the analytical solution is known. This experiment is absent of free surfaces but allows us to accurately evaluate the behavior of the pressure solver under deformation. We measure L_1 error of pressure and plot convergence as we double the grid resolutions while keeping the sizing function constant (Figure 11 bottom left). Figure 11 shows the result of varying resolutions ranging from 32×32 to 1024×1024. The error measurements clearly show that our pressure solver properly converges to the analytical solution under refinement. As a consequence, our method likewise yields the correct pressure gradient, which is crucial to enforce incompressibility for the FLIP simulations.

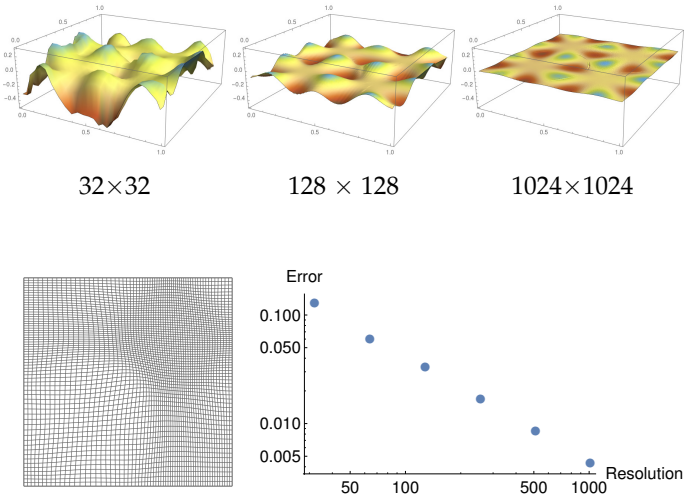


Fig. 11. Error plots of our Taylor Green vortex velocity experiment. Pressure error (top), warped grid from our sizing function (bottom left), and the log-log graph of the total error with respect to grid resolutions (bottom right).

ACKNOWLEDGMENT

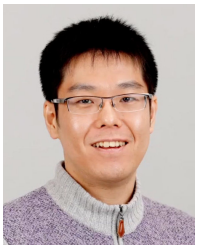
This work was partially supported by JSPS Grant-in-Aid for Young Scientists (Start-up) 16H07410, the ERC Starting Grants *realFlow* (StG-2015-637014) and *BigSplash* (StG-2014-638176). This research was supported by the Scientific Service Units (SSU) of IST Austria through resources provided by Scientific Computing. We would like to express my gratitude to Nobuyuki Umetani and Tomas Skrivan for insightful discussion.

Hikaru Ibayashi is a CS Ph.D. student at the University of Southern California (USC). He received his physics bachelors degree, in 2015 and computer science master's degree in 2017, both from the University of Tokyo. Now, he belongs to the MINDS Research Group at USC. His research interests include fluid simulation and statistical machine learning. He is on the Dean's list of the University of Tokyo for his master thesis.

Chris Wojtan received his B.S. in Computer Science in 2004 from the University of Illinois in Urbana Champaign and his Ph.D. in Computer Graphics from the Georgia Institute of Technology in 2010. He is a recipient of the National Science Foundation Graduate Research Fellowship, the Georgia Tech Sigma Xi Best Ph.D. Thesis Award, an ERC Starting Grant, the 2015 Eurographics Young Researcher Award, and the 2016 SIGGRAPH Significant New Researcher Award. Chris is currently a Professor at the Institute of Science and Technology Austria (IST Austria), and his current research interests are physically-based animation and geometry processing.

Nils Thuerey is an Associate-Professor at the Technical University of Munich (TUM). He works in the field of computer graphics, where a central theme of his research are physics simulations and deep learning algorithms. He acquired a Ph.D. for his work on liquid simulations in 2006 from the University of Erlangen-Nuremberg. Until 2010 he held a position as a post-doctoral researcher at ETH Zurich. He received a tech-Oscar from the AMPAS in 2013 for his research on controllable smoke effects. Subsequently, he worked for three years as R&D lead at ScanlineVFX, before starting at TUM in October 2013.

Takeo Igarashi is a professor at CS department, the University of Tokyo. He received PhD from Dept of Information Engineering, the University of Tokyo in 2000. His research interest is in user interface in general and current focus is on interaction techniques for 3D graphics. He received The Significant New Researcher Award at SIGGRAPH 2006.



Ryoichi Ando is an Assistant Professor at National Institute of Informatics, Japan. Prior to that, he has been a postdoctoral researcher at IST Austria. He earned Ph.D. of Design from the design department at Kyushu University in 2014. His research interests include physics-based animations for computer graphics with a strong emphasis on fluid.

