An Efficient Geometric Multigrid Solver for Viscous Liquids

MRIDUL AANJANEYA*, Rutgers University
CHENGGUIZI HAN, Rutgers University
RYAN GOLDADE, University of Waterloo
CHRISTOPHER BATTY, University of Waterloo

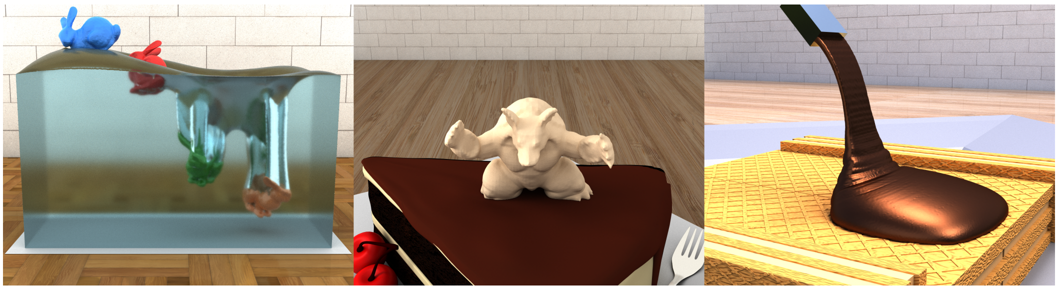


Fig. 1. (Left) Four bunnies of different masses are two-way coupled to a viscous liquid ($384^2 \times 192$ grid). (Middle) A creamy armadillo cake topping collapses under gravity (512^3 grid). (Right) Chocolate buckles as it pours onto a wafer ($256^2 \times 512$ grid).

We present an efficient geometric Multigrid solver for simulating viscous liquids based on the variational approach of Batty and Bridson [2008]. Although the governing equations for viscosity are elliptic, the strong coupling between different velocity components in the discrete stencils mandates the use of more exotic smoothing techniques to achieve textbook Multigrid efficiency. Our key contribution is the design of a novel box smoother involving small and sparse systems (at most 9×9 in 2D and 15×15 in 3D), which yields excellent convergence rates and performance improvements of 3.5× - 13.8× over a naïve Multigrid approach. We employ a hybrid approach by using a direct solver only inside the box smoother and keeping the remaining pipeline assembly-free, allowing our solver to efficiently accommodate more than 194 million degrees of freedom, while occupying a memory footprint of less than 16 GB. To reduce the computational overhead of using the box smoother, we precompute the Cholesky factorization of the subdomain system matrix for all interior degrees of freedom. We describe how the variational formulation, which requires volume weights computed at the centers of cells, edges, and faces, can be naturally accommodated in the Multigrid hierarchy to properly enforce boundary conditions. Our proposed Multigrid solver serves as an excellent preconditioner for Conjugate Gradients, outperforming existing state-of-the-art alternatives. We demonstrate the efficacy of our method on several high resolution examples of viscous liquid motion including two-way coupled interactions with rigid bodies.

CCS Concepts: • **Computing methodologies** → **Physical simulation**.

*Joint first and last author.

Authors' addresses: Mridul Aanjaneya, Rutgers University; Chengguizi Han, Rutgers University; Ryan Goldade, University of Waterloo; Christopher Batty, University of Waterloo.

2577-6193/2019/7-ART14 $15.00
https://doi.org/10.1145/3340255

Additional Key Words and Phrases: viscosity, liquid simulation, variational approach, Multigrid solver

ACM Reference Format:
Mridul Aanjaneya, Chengguizi Han, Ryan Goldade, and Christopher Batty. 2019. An Efficient Geometric Multigrid Solver for Viscous Liquids. *Proc. ACM Comput. Graph. Interact. Tech.* 2, 2, Article 14 (July 2019), 21 pages. https://doi.org/10.1145/3340255

1 INTRODUCTION

Liquids such as honey, syrup, glue, paints, molasses, and cake batter are ubiquitous in our daily lives. Unlike inviscid fluids where the dominant flow features (such as splashes) result from turbulence in the underlying velocity field, viscous liquids exhibit greater resistance to shearing flow. This results in laminar or damped motion in the interior of the liquid, yet gives rise to a host of interesting surface details such as liquid buckling and folding over itself. Thus, there is a strong demand for computational methods that can realistically reproduce such effects [Rasmussen et al. 2004; Ruilova 2007; Wiebe and Houston 2004]. Motivated by this need, Batty and Bridson [2008] proposed an implicit variational discretization that supports the rotational free surface motion that is necessary for such flows. Their method has also been integrated into the Houdini animation software [SideFX 2019]. However, to achieve these effects, their approach introduces cross-derivative terms that couple all three components of velocity together. This makes it computationally challenging to solve the resulting linear system, as compared to the scalar Poisson equation for pressure projection: the matrix is three times larger and it is not an M-matrix, the class of matrices for which standard solvers such as incomplete Cholesky preconditioned Conjugate Gradients are known to work well. In practice, the viscosity step is the major computational bottleneck in the simulation pipeline for flows with significant viscosity coefficients.

Therefore, our focus in this paper is on the design of a fast Multigrid solver for the implicit viscosity system arising in the variational formulation of Batty and Bridson [2008]. While great progress has been made on the design of fast solvers for pressure projection [Aanjaneya et al. 2017; Ando et al. 2015; Chu et al. 2017; Liu et al. 2016; McAdams et al. 2010; Molemaker et al. 2008; Setaluri et al. 2014; Zhang and Bridson 2014], to the best of our knowledge, ours is the first work to consider a fast solver for the viscosity system.

Two building blocks are essential for a geometric Multigrid solver: (1) a smoothing routine to remove high frequency errors in the solution; and (2) upsampling/downsampling operators to transfer information between grid levels. The same principles used to design the transfer operators in Multigrid solvers for pressure projection [McAdams et al. 2010] also apply to the viscosity problem. However, because the discrete finite difference stencils in the viscosity system strongly couple different velocity components, an appropriate smoother must be carefully designed for this specific application. Indeed, as we demonstrate in Section 7, the damped-Jacobi smoother, which is the standard choice for Multigrid pressure projection, performs extremely poorly for the viscosity problem. Our key contribution is the design of a novel box smoother that leads to fairly small and sparse systems (at most 9×9 in 2D and 15×15 in 3D); this enables the Multigrid solver to achieve excellent convergence rates and speedups of $3.5\times$ - $13.8\times$ over a naïve Multigrid approach.

For maximum computational efficiency, we adopt a hybrid approach that uses a direct solver inside the box smoother and keeps the remaining pipeline assembly-free, allowing our solver to efficiently accommodate more than 194 million degrees of freedom, while occupying a memory footprint of less than 16 GB. To reduce the computational overhead of our box smoother, we pre-factorize the subdomain system matrix for all interior degrees of freedom. Our Multigrid solver serves as an excellent preconditioner to Conjugate Gradients, reduces the residual by at least an order of magnitude *every* iteration, and outperforms existing state-of-the-art solutions. The variational approach of Batty and Bridson [2008] involves the use of volume fraction weights

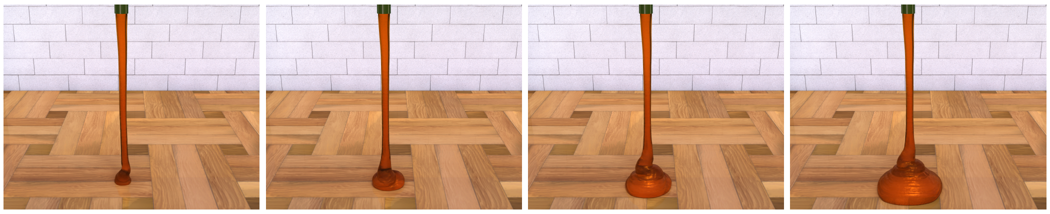


Fig. 2. A source pours honey on the floor, which displays buckling and folding patterns (simulated using a $192^2 \times 384$ grid).

computed at the cell, edge, and face centers of the background Eulerian grid to properly enforce the free surface boundary conditions. However, these weights are not well-defined at coarser levels of the Multigrid hierarchy as, by definition, these levels cannot capture all topological features in the computational domain. Our final contribution is a strategy to naturally accommodate this slight discrepancy while maintaining a computationally efficient Multigrid V-cycle.

In summary, our main technical contributions are as follows:

- a novel box smoother for the implicit viscosity system of Batty and Bridson [2008];
- a memory-efficient geometric Multigrid solver for liquid viscosity that utilizes our novel box smoother;
- enhancements to accommodate the variational approach for properly enforcing boundary conditions; and
- high resolution simulations of viscous liquids deforming and undergoing two-way coupled interactions with rigid bodies.

2 RELATED WORK

Our work is based on the staggered grid variable layout and advection/viscosity/pressure splitting approach that are common in hybrid/Eulerian fluid animation, as summarized by Bridson [2015]. Our review emphasizes this family of simulation methods; however, there is also substantial recent research on implicit viscosity treatments for smoothed particle hydrodynamics [Barreiro et al. 2017; Bender and Koschier 2017; Peer et al. 2015; Takahashi et al. 2015; Weiler et al. 2018].

