Simulating Liquids and Solid-Liquid Interactions with Lagrangian Meshes

Pascal Clausen, Martin Wicke, Jonathan R. Shewchuk, and James F. O'Brien
University of California, Berkeley

This paper describes a Lagrangian finite element method that simulates the behavior of liquids and solids in a unified framework. Local mesh improvement operations maintain a high-quality tetrahedral discretization even as the mesh is advected by fluid flow. We conserve volume and momentum, locally and globally, by assigning each element an independent rest volume and adjusting it to correct for deviations during remeshing and collisions. Incompressibility is enforced with per-node pressure values, and extra degrees of freedom are selectively inserted to prevent pressure locking. Topological changes in the domain are explicitly treated with local mesh splitting and merging. Our method models surface tension with an implicit formulation based on surface energies computed on the boundary of the volume mesh.

With this method we can model elastic, plastic, and liquid materials in a single mesh, with no need for explicit coupling. We also model heat diffusion and thermoelastic effects, which allow us to simulate phase changes. We demonstrate these capabilities in several fluid simulations at scales from millimeters to meters, including simulations of melting caused by external or thermoelastic heating.

Categories and Subject Descriptors: G.1.8 [**Numerical Analysis**]: Partial Differential Equations—*Finite element methods*; I.3.5 [**Computer Graphics**]: Computational Geometry—*Physically based modeling*; I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism—*Animation*; I.6.8 [**Simulation and Modeling**]: Types of Simulation—*Animation*

General Terms: Algorithms, Simulation

Additional Key Words and Phrases: Finite element method, physically-based computer animation, fluid dynamics, surface tension, solid-fluid interface, melting, thermoelasticity, dynamic mesh generation, local remeshing

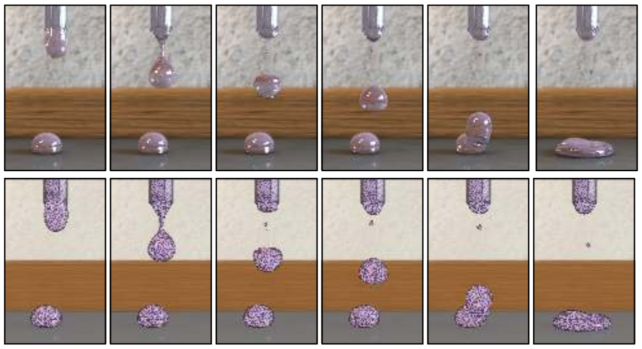

Fig. 1: A simulation of liquid dripping onto a hydrophobic surface. The top row shows rendered images; the bottom row visualizes the dynamic tetrahedral simulation mesh.

ACM Reference Format:

Clausen, P., Wicke, M., Shewchuk, J. R., and O'Brien, J. F. 2013. Simulating Liquids and Solid-Liquid Interactions with Lagrangian Meshes. ACM Trans. Graph. 32, 2, Article 17 (April 2013), 15 pages.
DOI = http://dx.doi.org/10.1145/2451236.2451243

1. INTRODUCTION

Most methods for simulating physical phenomena can be classified as either *Lagrangian*, where the discretization moves with the simulated material, or *Eulerian*, where the material moves through a stationary discretization. With few exceptions, Lagrangian methods are used to simulate elastic and plastic solids, while Eulerian methods are used to simulate fluids.

This paper presents a fully Lagrangian fluid simulator that employs a dynamically changing tetrahedral mesh to discretize a simulated material. Our method handles materials ranging from inviscid or viscous fluids to plastic or stiff elastic solids, smooth interfaces between them, and phase transitions such as melting or freezing.

Mesh-based Lagrangian methods have limitations that have so far prevented their widespread adoption for fluids. If a material undergoes large deformations or flow, the simulation mesh has to be restructured to accommodate the new configuration. Until recently, local three-dimensional remeshing was too difficult, and global remeshing introduced too much resampling error (often in the form of numerical viscosity) to support fluids. To simulate inviscid, turbulent, free-surface flows in a Lagrangian framework, we have designed algorithms to locally repair, and to robustly split and merge, tetrahedral meshes.