REFERENCES

[1] C. Wojtan, N. Thuerey, M. Gross, and G. Turk, "Physics-inspired topology changes for thin fluid features," *ACM Trans. Graph., vol. 29, no. 4, pp. 1–8, 2010.*

[2] *M. Müller, "Fast and robust tracking of fluid surfaces," in Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, ser. SCA '09. New York, NY, USA: ACM, 2009, pp. 237–245.*

[3] *H. Wang, P. J. Mucha, and G. Turk, "Water drops on surfaces," ACM Trans. Graph., vol. 24, no. 3, pp. 921–929, Jul. 2005.*

[4] *F. Da, C. Batty, C. Wojtan, and E. Grinspun, "Double bubbles sans toil and trouble: Discrete circulation-preserving vortex sheets for soap films and foams," ACM Trans. Graph., vol. 34, no. 4, pp. 149:1–149:9, Jul. 2015.*

[5] *O. Busaryev, T. K. Dey, H. Wang, and Z. Ren, "Animating bubble interactions in a liquid foam," ACM Trans. Graph., vol. 31, no. 4, pp. 63:1–63:8, Jul. 2012.*

[6] *B. Kim, Y. Liu, I. Llamas, X. Jiao, and J. Rossignac, "Simulation of bubbles in foam with the volume control method," ACM Trans. Graph., vol. 26, no. 3, Jul. 2007.*

[7] *S. Patkar, M. Aanjaneya, D. Karpman, and R. Fedkiw, "A hybrid lagrangian-eulerian formulation for bubble generation and dynamics," in Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation, ser. SCA '13. New York, NY, USA: ACM, 2013, pp. 105–114.*

[8] *J.-M. Hong, H.-Y. Lee, J.-C. Yoon, and C.-H. Kim, "Bubbles alive," ACM Trans. Graph., vol. 27, no. 3, pp. 48:1–48:4, Aug. 2008.*

[9] *N. Thuerey, C. Wojtan, M. Gross, and G. Turk, "A multiscale approach to mesh-based surface tension flows," ACM Trans. Graph., vol. 29, no. 4, pp. 48:1–48:10, Jul. 2010.*

[10] *J. A. Canabal, D. Miraut, N. Thuerey, T. Kim, J. Portilla, and M. A. Otaduy, "Dispersion kernels for water wave simulation," ACM Trans. Graph., vol. 35, no. 6, pp. 202:1–202:10, Nov. 2016.*

[11] *F. Losasso, F. Gibou, and R. Fedkiw, "Simulating water and smoke with an octree data structure," ACM Trans. Graph., vol. 23, no. 3, pp. 457–462, Aug. 2004.*

[12] *F. Ferstl, R. Westermann, and C. Dick, "Large-scale liquid simulation on adaptive hexahedral grids," Visualization and Computer Graphics, IEEE Transactions on, vol. 20, no. 10, pp. 1405–1417, Oct 2014.*

[13] *B. M. Klingner, B. E. Feldman, N. Chentanez, and J. F. O'Brien, "Fluid animation with dynamic meshes," ACM Trans. Graph., vol. 25, no. 3, pp. 820–825, Jul. 2006.*

[14] *B. E. Feldman, J. F. O'Brien, and B. M. Klingner, "Animating gases with hybrid meshes," ACM Trans. Graph., vol. 24, no. 3, pp. 904–909, Jul. 2005.*

[15] *P. Clausen, M. Wicke, J. R. Shewchuk, and J. F. O'Brien, "Simulating liquids and solid-liquid interactions with lagrangian meshes," ACM Trans. Graph., vol. 32, no. 2, pp. 17:1–17:15, Apr. 2013.*

[16] *N. Chentanez, B. E. Feldman, F. Labelle, J. F. O'Brien, and J. R. Shewchuk, "Liquid simulation on lattice-based tetrahedral meshes," in Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, ser. SCA '07. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2007, pp. 219–228.*

[17] *T. Brochu, C. Batty, and R. Bridson, "Matching fluid simulation elements to surface geometry and topology," ACM Trans. Graph., vol. 29, no. 4, pp. 47:1–47:9, Jul. 2010.*

[18] *F. de Goes, C. Wallez, J. Huang, D. Pavlov, and M. Desbrun, "Power particles: An incompressible fluid solver based on power diagrams," ACM Trans. Graph., vol. 34, no. 4, pp. 50:1–50:11, Jul. 2015.*

[19] *F. Sin, A. W. Bargteil, and J. K. Hodgins, "A point-based method for animating incompressible flow," in Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, ser. SCA '09. New York, NY, USA: ACM, 2009, pp. 247–255. [Online]. Available: http://doi.acm.org/10.1145/1599470.1599502*