2.1 Eulerian viscosity

By assuming constant viscosity coefficients, incompressibility, and simplified interface conditions, early work on animating viscous flow modeled viscosity as a componentwise velocity diffusion problem, i.e., one heat equation per velocity component. Foster and Metaxas adopted an explicit discretization of this model [Foster and Metaxas 1996] which was later supplanted by implicit discretizations for the sake of greater stability [Carlson et al. 2002; Fält and Roble 2003; Stam 1999]. Rasmussen et al. [2004] observed that in the presence of spatially varying viscosity, additional stress terms arise that couple the distinct velocity components together. They suggested a hybrid implicit-explicit scheme to reduce the cost of this coupling. Batty and Bridson [2008] showed that the coupled variable viscosity system could be discretized with a variational strategy that yields a symmetric positive definite (SPD) system. This approach has the additional benefit of naturally enforcing the zero-traction free surface boundary condition. This condition on the fluid stress is necessary for proper rotational behavior, even for constant viscosity coefficients, and is critical to plausible bending behavior in high viscosity liquids. Larionov et al. [2017] subsequently showed that, at the cost of tightly coupling viscosity and pressure and solving a much larger SPD system, slightly more physical surface behavior could be achieved, including recovering the classic liquid rope

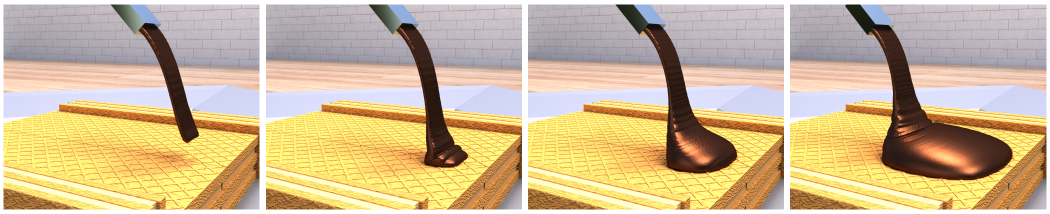


Fig. 3. A source pours chocolate that displays buckling patterns as it falls on the wafer (simulated using a $256^2 \times 512$ grid).

coiling instability. A viscous dissipation model similar to that of Batty and Bridson [2008] has also been adopted in the context of material point methods [Ram et al. 2015], albeit on co-located grids. More generally, models for (visco-)elastic fluids and solids often involve similar terms [Goktekin et al. 2004; Losasso et al. 2006], particularly for damping, due to the natural relationship between stress and strain rate (e.g., [Bergou et al. 2010]).

Tetrahedral schemes have also been proposed that support realistic viscosity, under both Eulerian [Batty and Houston 2011] and Lagrangian [Clausen et al. 2013; Erleben et al. 2011] frameworks, including spatial adaptivity to reduce computational cost. However, the less-structured nature of tetrahedral meshes makes them ill-suited to geometric Multigrid. Algebraic Multigrid (AMG) has been successfully applied to Poisson problems on tetrahedral meshes [Chentanez et al. 2007], although AMG is less readily parallelizable and also less optimal in settings where geometric Multigrid is feasible.

Scope. Because of the significant additional cost of Larionov's Stokes approach [Larionov et al. 2017] and the broader adoption of Batty and Bridson's decoupled method [Batty and Bridson 2008], we adopt the latter in the present work. As these authors noted, their model introduces new challenges for efficient numerical solution compared to earlier Laplacian-type discretizations because the cross-derivative terms yield a larger (albeit still symmetric positive definite) system, and more importantly, they introduce negative off-diagonal coefficients. Therefore, unlike finite difference discretizations of the Poisson or diffusion problems, the matrix is no longer an M-matrix and standard preconditioners for Conjugate Gradients, such as modified incomplete Cholesky, do not perform as effectively. As such, our goal is to develop an efficient Multigrid preconditioner for this problem.

2.2 Fast solvers

Fast linear system solvers like Multigrid [Trottenberg et al. 2000] and domain decomposition [Valli et al. 1999] have a long history in computational mathematics, and have proven quite effective for a variety of fluid animation problems over the last decade or so. Most of these efforts have focused on solving the Poisson problem to enforce incompressibility. An early example is work by Bolz et al. [2003] who applied Multigrid to Stam's classic stable fluid solver [Stam 1999] in two dimensions. Molemaker et al. [2008] applied Multigrid for smoke simulation on rectangular domains, using additional alternating orthogonal projections to enforce voxelized solid boundaries. McAdams et al. [2010] presented a Multigrid solver for voxelized projection with general Neumann (solid) and Dirichlet (free surface) boundaries. Dick et al. [2016] employed a graph-based coarsening strategy to avoid the topological discrepancies typically introduced in the geometric Multigrid hierarchy. Their approach outperforms competing geometric Multigrid methods in domains with many thin solid boundaries (e.g., mazes) at the expense of additional memory overhead and implementation

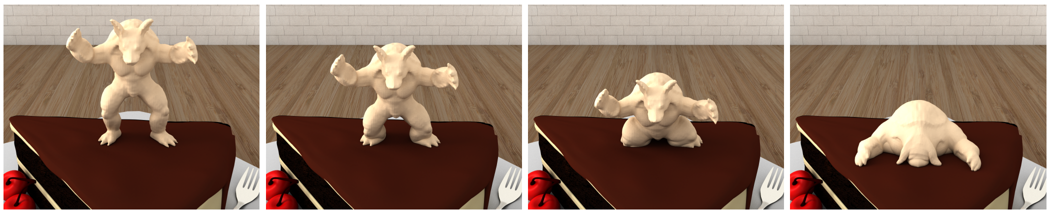


Fig. 4. A creamy armadillo topping on a cake collapses under its own weight (simulated using a 512^3 background grid).

complexity. Chentanez and co-authors [2011] applied Multigrid to a tall-cell grid problem with sub-grid boundary geometry to achieve real-time rates, later adding support for separating boundary conditions [Chentanez and Mueller-Fischer 2012]. Weber et al. [2015] achieved faster convergence by constructing higher levels in the hierarchy in a manner compatible with a cut-cell approach. Ferstl et al. [2014] applied Multigrid to an octree-based co-located cut-cell finite element solution of the Poisson problem for liquids. In a similar vein, Setaluri et al. [2014] and Aanjaneya et al. [2017] developed Multigrid solvers for octree finite volume discretizations of the Poisson problem for smoke and liquids, respectively, exploiting sparse grid structures.

Multigrid has been applied to other (non-fluid) animation tasks as well. Zhu et al. [2010] developed an efficient Multigrid scheme for linear and co-rotational elasticity on staggered grids. Unlike our viscosity problem, which is always positive definite, elasticity involves indefinite systems which complicates the design of a viable Multigrid scheme; Zhu et al. authors addressed this challenge with an augmented form of the problem, a distributive smoothing technique, and a specially-designed symmetric smoother near boundaries. This technique was later extended to character skinning with contact and collisions [McAdams et al. 2011]. Tamstorf et al. [2015] developed a smoothed aggregation variant of algebraic Multigrid tailored to large-scale cloth simulation. Because fast solution of linear systems is a perennial problem across computer graphics as a whole, many more applications have benefited from Multigrid strategies, and we shall not review them all; however, an illustrative example is the recent work of Kazhdan and Hoppe [2018], who proposed a general-purpose adaptive Multigrid solver using Gauss-Seidel smoothing for diverse finite element problems and considered applications to surface reconstruction, gradient-domain image stitching, and distance field computation.

Domain decomposition is another powerful acceleration technique, which adopts a divide and conquer strategy by subdividing the domain into smaller decoupled components that can be solved independently and in parallel (*divide* step), and subsequently re-combined (*conquer* step). This approach was demonstrated both by Liu et al. [2016], who advocated a hybrid GPU/CPU approach, and Chu et al. [2017], who applied a recursive decomposition and distinct solver types on the resulting sub-domains. Other common but largely orthogonal techniques for accelerating fluid solvers include h- or p-adaptivity (e.g., [Edwards and Bridson 2014; Losasso et al. 2004]) and GPU implementation [Bolz et al. 2003; Wu et al. 2018]. Most recently, Goldade et al. [2019] proposed a symmetric, variational octree extension of the method of Batty and Bridson [2008].

3 BACKGROUND

We first review the underlying theory behind Multigrid [Trottenberg et al. 2000], which is a *multi-scale* method for solving elliptic partial differential equations. Suppose we wish to solve a system of equations $\boldsymbol{A}_h \boldsymbol{x}_h = \boldsymbol{b}_h$, where $\boldsymbol{A}_h$ is an elliptic operator on a regular grid with voxel size h. Assuming an initial guess $\boldsymbol{x}_h^0$, one approach is to first compute the residual $\boldsymbol{r}_h = \boldsymbol{b}_h - \boldsymbol{A}_h \boldsymbol{x}_h^0$, and solve a new system, $\boldsymbol{A}_h \boldsymbol{e}_h = \boldsymbol{r}_h$, for the *error* $\boldsymbol{e}_h$. Subsequently, the exact solution $\boldsymbol{x}_h^\star$ can be computed by adding together $\boldsymbol{x}_h^0$ and $\boldsymbol{e}_h$ to produce:

$$\boldsymbol{x}_h^\star = \boldsymbol{x}_h^0 + \boldsymbol{e}_h = \boldsymbol{x}_h^0 + \boldsymbol{A}_h^{-1}\boldsymbol{r}_h = \boldsymbol{x}_h^0 + \boldsymbol{A}_h^{-1}(\boldsymbol{b}_h - \boldsymbol{A}_h\boldsymbol{x}_h^0) = \boldsymbol{A}_h^{-1}\boldsymbol{b}_h. \tag{1}$$

Notice that this approach is as expensive as solving the original system $\boldsymbol{A}_h\boldsymbol{x}_h = \boldsymbol{b}_h$. However, when the operator $\boldsymbol{A}_h$ is elliptic, then the following property holds: *for small residual $\boldsymbol{r}_h$, the error $\boldsymbol{e}_h$ is always smooth*, where the "smallness" of $\boldsymbol{r}_h$ is measured by an appropriate vector norm (e.g., either the L_2 or L_∞ norms). The smoothness of the error implies that it can be faithfully approximated on a coarser grid as well, with voxel size $2h$. Based on this principle, Multigrid solvers employ the following algorithm:

(1) Use a *smoothing routine* to reduce the norm of $\boldsymbol{r}_h$.
(2) *Downsample* $\boldsymbol{r}_h$ onto a grid with voxel size $2h$.
(3) Solve the system $\boldsymbol{A}_{2h}\boldsymbol{e}_{2h} = \boldsymbol{r}_{2h}$.
(4) *Upsample* $\boldsymbol{e}_{2h}$ back onto the original grid with voxel size h.
(5) Compute $\boldsymbol{x}_h^{\text{new}} = \boldsymbol{x}_h^{\text{old}} + \boldsymbol{e}_h$.
(6) Use the smoothing routine again to reduce the residual.