Because our dynamic local remesher maintains high element quality while changing as few elements as possible, our Lagrangian meshes achieve extremely low numerical viscosity even in simulations with moderately coarse resolutions and large time steps. We can also model surface tension accurately and reproduce emergent effects such as the period of an oscillating water drop or the con-

This work was supported in part by NSF Awards CCF-0635381 and IIS-0915462, UC Lab Fees Research Program Grant 09-LR-01-118889-OBRJ, Intel's Science and Technology Center for Visual Computing, and by gifts from Autodesk, NVIDIA, and Pixar.
Correspondence author: J. F. O'Brien, University of California, Berkeley, CA; email: job@berkeley.edu.

tact angles between liquids and hydrophilic or hydrophobic surfaces without explicitly enforcing such phenomena.

A distinguishing virtue of our simulation method is its ability to locally preserve a fluid's volume and momentum over long periods, irrespective of the shape of the fluid domain. In particular, thin sheets and streams can be resolved with great precision, and without the intrinsic volume loss that plagues Eulerian methods.

Because the tetrahedral mesh provides an explicit boundary surface, there is no need for a separate algorithm for surface tracking or extraction (such as level-set methods, particles, or semi-Lagrangian advection). The explicit boundary allows surface effects such as surface tension to be modeled with a physical accuracy that would otherwise be difficult to achieve.

2. OVERVIEW OF OUR METHOD

Our discretization uses piecewise linear basis functions defined on meshes of tetrahedral elements. Elasticity is modeled in a Lagrangian fashion with Cauchy's linearized strain. Large deformations are correctly treated with corotation [Müller and Gross 2004]. Plasticity is implemented as described by Wicke *et al.* [2010]. We use a $\mathcal{P}1$–$\mathcal{P}1$ formulation, where both velocities and pressures are stored at nodes and interpolated with linear basis functions over the elements. Heat diffusion is discretized on the same mesh and temperatures are also stored at the nodes.

We model liquids as perfectly plastic, incompressible materials in which the elastic shear stresses are always zero so that the material flows freely with a specified viscosity. We enforce incompressibility by discretizing the pressure field and solving for pressures and velocities simultaneously, enforcing constraints so that the velocity field has zero divergence. The zero-divergence constraint on velocity is discretized by enforcing it for the one-ring of each mesh node [Irving et al. 2007]. Unfortunately, these constraints can cause *pressure locking* at boundaries, because the velocities can be locally overconstrained. We remedy the problem by locally inserting nodes if a critical mesh configuration is detected; see Sec. 4.2.

Flow distorts the mesh causing the element quality to degrade until remeshing is required. Remeshing in turn requires resampling field variables, which introduces interpolation errors. We therefore prefer to remesh as little as possible, changing the mesh only locally to maintain a minimum quality, and remeshing with the least invasive operations available. We use the *Pulsar* mesh repair algorithm [Klingner and Shewchuk 2007; Wicke et al. 2010] and enhance it with new remeshing operations and a more sophisticated treatment of the mesh surface; see Sec. 6.

We also simulate splitting events caused by excessive stresses and thinning, and merging events when fluids or sticky materials collide. We implement splitting with operations that allow cracks to propagate through element interiors, as opposed to the more common methods in which cracks always follow existing element faces. We implement merging by subdividing overlapping elements so that they conform to each other, then deleting duplicate elements.

The following list summarizes the actions taken during each time step. They are described in detail in later sections.

1. **Plasticity –** Account for plastic flow by updating the reference embedding of the mesh and the plastic offsets of its elements.
2. **Merging –** Detect collisions and self-collisions. Modify the mesh to reflect topological merging events.
3. **Fracture –** Detect thinning and fracture. Split the mesh accordingly.
4. **Remeshing –** Locally improve the mesh to maintain a minimum threshold on tetrahedron quality.
5. **Surface mesh subdivision –** Subdivide parts of the mesh that might otherwise experience locking.
6. **Matrix assembly and solution –** Perform adaptive, implicit time integration to determine the velocities and pressures for the next time step. Optionally, compute heat diffusion.
7. **Revert surface mesh subdivision –** Reverse the mesh subdivision operations previously performed to avoid locking.
8. **Update the node positions**.
9. **Remeshing** (again).