[20] *R. Ando, N. Thuerey, and C. Wojtan, "Highly adaptive liquid simulations on tetrahedral meshes," ACM Trans. Graph. (Proc. SIGGRAPH 2013), July 2013.*

[21] *F. Ferstl, R. Ando, C. Wojtan, R. Westermann, and N. Thuerey, "Narrow Band FLIP for Liquid Simulations," Eurographics'16, vol. to appear, p. 8, May 2016.*

[22] *A. Selle, R. Fedkiw, B. Kim, Y. Liu, and J. Rossignac, "An unconditionally stable maccormack method," J. Sci. Comput., vol. 35, no. 2-3, pp. 350–371, Jun. 2008.*

[23] *C. B. Macdonald and S. J. Ruuth, "Level set equations on surfaces via the closest point method," Journal of Scientific Computing, vol. 35, no. 2, pp. 219–240, 2008.*

[24] *D. Enright, D. Nguyen, F. Gibou, and R. Fedkiw, "Using the particle level set method and a second order accurate pressure boundary condition for free surface flows," in In Proc. 4th ASME-JSME Joint Fluids Eng. Conf., number FEDSM200345144. ASME, 2003, pp. 2003–45 144.*

[25] *J. Stam, "Stable fluids," in Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques, ser. SIGGRAPH '99. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 1999, pp. 121–128.*

[26] *N. Foster and R. Fedkiw, "Practical animation of liquids," in Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, ser. SIGGRAPH '01. New York, NY, USA: ACM, 2001, pp. 23–30.*

[27] *F. H. Harlow and J. E. Welch, "Numerical calculation of timedependent viscous incompressible flow of fluid with free surface," Physics of Fluids, vol. 8, no. 12, pp. 2182–2189, 1965.*

[28] *N. Foster and D. Metaxas, "Controlling fluid animation," in Proceedings of the 1997 Conference on Computer Graphics International, ser. CGI '97. Washington, DC, USA: IEEE Computer Society, 1997, pp. 178–. [Online]. Available: http://dl.acm.org/citation.cfm?id=792756.792862*

[29] *C. Batty, F. Bertails, and R. Bridson, "A fast variational framework for accurate solid-fluid coupling," ACM Trans. Graph., vol. 26, no. 3, Jul. 2007.*

[30] *R. Bridson, Fluid simulation for computer graphics. Boca Raton: CRC Press, Taylor & Francis Group, CRC Press is an imprint of the Taylor & Francis Group, an informa Business, 2016.*

[31] *C. Hirt, A. A. Amsden, and J. Cook, "An arbitrary lagrangian-eulerian computing method for all flow speeds," Journal of computational physics, vol. 14, no. 3, pp. 227–253, 1974.*

[32] *M. Shah, J. M. Cohen, S. Patel, P. Lee, and F. Pighin, "Extended galilean invariance for adaptive fluid simulation," in Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation. Eurographics Association, 2004, pp. 213–221.*

[33] *N. Rasmussen, D. Enright, D. Nguyen, S. Marino, N. Sumner, W. Geiger, S. Hoon, and R. Fedkiw, "Directable photorealistic liquids," in Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation. Eurographics Association, 2004, pp. 193–202.*

[34] *B. M. Klingner, B. E. Feldman, N. Chentanez, and J. F. O'brien, "Fluid animation with dynamic meshes," in ACM Transactions on Graphics (TOG), vol. 25, no. 3. ACM, 2006, pp. 820–825.*

[35] *J. Donea, A. Huerta, J.-P. Ponthot et al., "Arbitrary lagrangian eulerian methods," Encyclopedia of Computational Mechanics, pp. Chapter–14, 2004.*

[36] *C. Batty, S. Xenos, and B. Houston, "Tetrahedral embedded boundary methods for accurate and flexible adaptive fluids," in Proceedings of Eurographics, 2010.*

[37] *P. Alliez, D. Cohen-Steiner, M. Yvinec, and M. Desbrun, "Variational tetrahedral meshing," ACM Trans. Graph., vol. 24, no. 3, pp. 617–625, Jul. 2005.*

[38] *F. Labelle and J. R. Shewchuk, "Isosurface stuffing: Fast tetrahedral meshes with good dihedral angles," ACM Trans. Graph., vol. 26, no. 3, Jul. 2007. [Online]. Available: http://doi.acm.org/10.1145/1276377.1276448*

[39] *R. Setaluri, M. Aanjaneya, S. Bauer, and E. Sifakis, "Spgrid: A sparse paged grid structure applied to adaptive smoke simulation," ACM Trans. Graph., vol. 33, no. 6, pp. 205:1–205:12, Nov. 2014. [Online]. Available: http://doi.acm.org/10.1145/2661229.2661269*