For clarity of exposition, the above algorithm is a simplified version of Multigrid on only two levels, but the same technique can be recursively applied to step (3) to obtain the general algorithm for an arbitrary number of levels. The *smoothing routine* mentioned in steps (1) and (6) is a "lightweight" operator that can quickly reduce the norm of the residual. For the standard 7-point Laplacian operator, a damped-Jacobi operator is a very effective smoother, which is also highly parallel [McAdams et al. 2010]. After the smoothing operation, the current residual $\boldsymbol{r}_h$ is *downsampled* onto a coarser grid with voxel size $2h$ to compute $\boldsymbol{r}_{2h}$ (step (2)). Subsequently, the system $\boldsymbol{A}_{2h}\boldsymbol{e}_{2h} = \boldsymbol{r}_{2h}$ is solved for the error $\boldsymbol{e}_{2h}$ on the coarse grid (step (3)). Note that this system is much smaller than the original system on the fine grid, and so its solution can be computed more efficiently. Moreover, we are assured from the property of ellipticity that $\boldsymbol{e}_{2h}$ is a faithful approximation to the original error $\boldsymbol{e}_h$ on the fine grid. The error $\boldsymbol{e}_{2h}$ is upsampled on the fine grid (step (4)) and added to the solution vector (step (5)), after which the smoothing routine is applied once again to decrease the norm of the residual (step (6)).

Steps (1) – (6) together comprise a *Multigrid V-cycle*. Depending on the effort spent in the smoothing steps (1) and (6), as well as the accuracy of the solve in step (3), one or more iterations of a V-cycle may be required to reduce the norm of the residual below a specified threshold. Notice that the entire Multigrid algorithm is hinged on the premise that the smoothing operator will *effectively* reduce the norm of the residual, after which the error can be approximated on a coarser grid based on the property of ellipticity. In the ideal case with a perfect smoother, a Multigrid solver should produce *resolution-independent* convergence rates. However, in the unfortunate circumstance when the smoother is poor, the remaining steps in the Multigrid algorithm do not work well, and the convergence behavior of the entire V-cycle degrades to that of the smoother. As shown in Section 7, the damped-Jacobi operator is a poor smoother for the viscosity system of Batty and Bridson [2008], motivating the need for a new smoother, as proposed in Section 6.3.

4 GOVERNING EQUATIONS

We review the governing laws that describe solid and fluid motion.

4.1 Fluid Equations

Consider the incompressible Navier-Stokes equations

$$\vec{u}_t + (\vec{u} \cdot \nabla)\vec{u} + \frac{\nabla p}{\rho} = \frac{1}{\rho}\nabla \cdot \tau + \vec{f} \tag{2}$$

$$\nabla \cdot \vec{u} = 0 \tag{3}$$

$$\tau = \mu(\nabla\vec{u} + (\nabla\vec{u})^T) \tag{4}$$

where ρ is density, $\vec{u}$ is velocity, p is pressure, μ is dynamic viscosity, $\vec{f}$ comprises accelerations due to external forces (such as gravity), and τ is the viscous shear stress tensor. We discretize these equations on a staggered Cartesian grid [Harlow and Welch 1965] and use standard operator splitting [Stam 1999]. We first employ the basic semi-Lagrangian method [Stam 1999] to solve for advection

$$\frac{\vec{u}^\star - \vec{u}^n}{\Delta t} + (\vec{u}^n \cdot \nabla)\vec{u}^n = \vec{f}^n \tag{5}$$

taking care to clip backward-cast rays near solid objects [Guendelman et al. 2005]. Next, we solve the pressure Poisson equation and update the velocity:

$$\vec{u}^{\star\star} = \vec{u}^\star - \Delta t \frac{\nabla p}{\rho}. \tag{6}$$

We then solve for viscous effects following Batty and Bridson [2008]

$$\frac{\vec{u}^\dagger - \vec{u}^{\star\star}}{\Delta t} = \nabla \cdot \tau \tag{7}$$

and finally, project velocities $\vec{u}^\dagger$ to be divergence-free once again. We use the particle level set method [Enright et al. 2005] and the reinitialization scheme of Losasso et al. [2005] for interface tracking, the velocity extrapolation of Adalsteinsson and Sethian [1999], and second order accurate pressure projection [Enright et al. 2003] (though our contributions do not require these particular choices). For two-way coupling rigid bodies with fluids, we use the impulse-based monolithic formulation of Robinson-Mosher et al. [2011] and efficiently solve the coupled system using the Multigrid-style approach of Aanjaneya [2018].

4.2 Solid Equations

We use Newton's laws of motion to evolve the rigid bodies

$$\begin{array}{ll} \vec{x}_t = \vec{v}, & q_t = \frac{1}{2}\vec{\omega}q \\ \vec{v}_t = \vec{F}/m, & \vec{L}_t = \vec{\alpha} \end{array} \tag{8}$$

where $\vec{x}$ is position, q is orientation (in quaternions), m is mass, $\vec{v}$ is linear velocity, $\vec{\omega}$ is angular velocity, $\vec{F}$ is net force, $\vec{\alpha}$ is net torque, and $\vec{L} = I\vec{\omega}$ is angular momentum with inertia tensor $I = RDR^T$ (R is the world space orientation matrix and D is the diagonal inertia tensor in object space). We follow Guendelman et al. [2003] for collisions and contact.

5 NUMERICAL DISCRETIZATION

Batty and Bridson [2008] express the solution to a backward Euler discretization of (7) as the minimum of a convex energy functional

$$\iiint_\Omega \left(\rho \|\vec{u} - \vec{u}^{\star\star}\|_2^2 + 2\mu\Delta t \left\| \frac{\nabla\vec{u} + (\nabla\vec{u})^\intercal}{2} \right\|_F^2 \right) dV \tag{9}$$

where Ω is the liquid domain and $||\cdot||_F$ denotes the Frobenius norm. The benefit of using equation (9) is that the zero-traction free surface boundary conditions are naturally enforced by this minimization procedure, while the simpler no-slip boundary conditions near solid walls can be applied explicitly to the discrete equations.

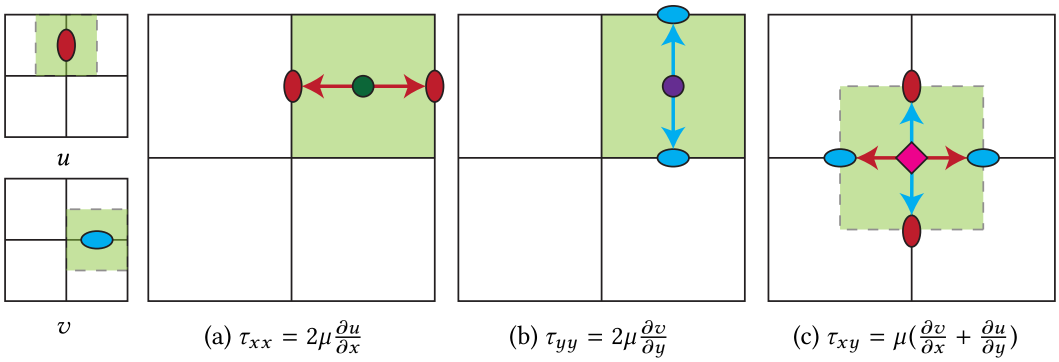

u
v
(a) $\tau_{xx} = 2\mu\frac{\partial u}{\partial x}$
(b) $\tau_{yy} = 2\mu\frac{\partial v}{\partial y}$
(c) $\tau_{xy} = \mu(\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y})$

Fig. 5. Variable locations for velocities (far left) and stresses (a-c) in 2D, with control volumes shaded in light green. Stress stencils are illustrated with colored arrows.

To discretize this formulation on staggered grids, centered differences are used to approximate quantities inside the integral at the sample locations. The contributions of sample points are scaled by the amount of liquid inside the corresponding cubic control volumes and summed to compute the total discrete energy, and finally the total energy is minimized with respect to velocity.

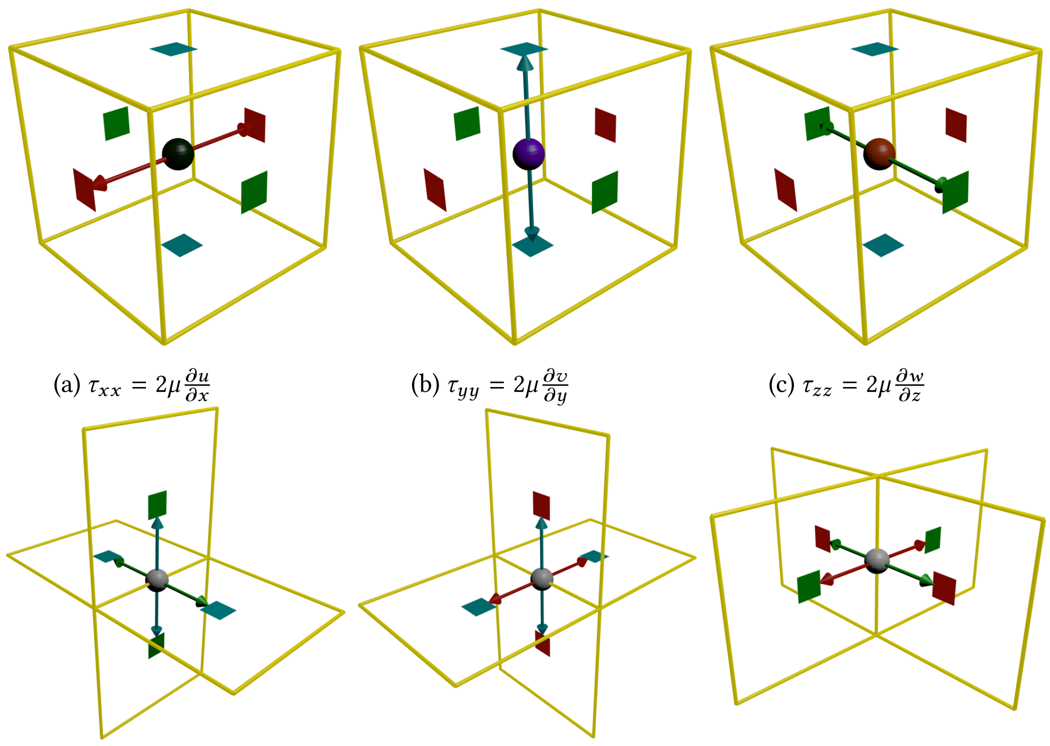
Figures 5 and 6 illustrate this approach in both 2D and 3D, where face-centered velocities u, v, w are shown in red, cyan, and light green, cell-centered stresses $\tau_{xx}, \tau_{yy}, \tau_{zz}$ are shown in dark green, purple, and orange, and edge-centered stresses $\tau_{xy}, \tau_{yz}, \tau_{xz}$ are shown in gray. Nodal stresses τ_{xy} in 2D are shown with magenta diamonds. The first term inside the integral in equation (9) is evaluated at face centers. The second term corresponds to the stress whose components live at centers of cells and nodes (resp. edges) in 2D (resp. 3D) and are computed using finite differences. The resulting discrete energy takes the form