3. RELATED WORK

Regular Eulerian grids are a common choice for fluid simulations in computer animation [Harlow and Welch 1965; Foster and Metaxas 1996; Stam 1999]. Grid-based Eulerian fluid simulators have been enhanced to support multi-phase flows [Hong and Kim 2003; Hong and Kim 2005; Losasso et al. 2006] including phase transitions [Carlson et al. 2002], viscoelastic [Goktekin et al. 2004] and hyperelastic behavior [Kamrin and Nave 2009], and even immersed rigid body simulation [Carlson et al. 2004]. Simulating accurate contact [Wang et al. 2005] and surface tension [Brackbill et al. 1992] requires special treatment.

Tetrahedral meshes [Feldman et al. 2005], octrees [Losasso et al. 2004], and hybrid meshes [Feldman et al. 2005] have been used to support varying resolution and accurate treatment of boundary conditions. In Arbitrary Lagrangian Eulerian (ALE) methods, the Eulerian discretization can change arbitrarily between time steps. This flexibility makes it possible to implement adaptivity [Losasso et al. 2004], accommodate arbitrarily oriented moving boundaries [Klingner et al. 2006], and accurately follow interfaces [Dai and Smith 2005; Chentanez et al. 2007].

Modeling free surfaces entails some method for tracking the motion of the surface boundary. Level-set methods store the surface as an implicit function which is advected with the velocity field [Osher and Fedkiw 2003]. These methods may cause excessive surface smoothing and are often augmented with tracker particles to help maintain surface detail [Foster and Fedkiw 2001; Enright et al. 2002; Enright et al. 2002; Losasso et al. 2008]. Volume-of-fluid methods are used for computational fluid dynamics problems [Kucharik et al. 2010], but they are rarely used in graphics as their focus is not on visual quality.

Other tracking approaches produce an explicit surface. For example, semi-Lagrangian advection produces a polygonal surface mesh each time step from an advected distance function of the previous surface mesh [Bargteil et al. 2006]. Some more recent methods directly advect the surface mesh, remeshing when necessary [Brochu and Bridson 2009; Brochu et al. 2010; Wojtan et al. 2010; Thürey et al. 2010].

Real-time animations of water drops have been performed with a deformable surface model. Each time step, the model uses an implicit mean curvature flow operator to produce surface tension effects, a contact angle operator to change droplet shapes on solid surfaces, and a set of mesh connectivity updates to handle topological changes and improve mesh quality [Zhang et al. 2012].

Lagrangian fluid simulations typically use meshless particle methods, such as Smoothed Particle Hydrodynamics (SPH) [Adams and Wicke 2009; Sin et al. 2009]. Meshless methods have also been used for the simulation of solids [Belytschko et al. 1996; Müller et al. 2004; Pauly et al. 2005; Gerszewski et al. 2009]. Such particle simulations can accommodate solids and liquids in the same framework, including phase transitions [Miller and Pearce 1989; Keiser et al. 2005; Becker et al. 2009].

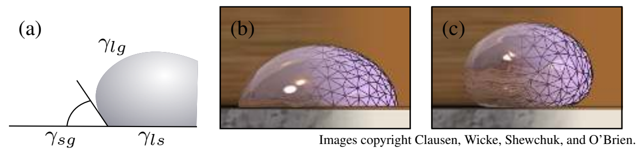

Fig. 2: Surface tension and contact angle. (a) The surface energy at an interface depends on the materials in contact and determines the contact angle. The images show drops resting on (b) a hydrophilic surface and (c) a hydrophobic surface.

Our work is similar to that of Misztal *et al.* on Lagrangian mesh-based simulations of liquids with a deformable simplicial complex [Misztal et al. 2012; Misztal et al. 2010; Erleben et al. 2011]. Like us, they also solve the weak form of the momentum equation with the standard $\mathcal{P}1$–$\mathcal{P}1$ finite element method (piecewise linear basis functions for both the velocity and pressure) in a single linear system. However, our work differs in several respects. Misztal *et al.* discretize the entire ambient space with an unstructured tetrahedral mesh, whereas we discretize only the material. The mesh of the surrounding air or vacuum allows Misztal *et al.* to treat collisions and topological changes as part of mesh maintenance, whereas we must explicitly detect collisions and merging events. Conversely, we avoid the cost of maintaining a mesh for the surrounding space.