[40] *M. B. Nielsen and R. Bridson, "Spatially adaptive flip fluid simulations in bifrost," in ACM SIGGRAPH 2016 Talks, ser. SIGGRAPH '16. New York, NY, USA: ACM, 2016, pp. 41:1–41:2. [Online]. Available: http://doi.acm.org/10.1145/2897839.2927399*

[41] *M. K. Misztal, K. Erleben, A. Bargteil, J. Fursund, B. B. Christensen, J. A. Bærentzen, and R. Bridson, "Multiphase flow of immiscible fluids on unstructured moving meshes," in Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation, ser. SCA '12. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2012, pp. 97–106.*

[42] *K. Erleben, M. K. Misztal, and J. A. Bærentzen, "Mathematical foundation of the optimization-based fluid animation method," in Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, ser. SCA '11. New York, NY, USA: ACM, 2011, pp. 101–110. [Online]. Available: http://doi.acm.org/10.1145/2019406.2019420*

[43] *K. Museth, J. Lait, J. Johanson, J. Budsberg, R. Henderson, M. Alden, P. Cucka, D. Hill, and A. Pearce, "Openvdb: An open-source data structure and toolkit for high-resolution volumes," in ACM SIGGRAPH*

2013 Courses, ser. SIGGRAPH '13. New York, NY, USA: ACM, 2013, pp. 19:1–19:1.

[44] *M. Aanjaneya, M. Gao, H. Liu, C. Batty, and E. Sifakis, "Power diagrams and sparse paged grids for high resolution adaptive liquids," ACM Trans. Graph., vol. 36, no. 4, Jul. 2017.*

[45] *J. Brackbill and H. Ruppel, "Flip: A method for adaptively zoned, particle-in-cell calculations of fluid flows in two dimensions," Journal of Computational Physics, vol. 65, no. 2, pp. 314 – 343, 1986. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0021999186902111*

[46] *V. C. Azevedo and M. M. Oliveira, "Efficient smoke simulation on curvilinear grids," Computer Graphics Forum, vol. 32, no. 7, pp. 235–244, 2013. [Online]. Available: http://dx.doi.org/10.1111/cgf.12231*

[47] *B. Zhu, W. Lu, M. Cong, B. Kim, and R. Fedkiw, "A new grid structure for domain extension," ACM Trans. Graph., vol. 32, no. 4, pp. 63:1–63:12, Jul. 2013.*

[48] *G. Irving, E. Guendelman, F. Losasso, and R. Fedkiw, "Efficient simulation of large bodies of water by coupling two and three dimensional techniques," ACM Trans. Graph., vol. 25, no. 3, pp. 805–811, Jul. 2006.*

[49] *N. Chentanez and M. Müller, "Real-time eulerian water simulation using a restricted tall cell grid," ACM Trans. Graph., vol. 30, no. 4, pp. 82:1–82:10, Jul. 2011.*

[50] *R. Gal, O. Sorkine, and D. Cohen-Or, "Feature-aware texturing," Rendering Techniques, vol. 2006, p. 17th, 2006.*

[51] *P. Krähenbühl, M. Lang, A. Hornung, and M. Gross, "A system for retargeting of streaming video," in ACM Transactions on Graphics (TOG), vol. 28, no. 5. ACM, 2009, p. 126.*

[52] *M. Lang, A. Hornung, O. Wang, S. Poulakos, A. Smolic, and M. Gross, "Nonlinear disparity mapping for stereoscopic 3d," ACM Transactions on Graphics (TOG), vol. 29, no. 4, p. 75, 2010.*

[53] *Y. Dobashi, Y. Matsuda, T. Yamamoto, and T. Nishita, "A fast simulation method using overlapping grids for interactions between smoke and rigid objects," Computer Graphics Forum, vol. 27, no. 2, pp. 477–486, 2008.*

[54] *R. E. English, L. Qiu, Y. Yu, and R. Fedkiw, "Chimera grids for water simulation," in Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation, ser. SCA '13. New York, NY, USA: ACM, 2013, pp. 85–94.*

[55] *J. M. Owen, J. V. Villumsen, P. R. Shapiro, and H. Martel, "Adaptive Smoothed Particle Hydrodynamics: Methodology. II." The Astrophysical Journal Supplement Series, vol. 116, no. 2, pp. 155–209, Jun. 1998.*

[56] *M. Lastiwka, N. Quinlan, and M. Basa, "Adaptive particle distribution for smoothed particle hydrodynamics," International Journal for Numerical Methods in Fluids, vol. 47, no. 10-11, pp. 1403–1409, 2005. [Online]. Available: http://dx.doi.org/10.1002/fld.891*