$$(\boldsymbol{u} - \boldsymbol{u}^{\star\star})^T V_{\boldsymbol{u}} M(\boldsymbol{u} - \boldsymbol{u}^{\star\star}) + 2\Delta t \boldsymbol{u}^T D^T S K V_{\tau} D \boldsymbol{u} \tag{10}$$

where M is a diagonal matrix of per-velocity densities, Δt is the time step, $V_{\boldsymbol{u}}$ (resp. V_τ) is a diagonal matrix of liquid control volumes per velocity (resp. stress) sample, D is the finite difference deformation rate operator such that $D \approx (\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^T)/2$, K is a diagonal matrix of viscosity coefficients, and S is a diagonal matrix of scale factors to correctly account for the contribution of cross-derivative terms to produce the Frobenius norm. Since the discrete energy in equation (10) is convex and quadratic in $\boldsymbol{u}$, differentiating and equating to zero yields the following symmetric and positive definite system:

$$(V_{\boldsymbol{u}} M + 2\Delta t D^T S K V_\tau D)\boldsymbol{u} = V_{\boldsymbol{u}} M \boldsymbol{u}^{\star\star}. \tag{11}$$

Up to a constant scaling factor, the resulting discrete system of equations matches a direct finite difference discretization of the partial differential equation in equation (7) deep in the interior of the liquid, but automatically handles the free surface boundary condition through the use of the control volume weights.



(a) $\tau_{xx} = 2\mu\frac{\partial u}{\partial x}$ (b) $\tau_{yy} = 2\mu\frac{\partial v}{\partial y}$ (c) $\tau_{zz} = 2\mu\frac{\partial w}{\partial z}$

(d) $\tau_{xy} = \mu(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x})$ (e) $\tau_{yz} = \mu(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y})$ (f) $\tau_{xz} = \mu(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x})$

Fig. 6. Variable locations for face velocities in red, cyan, and light green, (a-c) cell-centered stresses in dark green, purple, and orange, and (d-f) edge-centered stresses in 3D in gray. Stress stencils are illustrated with colored arrows.

6 MULTIGRID SOLVER

We follow the convention of McAdams et al. [2010] to discretize the simulation domain (Figure 7(left)) into voxels (Figure 7(right)). Each grid voxel's label is either `FLUID` (blue), `SOLID` (green), or `AIR` (red), depending on whether its cell center lies inside a liquid, solid, or air region. To simplify implementation we keep a "ghost" layer padding of `SOLID` voxels along the domain boundary. For convenience, we assume that the finest grid size is divisible by 2^{k-1} along each axis, where k is the number of levels in the Multigrid hierarchy.

6.1 Multigrid Hierarchy

We adopt the same coarsening strategy as Zhu et al. [2010], where each coarse voxel is marked as `SOLID` if any one of its fine children (4 in 2D and 8 in 3D) is marked as `SOLID`. Otherwise, it is marked as `FLUID` if any one of its fine children is marked as `FLUID`. If none of the above are true, then it is marked as `AIR`. Each coarse level in the Multigrid hierarchy also has a "ghost" padding of `SOLID` voxels. As illustrated in Figure 8, this procedure recursively constructs the Multigrid hierarchy. The above coarsening procedure is known to create geometric and topological discrepancies at coarser levels of the hierarchy, but the resulting inaccuracies are mitigated by an efficient smoother, as well as the use of the Multigrid solver as a *preconditioner* for Conjugate Gradients [McAdams et al. 2010]. In the variational method of Batty and Bridson [2008], velocity samples at grid faces are

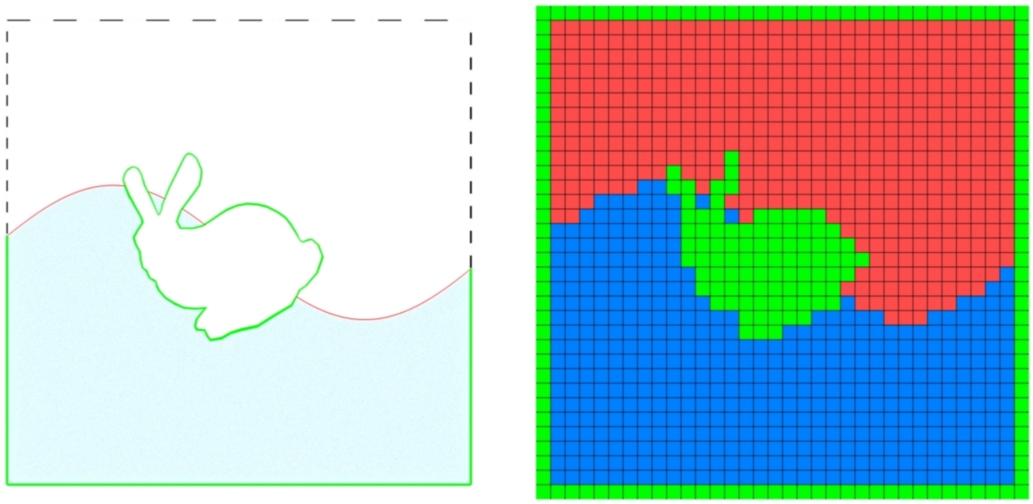


Fig. 7. (Left) A sample domain. (Right) Discretized domain with voxels marked as either `FLUID` (blue), `SOLID` (green), or `AIR` (red), depending on whether corresponding cell centers lie completely inside the liquid, solid, or air regions.

marked as active degrees of freedom if they themselves have a non-zero control volume weight, or if any of the neighboring stress samples have a non-zero control volume weight. Because we handle control volumes implicitly at coarse levels, we mark coarse grid faces as active degrees of freedom if they satisfy one of two conditions: (1) the face is incident to a `FLUID` voxel, implying a non-zero control volume weight at the face and neighboring the cell-centered stress, or (2) at least one of the edges on the face's boundary is itself incident to a `FLUID` voxel, implying a non-zero control volume weight at a neighboring edge stress sample.

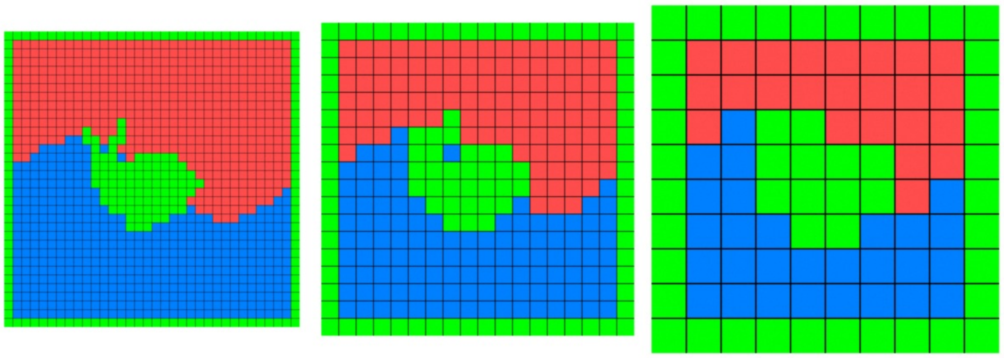


Fig. 8. Recursively computed Multigrid hierarchy, showing the various geometric and topological discrepancies.

6.2 Restriction/Prolongation Operators

The restriction operator $\mathcal{R}$ interpolates values stored at centers of grid faces from a given level to faces of the next (coarser) level in the Multigrid hierarchy. We construct this operator as the tensor

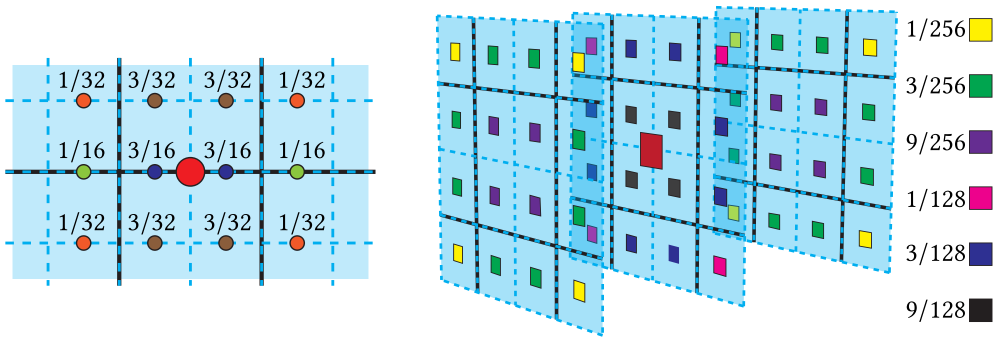


Fig. 9. Multigrid restriction stencil for a velocity degree of freedom (bright red) in (Left) 2D and (Right) 3D for down-sampling values from finer to coarser grids. The interpolation values have been color-coded in 3D according to the legend.

product of the 1D stencil $\mathcal{A}$ with 3 samples points *along* the axis of the face, and the 1D stencil $\mathcal{B}$ with 4 sample points for all other axes, where $\mathcal{A}$ and $\mathcal{B}$ are defined as:

$$(\mathcal{A}\boldsymbol{u}^h)(x) = \frac{1}{4}u^h(x-h) + \frac{1}{2}u^h(x) + \frac{1}{4}u^h(x+h) \quad (12)$$

$$\begin{aligned}(\mathcal{B}\boldsymbol{u}^h)(x) &= \frac{1}{8}u^h\left(x-\frac{3h}{2}\right) + \frac{3}{8}u^h\left(x-\frac{h}{2}\right) \\ &+ \frac{3}{8}u^h\left(x+\frac{h}{2}\right) + \frac{1}{8}u^h\left(x+\frac{3h}{2}\right) \quad (13)\end{aligned}$$

Here, $\boldsymbol{u}$ is the quantity being interpolated at a given location x, and h refers to the fine grid voxel size. Figure 9(left) shows the restriction stencil $\mathcal{R} = \mathcal{B} \otimes \mathcal{A}$ for the Y-faces in 2D, while Figure 9(right) shows the restriction stencil $\mathcal{R} = \mathcal{B} \otimes \mathcal{B} \otimes \mathcal{A}$ for the Z-faces in 3D. In both cases, the resulting coarse degree of freedom is highlighted in red and the fine degrees of freedom are color-coded according to their interpolation weights. Similar to McAdams et al. [2010], the prolongation operator $\mathcal{P}$ is defined as a scaled transpose of the restriction operator $\mathcal{R}$ to ensure the symmetry of a Multigrid V-cycle. The prolongation operator $\mathcal{P}$ corresponds exactly to trilinear interpolation from coarse to fine degrees of freedom. We limit the input/output of the restriction and prolongation operators to active degrees of freedom, substituting a zero value elsewhere.