Although elastic solids can be modeled with an Eulerian formulation [Goktekin et al. 2004; Levin et al. 2011], typically they are modeled with Lagrangian formulations [Gourret et al. 1989; Chen and Zeltzer 1992; Bro-Nielsen and Cotin 1996; Zhu et al. 1998; O'Brien and Hodgins 1999]. Linear corotated tetrahedral finite elements [Belytschko and Glaum 1979; Nour-Omid and Rankin 1991; Müller et al. 2002; Etzmuss et al. 2003; Müller and Gross 2004; Irving et al. 2004; Parker and O'Brien 2009] have been used widely in computer graphics, and we use them in this work with extensions to enforce incompressibility where needed [Irving et al. 2007].

There is a huge body of literature on adaptive methods, which improve efficiency by locally tailoring element sizes according to the physics. Budd, Huang, and Russell [2009] give a survey of *r-adaptive* methods, which move nodes to concentrate degrees of freedom where appropriate. For overviews of *h-adaptive* methods, which subdivide elements into smaller ones, see Oden and Demkowicz [1989] and Jones and Plassmann [1997]. In computer graphics, mesh refinement [Debunne et al. 2001; Capell et al. 2002; Otaduy et al. 2007; Sifakis et al. 2007; Martin et al. 2008; Wicke et al. 2010; Narain et al. 2012] and basis function refinement [Grinspun et al. 2002; Kaufmann et al. 2009] have been used to speed up simulations. Some methods simulate complex geometry by embedding it in coarse simulation meshes [Nesme et al. 2009; Kharevych et al. 2009].

Our present work takes advantage of prior work on methods for simulating large plastic flows in elastic materials [Bargteil et al. 2007; Wojtan and Turk 2008; Wojtan et al. 2009]. To maintain a viable simulation mesh as a solid reshapes itself, these methods periodically remesh the entire domain. Wicke *et al.* [2010] observe that better accuracy can be obtained by remeshing locally instead of globally. We build on their framework that simultaneously maintains a world-space mesh, a material-space embedding that minimizes the global internal elastic energy, and an individual rest space for each element to account for plastic flow. The mappings from the element rest spaces to the material-space embedding are called *plastic offsets*; these make it possible to represent materials that, because of plastic flow, do not have an embedding in space that is free of internal strains.

Fluids and solids are usually simulated by distinctly different methods that require significant effort to couple effectively [Guendelman et al. 2005; Chentanez et al. 2006]. To simulate phase transitions, the coupling method must support the transfer of material from one representation to the other [Losasso et al. 2006]. In contrast, we use the same mesh representation for both fluids and solids and can therefore accommodate their interactions without explicit coupling terms.

4. GOVERNING EQUATIONS

We use a unified physical framework to represent the movement of both solids and fluids. The time-dependent equations of motion are

$$\rho \frac{\partial^2 \mathbf{x}}{\partial t^2} = -\nabla \cdot \boldsymbol{\sigma} + \mathbf{f}_S + \mathbf{f}_V \ , \tag{1}$$

where $\mathbf{x}$ is the world position of a point on the simulated object, t time, ρ density, $\boldsymbol{\sigma} = \boldsymbol{\sigma}^e + \boldsymbol{\sigma}^v + \boldsymbol{\sigma}^P + \boldsymbol{\sigma}^T$ the sum of elastic, viscous, pressure, and thermal stresses, $\mathbf{f}_S$ the sum of external surface forces (surface tension and collision forces), and $\mathbf{f}_V$ the sum of external body forces (gravity and other force fields).

4.1 Elastic and Viscous Stresses

For isotropic materials Hooke's law governs the linear relationship between the stress tensor $\boldsymbol{\sigma}^e$ and the strain tensor $\boldsymbol{\epsilon}^e$,

$$\boldsymbol{\sigma}^e = \lambda^e \operatorname{tr}(\boldsymbol{\epsilon}^e)\mathbf{I} + 2\mu^e \boldsymbol{\epsilon}^e, \tag{2}$$

where the bulk modulus λ^e and shear modulus μ^e are the Lamé constants. We use Cauchy's linear strain

$$\boldsymbol{\epsilon}^e = \left(\nabla \mathbf{x} + \nabla \mathbf{x}^\mathsf{T}\right)/2 \ . \tag{3}$$

We model plasticity with the method described by Wicke *et al.* [2010] wherein changes to the rest shape are represented by evolving the material-space positions $\mathbf{u}$ over time. We note that Eq. 3 is valid for infinitesimal displacements only, so we use a corotational finite element method [Müller and Gross 2004].