[57] *Y. Zhang, B. Solenthaler, and R. Pajarola, "Adaptive sampling and rendering of fluids on the gpu," in Proceedings of the Fifth Eurographics / IEEE VGTC Conference on Point-Based Graphics, ser. SPBG'08. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2008, pp. 137–146.*

[58] *B. Adams, M. Pauly, R. Keiser, and L. J. Guibas, "Adaptively sampled particle fluids," ACM Trans. Graph., vol. 26, no. 3, Jul. 2007. [Online]. Available: http://doi.acm.org/10.1145/1276377.1276437*

[59] *B. Solenthaler and M. Gross, "Two-scale particle simulation," ACM Trans. Graph., vol. 30, no. 4, pp. 81:1–81:8, Jul. 2011. [Online]. Available: http://doi.acm.org/10.1145/2010324.1964976*

[60] *A. Treuille, A. Lewis, and Z. Popović, "Model reduction for real-time fluids," ACM Trans. Graph., vol. 25, no. 3, pp. 826–834, Jul. 2006.*

[61] *T. De Witt, C. Lessig, and E. Fiume, "Fluid simulation using laplacian eigenfunctions," ACM Trans. Graph., vol. 31, no. 1, pp. 10:1–10:11, Feb. 2012.*

[62] *E. Edwards and R. Bridson, "Detailed water with coarse grids: Combining surface meshes and adaptive discontinuous galerkin," ACM Trans. Graph., vol. 33, no. 4, pp. 136:1–136:9, Jul. 2014. [Online]. Available: http://doi.acm.org/10.1145/2601097.2601167*

[63] *D. Demidov, "Amgcl: C++ library for solving large sparse linear systems with algebraic multigrid method (https://github.com/ddemidov/amgcl)," 2009.*

[64] *R. Ando, N. Thürey, and C. Wojtan, "A dimension-reduced pressure solver for liquid simulations," in Computer Graphics Forum, vol. 34, no. 2. Wiley Online Library, 2015, pp. 473–480.*

[65] *D. Enright, F. Losasso, and R. Fedkiw, "A fast and accurate semi-lagrangian particle level set method," Comput. Struct., vol. 83, no. 6-7, pp. 479–490, Feb. 2005. [Online]. Available: http://dx.doi.org/10.1016/j.compstruc.2004.04.024*

[66] *C. Batty, "A cell-centred finite volume method for the poisson problem on non-graded quadtrees with second order accurate gradients," J. Comput. Phys., vol. 331, no. C, pp. 49–72, Feb. 2017. [Online]. Available: https://doi.org/10.1016/j.jcp.2016.11.035*

[67] *H. Liu, N. Mitchell, M. Aanjaneya, and E. Sifakis, "A scalable schur-complement fluids solver for heterogeneous compute platforms," ACM Trans. Graph., vol. 35, no. 6, pp. 201:1–201:12, Nov. 2016.*

[68] *A. McAdams, E. Sifakis, and J. Teran, "A parallel multigrid poisson solver for fluids simulation on large grids," in Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, ser. SCA '10. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2010, pp. 65–74. [Online]. Available: http://dl.acm.org/citation.cfm?id=1921427.1921438*

[69] *B. Zhu, E. Quigley, M. Cong, J. Solomon, and R. Fedkiw, "Codimensional surface tension flow on simplicial complexes," ACM Trans. Graph., vol. 33, no. 4, pp. 111:1–111:11, Jul. 2014. [Online]. Available: http://doi.acm.org/10.1145/2601097.2601201*

[70] *Y. Fan, J. Litven, D. I. W. Levin, and D. K. Pai, "Eulerian-on-lagrangian simulation," ACM Trans. Graph., vol. 32, no. 3, pp. 22:1–22:9, Jul. 2013.*

[71] *B. Van Opstal, L. Janin, K. Museth, and M. Aldén, "Large scale simulation and surfacing of water and ice effects in dragons 2," in ACM SIGGRAPH 2014 Talks, ser. SIGGRAPH '14. New York, NY, USA: ACM, 2014, pp. 11:1–11:1. [Online]. Available: http://doi.acm.org/10.1145/2614106.2614156*

[72] *J. Budsberg, M. Losure, K. Museth, and M. Baer, "Liquids in the croods," ACM DigiPro., 2013.*

[73] *A. W. Marshall and I. Olkin, "Norms and inequalities for condition numbers, iii," Linear Algebra and its Applications, vol. 7, no. 4, pp. 291–300, 1973.*