6.3 Smoother Design

As we will show in Section 7, a damped Jacobi smoother, which is standard practice for Multigrid-based pressure projection [McAdams et al. 2010], performs poorly on the viscosity system (11) due to the strong coupling between different components of velocity. Brandt and Dinar [1978] proposed to address such problems with a so-called *box smoother* that locally solves for the subset of variables within a small subdomain while all exterior variables are held fixed.

Consider the local neighborhood for the X-velocity component shown in purple in Figure 10(left). Assuming this degree of freedom (DOF) is completely inside the liquid domain, its rectangular neighborhood comprises 17 staggered velocity variables in 2D (9 along the X axis, and 8 along the Y axis) and 75 in 3D (27 along the X axis, and 24 each along the Y and Z axes). A standard box smoother would require solving a 17×17 matrix in 2D and 75×75 matrix in 3D *per* velocity sample. While we have verified that this smoother *does* indeed achieve excellent convergence rates, it presents a significant computational overhead that limits its practical utility. Instead, we have

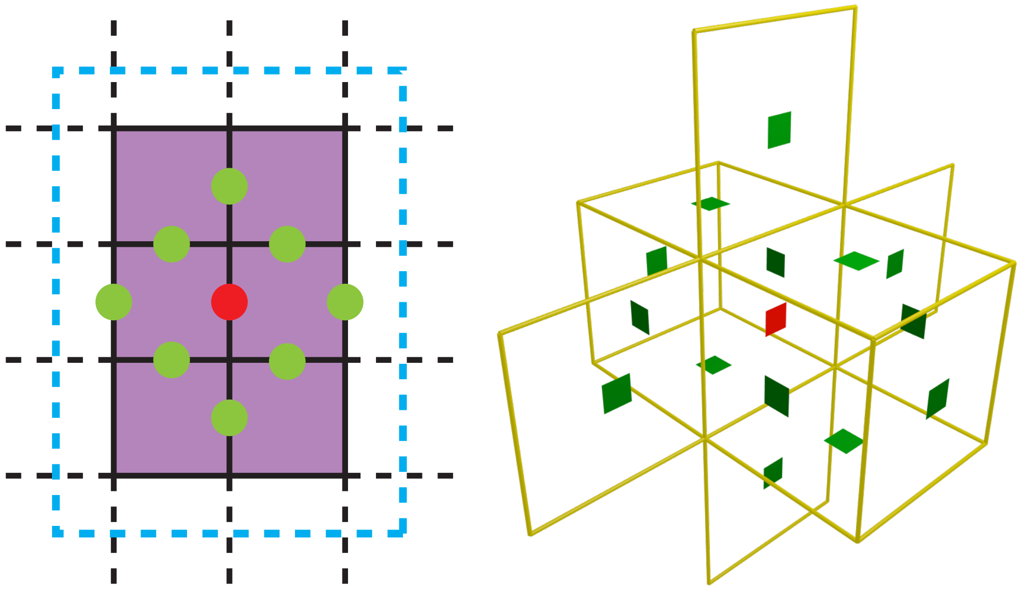


Fig. 10. Neighboring degrees of freedom (green) in the stencil for an X-face (red) for the variational viscosity system in (left) 2D and (right) 3D. The naïve box smoothing neighborhood for the 2D case is shown in purple.

empirically observed that excellent convergence rates for the Multigrid solver can still be preserved while *restricting* the small system for the box smoother to include *only* the neighbors inside the stencil of the given DOF. That is, we take the original local neighborhood subdomain matrix that the box smoother would have assembled for a given DOF and treat all the variables that are not involved in the immediate stencil of that single DOF as Dirichlet boundary conditions. Conveniently, this stencil involves just 9 neighbors in 2D and 15 neighbors in 3D, as shown in Figure 10, and therefore yields a much smaller system of equations (at most 9×9 in 2D and 15×15 in 3D). This smoother can be viewed as a tighter, modified box smoother in which the "box" is non-rectangular. Note that the subdomain system matrix can be even smaller if some of the variables inside the stencil of the DOF fall completely inside the solid or air regions. In our implementation, we use a direct Cholesky solver inside the smoother. For all strictly interior degrees of freedom away from free surface and solid boundaries, the small subdomain system matrices are identical. Thus, as an additional optimization, we precompute the Cholesky decomposition of this matrix, allowing us to perform only forward and backward substitutions for these DOFs during smoothing.

The above formulation has two key advantages: (1) the small size of the subdomain system matrix allows the box smoother to be applied at *all* degrees of freedom, which helps in decreasing the norm of the residual more rapidly, and (2) the pre-factorization of the subdomain system matrix for all strictly interior degrees of freedom allows our smoother to scale to large problem sizes. The use of a box smoother for Multigrid is not new itself, and has been previously explored by Zhu et al. [2010] in the context of elasticity simulation. However, they did not explore the "non-rectangular box" as we describe, nor consider any opportunities for pre-factorizing the subdomain system matrix. As such, they reported that the box smoother was impractical for their application. While line-based box smoothers have been previously explored in the context of two-phase flow simulation [Brandt and Dinar 1978], to the best of our knowledge, the non-rectangular box smoother is novel. We have empirically found it to work well for smoothing the implicit viscosity system of Batty and Bridson

[2008]. Its formulation, however, is not tied to the viscosity system in any way, and we conjecture that it should work well for the unsteady Stokes equations [Larionov et al. 2017] and volumetric elasticity [Zhu et al. 2010], among other problems with strongly coupled system of equations.

Solver	256^2 grid $57,341$ DOFs	512^2 grid $229,276$ DOFs	1024^2 grid $(916,912$ DOFs)	2048^2 grid $3,667,424$ DOFs
Multigrid (Naïve baseline)	11 iterations 1.9 seconds 3 levels	13 iterations 3.1 seconds 4 levels	14 iterations 4.7 seconds 5 levels	16 iterations 10.3 seconds 6 levels
Multigrid (Improved baseline)	12 iterations 0.74 seconds 3 levels	14 iterations 1.4 seconds 4 levels	16 iterations 3.1 seconds 5 levels	18 iterations 8.9 seconds 6 levels
Multigrid (Ours)	4 iterations 0.68 seconds 3 levels	5 iterations 1.8 seconds 4 levels	5 iterations 5.0 seconds 5 levels	4 iterations 12.5 seconds 6 levels
Eigen	729 iterations 0.91 seconds	1434 iterations 6.3 seconds	2816 iterations 51.9 seconds	5522 iterations 470.4 seconds
ICPCG	80 iterations 0.22 seconds	143 iterations 1.7 seconds	255 iterations 15.2 seconds	465 iterations 118.6 seconds
PARDISO	0.36 seconds	1.3 seconds	5.3 seconds	24.3 seconds

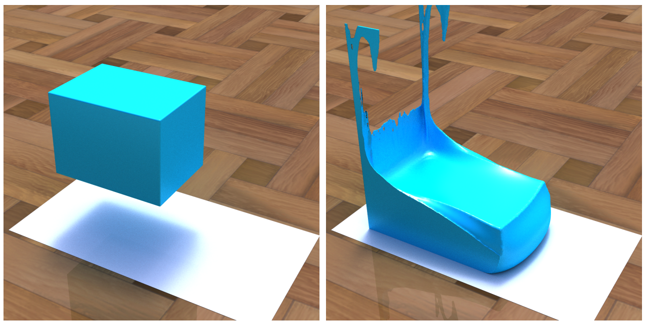
Table 1. Solver comparisons in 2D on an Intel Xeon E5-2630 v3 16-core CPU with 128 GB of RAM and 2.4 GHz processing speed.

6.4 Enhancements to Accommodate Variational Volume Weights

The variational approach of Batty and Bridson [2008] requires multiplying stencil coefficients with face-, edge-, and cell-centered weights to enforce boundary conditions, as described in Section 5. At coarser levels of the Multigrid hierarchy, however, these weights are not well-defined owing to the various geometric and topological discrepancies introduced during the coarsening procedure (see Section 6.1). We assume that all weights are either 1 or 0 at these coarser levels (and do not explicitly store them), essentially using a voxelized representation for the liquid. While this *does* create a mismatch between the fine and coarse level operators, this mismatch is localized *only* near the boundaries. Thus, as long as care is taken to ensure that both the residual computation and the smoothing routine at the finest level use the variational weights, this discrepancy does not affect the convergence of the Multigrid solver, because the fine level smoother effectively reduces the error near the boundaries. In practice, we have found this approach to be more efficient than the *deferred correction* suggested in prior work [Aanjaneya et al. 2017; Liu et al. 2016], where *all* levels in the Multigrid hierarchy would use weights that are either 1 or 0, and *extra* Jacobi smoothing iterations would be required (with the actual volume weights at the finest level) before and after the Multigrid V-cycle. By *directly* using the correct volume weights at the finest level, we avoid this additional computational overhead.

6.5 Moving Solids

Kinematic and two-way coupled solids impose non-zero velocity Dirichlet boundary conditions on the variational viscosity system in equation (11), which are not well-defined at coarser levels of the Multigrid hierarchy. We handle this issue by adjusting the right hand side vector with these boundary terms before the viscosity solve, essentially translating all non-zero Dirichlet boundary conditions to zero Dirichlet boundary conditions. This modification is also well-aligned with the design choice that the input and output of our restriction and prolongation operators are limited to active degrees of freedom only, substituting a value of zero elsewhere.