The viscous stress tensor $\boldsymbol{\sigma}^v$ is

$$\boldsymbol{\sigma}^v = \lambda^v \operatorname{tr}(\boldsymbol{\epsilon}^v)\mathbf{I} + 2\mu^v \boldsymbol{\epsilon}^v, \tag{4}$$

where λ^v and μ^v are the bulk and shear viscosity, $\boldsymbol{\epsilon}^v$ is the strain rate tensor

$$\boldsymbol{\epsilon}^v = \left(\nabla \mathbf{v} + \nabla \mathbf{v}^\mathsf{T}\right)/2, \tag{5}$$

and $\mathbf{v}$ is velocity. These viscous terms are equivalent to the viscous terms in the Navier–Stokes equations. In the context of elasticity, μ^v is often called the damping coefficient.

4.2 Incompressibility and Pressure Stress

Ideal fluids do not resist shear, but many do resist compression quite strongly. One could model a fluid as an elastic object with $\mu^e = 0$ and λ^e set very high. However, enforcing incompressibility with large λ^e tends to cause stiffness and instability. Instead we enforce incompressibility through constraints on the velocity field, forcing it to be divergence-free.

$$\nabla \cdot \mathbf{v} = 0. \tag{6}$$

The force necessary to enforce this continuity equation is given by the divergence of the pressure stress $-\nabla \cdot \boldsymbol{\sigma}^P = -\nabla P$, so that the pressure P functions as the constraint's Lagrange multipliers. This approach works well for both incompressible fluids and volume-preserving elastic solids [Irving et al. 2007].

4.3 Surface Tension

Both fluids and solids can experience surface tension forces, which can be defined by the mechanical work $dW = \gamma\, dA$ required to increase the surface area by dA, where γ is the surface tension energy per unit surface area. The surface force at a point on the surface is then

$$\mathbf{f}_S = -\gamma \lim_{A\to 0} \frac{1}{A}\frac{\partial A}{\partial \mathbf{x}}\ , \tag{7}$$

where A is the area of a small surface patch around a point $\mathbf{x}$ on the surface and the derivative is taken with respect to movement of the surface point $\mathbf{x}$ in three dimensions.

Through the Laplace–Beltrami operator, Eq. (7) can be written $\mathbf{f}_S = 2\kappa_H \mathbf{n}$, where κ_H and $\mathbf{n}$ are the mean curvature and the surface normal at point $\mathbf{x}$, respectively [Gray 1998; Dierkes et al. 1992; Thürey et al. 2010]. Although the two expressions are in principle equivalent, it is in practice difficult to stably compute κ_H on a triangulated surface and numerical approximations to the Laplace–Beltrami operator can cause spurious damping. (For a comparison, see Brackbill *et al.* [1992] and Scardovelli and Zaleski [1999].) Therefore we prefer the expression based directly on the change in surface area. When applied to our tetrahedral discretization with linear basis functions (see Sec. 5) the nodal force from Eq. (7) on node i due to surface face j is given by

$$\mathbf{f}_i = -\gamma \frac{\partial}{\partial \mathbf{x}_i} A_j\ , \tag{8}$$

where $\mathbf{f}_i$, $\mathbf{x}_i$, and A_j are respectively the nodal force on node i, its position, and the area of surface face j.

The surface energy parameter γ depends on the materials on both sides of the interface. As shown in Fig. 2, there are different interface energies γ_{ls}, γ_{sg}, and γ_{lg} for liquid-solid, solid-gas, and liquid-gas interfaces. We do not add any constraint to the linear system to enforce expected contact angles (as *e. g.* Wang *et al.* [2005] do). Instead, these angles arise naturally as a result of the balance between surface tension forces and other forces such as gravity. In particular, the ratio between the surface energies determines the contact angle θ at rest: $\cos\theta = (\gamma_{ls} + \gamma_{sg})/\gamma_{lg}$.

Generally γ_{sg} is very small; we approximate it as zero without significant loss in accuracy. For each surface triangle, we determine which types of materials it is in contact with and choose the appropriate surface energy coefficient. To decide whether or not a node collides with the solid, we use a distance threshold set to 1% of the mesh's mean segment length.

4.4 Heat Flow

The behavior of the temperature T is modeled by the heat equation,

$$\frac{\partial T}{\partial t} + \nabla \cdot (\alpha_D \nabla T) = Q_S + Q_V, \tag{9}$$

where $\alpha_D = \kappa_T/(\rho\xi)$ is the thermal diffusivity, κ_T is the thermal conductivity, ρ is the density, ξ is the specific heat capacity, and Q_S and Q_V are surface and volume heat sources, respectively. This equation specifies that any change in temperature is the consequence of external heat sources and the divergence (flux) of the temperature gradient.

Variations of temperature within a material induce stresses, causing it to deform. If sufficiently high, these stresses can cause structural failure, especially for incompliant brittle materials with low thermal diffusivity such as glass. In linear thermoelasticity, the stress-strain relationship includes the thermal contribution $\boldsymbol{\epsilon}^T = \alpha_T(T - T_0)\mathbf{I}$ to the total strain and $\boldsymbol{\sigma}^T = \beta_T(T - T_0)\mathbf{I}$ to the total stress $\boldsymbol{\sigma}$, where α_T is the thermal expansion coefficient, $\beta_T = -\alpha_T(3\lambda^e + 2\mu^e)$ is the thermal coupling constant, and T_0 is the reference temperature at zero thermal stress [Landau and Lifshitz 1970].

Temperature variations and stresses both influence each other. First, temperature variations cause the material to deform by contributing to $\boldsymbol{\sigma}$ in Eq. 1. Second, viscous stresses cause thermoelastic heating, increasing the heat energy by $\mathrm{tr}(\boldsymbol{\sigma}\boldsymbol{\epsilon}^v)$. The heat flux, after we discard nonlinear terms, is

$$Q_V = \frac{\beta_T T}{\rho\xi}\,\mathrm{tr}\,(\boldsymbol{\epsilon}^v)\,. \tag{10}$$

A sufficiently high strain rate can make a material with high thermal expansion melt. Because Eq. 10 considers only the trace of the viscous strain, it does not model heating due to shear deformations.

Heating of a material by a hot plate and cooling by the ambient air are modeled by the surface source term

$$Q_S = \frac{h_{\mathrm{air}}}{\rho\xi}(T_{\mathrm{air}} - T), \tag{11}$$

where h_{air} is the heat transfer coefficient between the air and the material, and T_{air} is the temperature of the air.

5. DISCRETIZATION

We discretize the equations of motion and heat (Eqs. 1 and 9) with an implicit Euler time integration scheme and piecewise linear basis functions over a tetrahedral finite element mesh [Cook et al. 2001]. We combine both the equations of motion and the continuity equation in a single linear system where the unknowns are the node velocities $\mathbf{v}^{n+1}$ and pressures $\mathbf{p}^{n+1}$ at time $t^{n+1} = t^n + \Delta t$:

$$\begin{pmatrix} \frac{1}{\Delta t}\mathbf{M} + \Delta t\,\mathbf{K} + \mathbf{D} & \frac{1}{\Delta t}\mathbf{B}^\mathsf{T} \\ \frac{1}{\Delta t}\mathbf{B} & 0 \end{pmatrix} \begin{pmatrix} \mathbf{v}^{n+1} \\ \Delta t\,\mathbf{p}^{n+1} \end{pmatrix} = \begin{pmatrix} -\Delta t\,\mathbf{K}\mathbf{v}^n - \mathbf{f} \\ 0 \end{pmatrix} \tag{12}$$

where $\mathbf{M}$ is the (lumped) mass matrix, $\mathbf{K}$ is the stiffness matrix, $\mathbf{D}$ is the damping matrix (containing viscous terms), $\mathbf{B}$ and $\mathbf{B}^\mathsf{T}$ are the discretized divergence and gradient operators, and $\mathbf{f}$ is the external forces. The vectors with superscripts denote system-sized state vectors containing the values for all nodes in the system at the indicated time. The pressure vector $\mathbf{p}^{n+1}$ functions as Lagrange multipliers for the incompressibility constraints. We scale the constraints by $1/\Delta t$ to make the linear system's conditioning less dependent on the time step. After $\mathbf{v}^{n+1}$ is computed, the node positions in world space are updated as $\mathbf{x}^{n+1} = \mathbf{x}^n + \Delta t\,\mathbf{v}^{n+1}$.