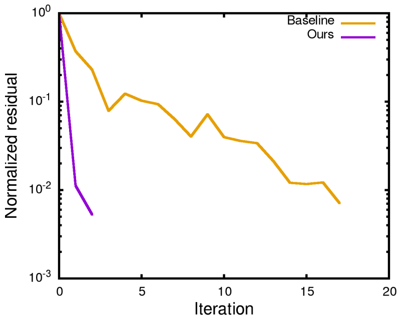


Fig. 11. (Left) A block of liquid with spatially varying viscosity is dropped ($256^2 \times 512$ grid), creating a splash at the low viscosity end. The deformation on the opposite end with a higher viscosity coefficient is much less pronounced. (Right) Convergence profiles for the Multigrid solver using Jacobi smoothing, and our proposed box smoothing.

6.6 Variable Viscosity

Our solver also supports variable viscosity, as shown in Figure 11(top), where a block of liquid with spatially varying viscosity is dropped. A splash occurs at the end with lower viscosity, while the deformation on the opposite end with a higher viscosity coefficient is much less pronounced. Since the viscosity coefficients vary in space, our smoother cannot employ the pre-factorization of the subdomain system matrix for all strictly interior degrees of freedom, as the subdomain system matrices change from point to point. Thus, our box smoother is not as efficient as it would have been for the spatially constant viscosity case. However, compared to Jacobi-preconditioned Multigrid (see baseline solver in Section 7) with an average solve time of 211.2 s for viscosity per time step, our proposed solver has an average solve time of 140.9 s for reducing the residual by two orders of magnitude, obtaining a 1.5× speedup. Figure 11(bottom) shows the convergence profiles of the two solvers.

7 RESULTS

We use the direct Cholesky solver from the Eigen library [Guennebaud et al. 2010] in the implementation of our box smoother; for boundary DOFs we perform a standard solve of the corresponding small subdomain system while for interior DOFs we can quickly perform forward/backward substitution using a precomputed factorization. We use the sparse Conjugate Gradients solver from the Eigen library [Guennebaud et al. 2010] as an approximation for the "exact" solver at the bottom of the Multigrid V-cycle, which we found to be more efficient than either a direct solve or additional box smoother iterations. We set the floating-point precision of 10^{-7} as the error tolerance for this "exact" solver, to preserve the symmetry of the V-cycle.

For benchmarking purposes, we consider a test problem, where a solid sphere rests in the center of a square domain going from $[0, 0]$ to $[1, 1]$ in 2D (resp. $[0, 0, 0]$ to $[1, 1, 1]$ in 3D), and the liquid-air interface is dictated by the analytic function $\phi = x - 0.5 + 0.25 \cdot \sin(x \cdot y)$ in 2D (resp. $\phi = x - 0.5 + 0.25 \cdot \sin(x \cdot y \cdot z)$ in 3D). The right hand side is initialized as a delta function at

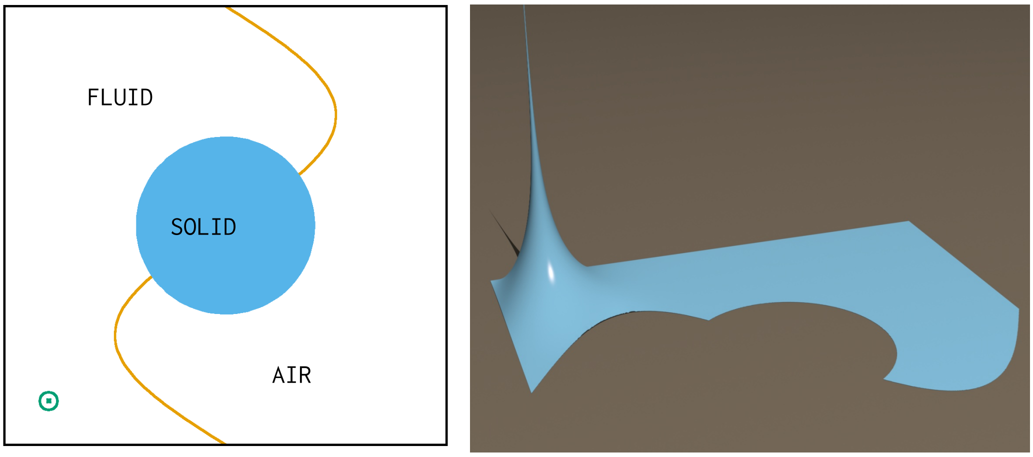


Fig. 12. A test problem with mixed boundary conditions. The right hand side is a delta function centered at the point shown in green. The exact solution is shown in the right.

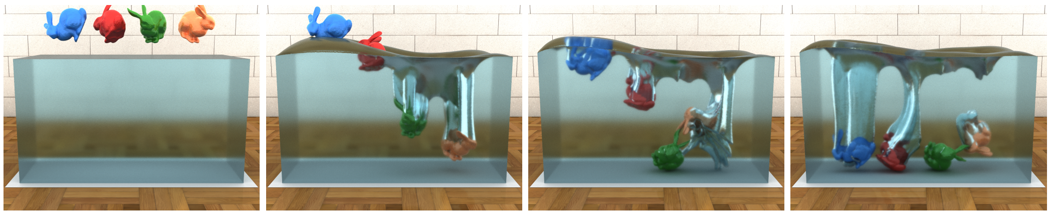


Fig. 13. Four bunnies, increasing in density from left to right, are dropped in a pool of viscous liquid. The heavier bunnies sink immediately, causing the liquid to rise on the left and cushion the fall of the lighter bunnies. Then, due to insufficient buoyancy force to support their weight, the lighter bunnies subsequently also sink (simulated on a $384^2 \times 192$ grid).

the faces nearest to the point $(0.1, 0.1)$ in 2D (resp. $(0.1, 0.1, 0.1)$ in 3D). The exact solution for the liquid velocity (u, v) in 2D is visualized as a height field, and resembles a "tent" function, as shown in Figure 12(right).

To provide a baseline Multigrid for comparison, we followed typical practice in the design of Multigrid solvers for pressure projection and implemented the damped-Jacobi smoother of McAdams et al. [2010]. At each level in the Multigrid hierarchy, we perform 3 Jacobi iterations in a 3 voxel-wide narrow band along the boundary, followed by 1 Jacobi iteration in the entire domain, followed by 3 boundary Jacobi iterations again to maintain symmetry. At the bottom of the Multigrid hierarchy, we simply use 200 Jacobi iterations in the entire domain as an approximation for the "exact" solver (again following McAdams et al. [2010]).

We stress that while we use this Multigrid solver as a baseline point of comparison to hold our proposed solver to a high standard, even this "baseline" approach has not been previously applied to the viscous liquid animation problem we consider [Batty and Bridson 2008]. Standard practice is to use solvers such as Eigen [Guennebaud et al. 2010], Intel's MKL PARDISO solver [De Coninck et al. 2016], or ICPCG, which are far less efficient (or wasteful in terms of memory usage) compared to our baseline Multigrid (see Tables 1 and 2). For example, Houdini [SideFX 2019] uses Jacobi-preconditioned Conjugate Gradients. Figure 14 shows the convergence profiles of the baseline Multigrid solver (labeled as J, and hereon referred to as the "naïve" baseline Multigrid solver),

Solver	128^3 grid 3, 035, 346 DOFs	256^3 grid 24, 303, 388 DOFs	512^3 grid 194, 503, 048 DOFs
Multigrid (Naïve baseline)	20 iterations 377.3 s 0.256 GB 2 levels	34 iterations 975.3 s 2.18 GB 3 levels	39 iterations 2920.4 s 15.94 GB 4 levels
Multigrid (Improved baseline)	21 iterations 90.1 s 0.256 GB 2 levels	36 iterations 333.8 s 2.18 GB 3 levels	41 iterations 1512.4 s 15.94 GB 4 levels
Multigrid (Ours)	3 iterations 27.3 s 0.256 GB 2 levels	3 iterations 119.6 s 2.18 GB 3 levels	4 iterations 834.7 s 15.94 GB 4 levels
Eigen	377 iterations 35.6 s 1.536 GB	707 iterations 619.4 s 11.78 GB	out of memory
ICPCG	39 iterations 6.7 s 1.024 GB	53 iterations 95.3 s 8.32 GB	out of memory
PARDISO	1626.9 s 24.96 GB	N/A	N/A

Table 2. Solver performance comparisons in 3D for the test problem in Figure 12 on an Intel Xeon E5-2630 v3 CPU with 16 cores, 128 GB of RAM, and 2.4 GHz processing speed.

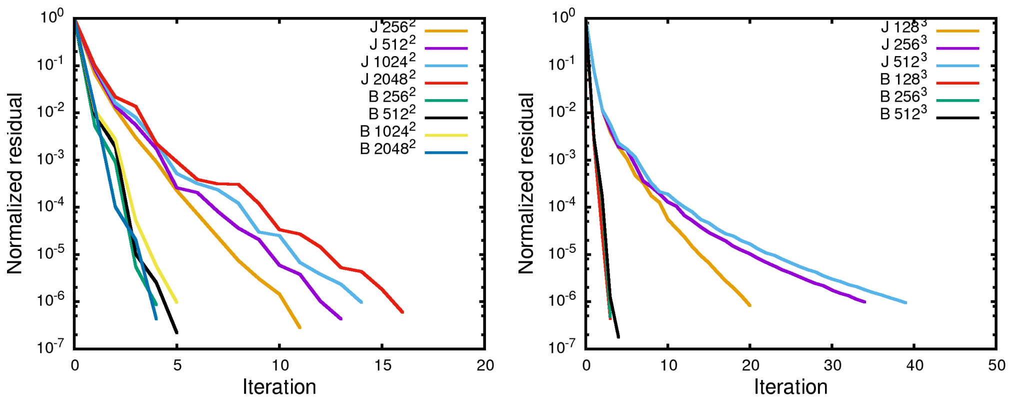


Fig. 14. Convergence profiles for the Multigrid solver using Jacobi smoothing, and our proposed box smoothing in 2D (left) and 3D (right) for the test problem in Figure 12.

compared to our proposed Multigrid solver using a box smoother (labeled as B). As can be seen, our solver requires far fewer iterations.

Tables 1 and 2 provide computational performance and memory usage numbers for reducing the residual (computed in the L_∞ norm) by 6 orders of magnitude, comparing: our Multigrid solver, the baseline Multigrid solver, the sparse Conjugate Gradients solver from the Eigen library [Guennebaud et al. 2010], Intel's MKL PARDISO solver [De Coninck et al. 2016], and our own implementation of Modified Incomplete Cholesky-preconditioned Conjugate Gradients (ICPCG). To better isolate the benefits of using our box smoother, we also provide performance comparisons with an "improved" baseline Multigrid solver that uses the sparse CG solver from the Eigen [Guennebaud et al. 2010] library as an approximation for the coarsest-level "exact" solver (consistent with our recommended approach), rather than a large number of smoothing steps as suggested by e.g., McAdams et al. [2010]. In 2D, we find that this improved baseline Multigrid is the fastest solver for large enough grid resolutions. This is because the computational overhead of our box smoother is not insignificant, and the coupling between different components of the velocity in 2D is not strong enough to hurt the effectiveness of Jacobi smoothing. However, the story changes dramatically in 3D. We find that the PARDISO solver is far too slow with a very large memory footprint, most likely due to the presence of many more non-zero terms compared to Poisson problems, and so we did not run it beyond the 128^3 grid. Both the Eigen solver and ICPCG run out of memory for our largest example with grid resolution 512^3. Finally, our proposed solver is faster by a factor between $3.5\times$ - $13.8\times$ compared to the naïve baseline Multigrid solver, and by a factor between $1.8\times$ - $3.3\times$ compared to the improved baseline Multigrid solver. All Multigrid solvers are also memory-efficient, requiring an overall footprint of less than 16 GB for our largest example with grid resolution 512^3. Although the observed *relative* gains offered by the current implementation of our proposed solver vs. the baseline Multigrid solvers decrease noticeably with higher grid resolutions, they remain significant. Moreover, we believe that the use of SIMD vectorization in our box smoother, similar to the use of *macroblocks* for non-linear elasticity as proposed by Mitchell et al. [2016], could further boost the computational performance of our solver.

Implementation notes: Compared to a Multigrid solver for pressure projection [McAdams et al. 2010] on a given domain, a Multigrid solver for viscosity inherently incurs a higher memory overhead because the variational viscosity system in equation (11) is three times larger than the scalar Poisson equation, with more non-zeros to account for the cross-derivative terms between different components of the velocity. At each level of the Multigrid hierarchy, we store the solution vector $\vec{x}$, the right hand side $\vec{b}$, the residual vector $\vec{r}$, and a bitmask indicating all active faces (computed using the cell labels `FLUID`, `SOLID`, and `AIR`, as outlined in Section 6). In addition, we store the cell-centered, edge-centered, and face-centered volume weights at the finest level of the hierarchy to apply the variational finite difference operators, as explained in Section 5. Even so, our solver can efficiently solve for more than 194 million degrees of freedom, while occupying a memory footprint of less than 16 GB, as shown in Table 2. We use the post-advected divergence-free velocities as initial guess for Conjugate Gradients when solving for the effects of viscosity, obtaining faster convergence. We partition the simulation domain along the X-axis for multi-threading, and use red-black coloring to avoid race conditions in our box smoother, iterating only on partitions of one color in each sweep.

7.1 Simulation Examples

To highlight the benefits of our solver, we simulated high resolution examples of viscous liquids. Figure 3 shows a moving rectangular source pouring chocolate that displays gentle buckling patterns as it falls on the wafer. Figure 4 shows a cake topped with a creamy armadillo topping that collapses under its own weight, exhibiting detailed folding behavior. (The slightly "voxelized" appearance of the armadillo surface is due to a coarse sampling of the initial signed distance field from an explicit mesh representation.) Figure 13 shows an example with two-way coupled rigid bodies in which four bunnies, of increasing weight from left to right, are dropped in a pool of viscous liquid. The heavier bunnies sink in first, causing the liquid surface to rise on the left and cushion the fall of the lighter bunnies. The heaviest bunny collides with the ground and rebounds, creating interesting flow features. After a while, the two lighter bunnies also sink into the liquid, since the liquid cannot provide sufficient buoyancy force to support their weight. Table 3 gives the average solve times per time step for all our examples for reducing the residual by 2 orders of magnitude.

	Figure 2	Figure 3	Figure 4	Figure 13
Viscosity Solve	25.5 s	30.2 s	53.1 s	47.8 s
MG levels	5	5	5	5
PCG iterations	2	2	2	2

Table 3. Average timings for all examples, computed on an Intel Xeon E5-2630 v3 (16-core, 2.4GHz) CPU.

8 CONCLUSION AND FUTURE WORK

We presented a geometric Multigrid-preconditioned Conjugate Gradients solver for efficiently solving the viscosity system of Batty and Bridson [2008]; it exploits assembly-free restriction and prolongation operators along with an effective specially-tailored box smoother that only requires the solution of a small sparse matrix (at most 9×9 in 2D and 15×15 in 3D). Our solver is fully parallelizable, memory-efficient, and outperforms existing solutions such as Eigen [Guennebaud et al. 2010], Intel's MKL [De Coninck et al. 2016], and Modified Incomplete Cholesky-preconditioned Conjugate Gradients (ICPCG). To the best of our knowledge, this is also the first time Multigrid techniques have been applied to the viscosity equations in computer graphics applications.

Perhaps the biggest limitation of our existing implementation is the computational overhead incurred by the box smoother. It would be interesting to explore the use of SIMD vectorization techniques, similar to the macroblock-based approach for non-linear elasticity, as proposed by Mitchell et al. [2016]. Alternatively, the use of *distributive smoothing* techniques [Brandt and Dinar 1978; Gaspar et al. 2008; Zhu et al. 2010] could also be beneficial; these use an appropriate change of variables to convert the system into one that only has Laplacians on the diagonal, since damped-Jacobi smoothers are known to work well in such situations. The latter approach would still require some special treatment for handling the free surface and solid wall boundary conditions, by either tweaking the discretization itself [Zhu et al. 2010], or using our box smoother.

Goldade et al. [2019] developed an octree-based discretization for the same viscosity PDE, and we would like to explore combining it with our solver. Finally, it would be interesting to extend our approach to design a geometric Multigrid solver for the full unsteady Stokes scheme of Larionov et al. [2017]. Their discretization entails solving an even larger coupled system for velocity and pressure to simultaneously capture the effects of viscosity and incompressibility, achieving markedly improved behavior at the liquid surface (e.g., classic rope coiling). Its increased computational cost currently renders it much less attractive for practical applications, but an efficient Multigrid scheme could at least partially alleviate this concern.

9 ACKNOWLEDGEMENTS

This work was supported in part by the Rutgers University start-up grant, the Ralph E. Powe Junior Faculty Enhancement Award, and the Natural Sciences and Engineering Research Council of Canada (RGPIN-04360-2014, CRDPJ-499952-2016). We would like to thank Cristin Barghiel from SideFX Software for their generous software donation. We would also like to thank the RenderMan team at Pixar for their education licenses, which was used for rendering all our data, and the anonymous reviewers for their valuable feedback.

REFERENCES

Mridul Aanjaneya. 2018. An Efficient Solver for Two-way Coupling Rigid Bodies with Incompressible Flow. *Computer Graphics Forum* 37, 8 (2018), 59–68.

Mridul Aanjaneya, Ming Gao, Haixiang Liu, Christopher Batty, and Eftychios Sifakis. 2017. Power Diagrams and Sparse Paged Grids for High Resolution Adaptive Liquids. *ACM Trans. Graph.* 36, 4, Article 140 (2017), 12 pages.

D Adalsteinsson and J.A Sethian. 1999. The Fast Construction of Extension Velocities in Level Set Methods. *J. Comput. Phys.* 148, 1 (1999), 2 – 22.

Ryoichi Ando, Nils Thürey, and Chris Wojtan. 2015. A Dimension-reduced Pressure Solver for Liquid Simulations. *Computer Graphics Forum* 34, 2 (2015), 473–480.

Héctor Barreiro, Ignacio García-Fernández, Iván Alduán, and Miguel A Otaduy. 2017. Conformation constraints for efficient viscoelastic fluid simulation. *ACM Transactions on Graphics (TOG)* 36, 6 (2017), 221.

Christopher Batty and Robert Bridson. 2008. Accurate Viscous Free Surfaces for Buckling, Coiling, and Rotating Liquids. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '08)*. 219–228.

Christopher Batty and Ben Houston. 2011. A simple finite volume method for adaptive viscous liquids. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. ACM, 111–118.

Jan Bender and Dan Koschier. 2017. Divergence-free SPH for incompressible and viscous fluids. *IEEE Transactions on Visualization and Computer Graphics* 23, 3 (2017), 1193–1206.

Miklós Bergou, Basile Audoly, Etienne Vouga, Max Wardetzky, and Eitan Grinspun. 2010. Discrete viscous threads. In *ACM Transactions on Graphics (TOG)*, Vol. 29. ACM, 116.

Jeff Bolz, Ian Farmer, Eitan Grinspun, and Peter Schröoder. 2003. Sparse matrix solvers on the GPU: conjugate gradients and multigrid. In *ACM transactions on graphics (TOG)*, Vol. 22. ACM, 917–924.

Achi Brandt and Nathan Dinar. 1978. Multigrid Solutions to Elliptic Flow Problems. *Numerical methods for partial differential equations* (1978), 53–147.

Robert Bridson. 2015. *Fluid simulation for computer graphics, 2nd edition.* CRC Press.

Mark Carlson, Peter J. Mucha, R. Brooks Van Horn, III, and Greg Turk. 2002. Melting and Flowing. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '02)*. 167–174.

Nuttapong Chentanez, Bryan E Feldman, François Labelle, James F O'Brien, and Jonathan R Shewchuk. 2007. Liquid simulation on lattice-based tetrahedral meshes. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*. Eurographics Association, 219–228.

Nuttapong Chentanez and Matthias Mueller-Fischer. 2012. A multigrid fluid pressure solver handling separating solid boundary conditions. *IEEE transactions on visualization and computer graphics* 18, 8 (2012), 1191–1201.

Nuttapong Chentanez and Matthias Müller. 2011. Real-time Eulerian water simulation using a restricted tall cell grid. In *ACM Transactions on Graphics (TOG)*, Vol. 30. ACM, 82.

Jieyu Chu, Nafees Bin Zafar, and Xubo Yang. 2017. A schur complement preconditioner for scalable parallel fluid simulation. *ACM Transactions on Graphics (TOG)* 36, 5 (2017), 163.

Pascal Clausen, Martin Wicke, Jonathan R Shewchuk, and James F O'brien. 2013. Simulating liquids and solid-liquid interactions with lagrangian meshes. *ACM Transactions on Graphics (TOG)* 32, 2 (2013), 17.

Arne De Coninck, Bernard De Baets, Drosos Kourounis, Fabio Verbosio, Olaf Schenk, Steven Maenhout, and Jan Fostier. 2016. Needles: Toward Large-Scale Genomic Prediction with Marker-by-Environment Interaction. 203, 1 (2016), 543–555.

C. Dick, M. Rogowsky, and R. Westermann. 2016. Solving the Fluid Pressure Poisson Equation Using MultigridâĂŤEvaluation and Improvements. *IEEE Transactions on Visualization and Computer Graphics* 22, 11 (Nov 2016), 2480–2492. https://doi.org/10.1109/TVCG.2015.2511734

Essex Edwards and Robert Bridson. 2014. Detailed water with coarse grids: combining surface meshes and adaptive discontinuous Galerkin. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 136.

Douglas Enright, Frank Losasso, and Ronald Fedkiw. 2005. A Fast and Accurate semi-Lagrangian Particle Level Set Method. *Comput. Struct.* 83, 6-7 (2005), 479–490.

D. Enright, D. Nguyen, F. Gibou, and R. Fedkiw. 2003. Using the Particle Level Set Method and a Second Order Accurate Pressure Boundary Condition for Free Surface Flows. In *Proc. 4th ASME-JSME Joint Fluids Eng. Conf.*

Kenny Erleben, Marek Krzysztof Misztal, and J Andreas Bærentzen. 2011. Mathematical foundation of the optimization-based fluid animation method. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. ACM, 101–110.

Henrik Fält and Douglas Roble. 2003. Fluids with Extreme Viscosity. In *ACM SIGGRAPH Sketches*.

Florian Ferstl, Rüdiger Westermann, and Christian Dick. 2014. Large-scale liquid simulation on adaptive hexahedral grids. *IEEE transactions on visualization and computer graphics* 20, 10 (2014), 1405–1417.

Nick Foster and Dimitri Metaxas. 1996. Realistic animation of liquids. *Graphical models and image processing* 58, 5 (1996), 471–483.

F. J. Gaspar, J. L. Gracia, F. J. Lisbona, and C. W. Oosterlee. 2008. Distributive Smoothers in Multigrid for Problems with Dominating grad-div Operators. *Numerical Linear Algebra with Applications* 15, 8 (2008), 661–683.

Tolga G Goktekin, Adam W Bargteil, and James F O'Brien. 2004. A method for animating viscoelastic fluids. In *ACM Transactions on Graphics (TOG)*, Vol. 23. ACM, 463–468.

Ryan Goldade, Yipeng Wang, Mridul Aanjaneya, and Christopher Batty. 2019. An Adaptive Variational Finite Difference Framework for Efficient Symmetric Octree Viscosity. *ACM Trans. Graph. (SIGGRAPH)* 38, 4 (2019).

Eran Guendelman, Robert Bridson, and Ronald Fedkiw. 2003. Nonconvex Rigid Bodies with Stacking. *ACM TOG* 22, 3 (2003), 871–878.

Eran Guendelman, Andrew Selle, Frank Losasso, and Ronald Fedkiw. 2005. Coupling Water and Smoke to Thin Deformable and Rigid Shells. *ACM Trans. Graph.* 24, 3 (2005), 973–981.

Gaël Guennebaud, Benoît Jacob, et al. 2010. Eigen v3. http://eigen.tuxfamily.org.

Francis H. Harlow and J. Eddie Welch. 1965. Numerical Calculation of Time-Dependent Viscous Incompressible Flow of Fluid with Free Surface. *Physics of Fluids (1958-1988)* 8, 12 (1965), 2182–2189.

Misha Kazhdan and Hugues Hoppe. 2018. An Adaptive Multi-grid Solver for Applications in Computer Graphics. *Computer Graphics Forum* 38 (2018), 138–150. Issue 1.

Egor Larionov, Christopher Batty, and Robert Bridson. 2017. Variational stokes: a unified pressure-viscosity solver for accurate viscous liquids. *ACM Trans. Graph. (SIGGRAPH)* 36, 4, Article 101 (2017), 11 pages.

Haixiang Liu, Nathan Mitchell, Mridul Aanjaneya, and Eftychios Sifakis. 2016. A scalable Schur-complement fluids solver for heterogeneous compute platforms. *ACM Trans. Graph.* 35, 6, Article 201 (2016), 12 pages.

Frank Losasso, Ronald Fedkiw, and Stanley Osher. 2005. Spatially adaptive techniques for level set methods and incompressible flow. *Computers and Fluids* 35 (2005), 2006.

Frank Losasso, Frédéric Gibou, and Ron Fedkiw. 2004. Simulating water and smoke with an octree data structure. In *ACM transactions on graphics (TOG)*, Vol. 23. ACM, 457–462.

Frank Losasso, Tamar Shinar, Andrew Selle, and Ronald Fedkiw. 2006. Multiple interacting liquids. In *ACM Transactions on Graphics (TOG)*, Vol. 25. ACM, 812–819.

A. McAdams, E. Sifakis, and J. Teran. 2010. A Parallel Multigrid Poisson Solver for Fluids Simulation on Large Grids. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '10).* 65–74.

Aleka McAdams, Yongning Zhu, Andrew Selle, Mark Empey, Rasmus Tamstorf, Joseph Teran, and Eftychios Sifakis. 2011. Efficient Elasticity for Character Skinning with Contact and Collisions. *ACM Trans. Graph.* 30, 4, Article 37 (2011), 12 pages.

Nathan Mitchell, Michael Doescher, and Eftychios Sifakis. 2016. A Macroblock Optimization for Grid-based Nonlinear Elasticity. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '16).* 11–19.

Jeroen Molemaker, Jonathan M. Cohen, Sanjit Patel, and Jonyong Noh. 2008. Low Viscosity Flow Simulations for Animation. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '08).* 9–18.

Andreas Peer, Markus Ihmsen, Jens Cornelis, and Matthias Teschner. 2015. An implicit viscosity formulation for SPH fluids. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 114.

Daniel Ram, Theodore Gast, Chenfanfu Jiang, Craig Schroeder, Alexey Stomakhin, Joseph Teran, and Pirouz Kavehpour. 2015. A material point method for viscoelastic fluids, foams and sponges. In *Proceedings of the 14th ACM SIGGRAPH/Eurographics Symposium on Computer Animation.* ACM, 157–163.

Nick Rasmussen, Doug Enright, Duc Nguyen, Sebastian Marino, N. Sumner, Willi Geiger, Samir Hoon, and Ron Fedkiw. 2004. Directable photorealistic liquids. In *Symposium on Computer Animation.* 193–202.

Avi Robinson-Mosher, Craig Schroeder, and Ronald Fedkiw. 2011. A Symmetric Positive Definite Formulation for Monolithic Fluid Structure Interaction. *J. Comput. Phys.* 230, 4 (2011), 1547–1566.

Allen Ruilova. 2007. Creating Realistic CG Honey. In *ACM SIGGRAPH Posters.*

Rajsekhar Setaluri, Mridul Aanjaneya, Sean Bauer, and Eftychios Sifakis. 2014. SPGrid: A Sparse Paged Grid Structure Applied to Adaptive Smoke Simulation. *ACM Trans. Graph.* 33, 6, Article 205 (2014), 12 pages.

SideFX. 2019. Houdini.

Jos Stam. 1999. Stable Fluids.. In *Siggraph*, Vol. 99. 121–128.

Tetsuya Takahashi, Yoshinori Dobashi, Issei Fujishiro, Tomoyuki Nishita, and Ming C Lin. 2015. Implicit formulation for SPH-based viscous fluids. In *Computer Graphics Forum*, Vol. 34. Wiley Online Library, 493–502.

Rasmus Tamstorf, Toby Jones, and Stephen F McCormick. 2015. Smoothed aggregation multigrid for cloth simulation. *ACM Transactions on Graphics (TOG)* 34, 6 (2015), 245.

Ulrich Trottenberg, Cornelius W Oosterlee, and Anton Schuller. 2000. *Multigrid.* Elsevier.

Alberto Valli, ALFIO Quarteroni, et al. 1999. Domain decomposition methods for partial differential equations. *Numerical Mathematics and Scientific Computation, The Clarendon Press, Oxford University Press, New York* (1999).

Daniel Weber, Johannes Mueller-Roemer, André Stork, and Dieter Fellner. 2015. A Cut-Cell Geometric Multigrid Poisson Solver for Fluid Simulation. In *Computer Graphics Forum*, Vol. 34. Wiley Online Library, 481–491.

Marcel Weiler, Dan Koschier, Magnus Brand, and Jan Bender. 2018. A physically consistent implicit viscosity solver for SPH fluids. In *Computer Graphics Forum*, Vol. 37. Wiley Online Library, 145–155.

Mark Wiebe and Ben Houston. 2004. The Tar Monster: Creating a Character with Fluid Simulation. In *ACM SIGGRAPH 2004 Sketches (SIGGRAPH '04)*.

Kui Wu, Nghia Truong, Cem Yuksel, and Rama Hoetzlein. 2018. Fast fluid simulations with sparse volumes on the GPU. In *Computer Graphics Forum*, Vol. 37. Wiley Online Library, 157–167.

Xinxin Zhang and Robert Bridson. 2014. A PPPM Fast Summation Method for Fluids and Beyond. *ACM Trans. Graph.* 33, 6, Article 206 (2014), 11 pages.

Yongning Zhu, Eftychios Sifakis, Joseph Teran, and Achi Brandt. 2010. An efficient multigrid method for the simulation of high-resolution elastic solids. *ACM Trans. Graph. (presented at SIGGRAPH 2010)* 29, 2 (2010), 16:1–16:18.