The heat equation is modeled by a separate linear system whose solution is the new vector of nodal temperatures $\mathbf{t}^{n+1}$.

$$\left(\frac{1}{\Delta t}\mathbf{I} + \mathbf{L}\right)\mathbf{t}^{n+1} = \frac{1}{\Delta t}\mathbf{t}^n + \mathbf{q}, \tag{13}$$

where $\mathbf{I}$ is the identity matrix, $\mathbf{L}$ is the discretized Laplacian matrix including the thermal diffusivity, and $\mathbf{q}$ contains heat source terms.

If the material is plastic, or partially liquid, we assign new material space coordinates to the nodes by computing the configuration of the mesh that minimizes the total internal strain energy, as described by Wicke *et al.* [2010]. If the material is fully liquid, there is no internal strain, and we simply set the material space coordinates to the world coordinates.

We incorporate Dirichlet boundary conditions that are axis-aligned by eliminating the affected variables from the linear system. For non-aligned Dirichlet boundary conditions we rotate the nodes' degrees of freedom to match the constraint [Chentanez et al. 2009]. Otherwise, we use implicit penalty forces to enforce constraints.

Eq. 2 assumes a linear relationship between the stress and the strain tensor, which can cause unacceptable errors for large deformations. To remove these nonlinearities, we use the corotational linear finite element method to compute and assemble the stiffness matrix $\mathbf{K}$. This method, which has become a standard for simulations of large elastic deformations, factors the strain into rotation and extension/compression parts, computes the stresses from the second, rotation-invariant part, and rotates the stresses back into the proper coordinate frame [Etzmuss et al. 2003; Müller and Gross 2004; Parker and O'Brien 2009].

5.1 Choosing a Time Step

For overly large time steps, the world-space configuration of elements may become degenerate or inverted. Because the linear system (12) depends on Δt, there is no simple way to compute an admissible time step that would not cause inversion. Instead our integrator tries a target Δt and halves Δt if inversion occurs. This procedure iterates until an admissible step is found or Δt falls below a threshold $\Delta t_{\min}$. In the latter case, we treat inverted elements by other means. We remove most inverted elements whose volumes are small (below a specified threshold) by contracting their shortest edges. We permit the larger inverted elements to remain, but we treat them numerically with isotropic strain limiting [Wang et al. 2010] to maintain the stability of the simulation: we iteratively move the nodes of the world-space mesh to minimize a deformation energy function with respect to the material mesh that penalizes elements undergoing excessive strains. These operations introduce small errors into the solution, but they are rarely necessary and we have not observed undesirable visual artifacts from their use.

5.2 Incompressibility and Volume Conservation

We define a piecewise linear pressure field whose degrees of freedom lie at the mesh nodes and we enforce per-node incompressibility constraints, following Irving *et al.* [2007]. Although the constraints guarantee that the velocity field will be divergence-free at the beginning of the time step, they do not guarantee that the integral motion will be divergence-free. Irving *et al.* explicitly correct for volume loss or gain arising from discretization error by adding a correction term for each node $E_{x,i}$ on the right-hand side of the discretized continuity equation Eq. 6. Doing so assures that volume lost or added due to numerical inaccuracies is recovered over time.

Similarly, we store the rest volume for each element. The process for maintaining correct element volumes during remeshing is described in Sec. 6. As Irving *et al.* point out, recovering all the volume at once in the next step can cause instabilities in the simulation. We therefore clamp the volume recovered per time step to a fraction of the true element volume.

Enforcing node-based incompressibility by incorporating pressure terms in Eq. 12 can cause linear tetrahedral elements to lock because there are insufficient local degrees of freedom to represent a reasonable approximate solution. The most egregious form of locking occurs in a single tetrahedron whose volume cannot change. Such elements appear on the surface of the domain, especially in the presence of positional constraints and sharp corners. Fig. 3 illustrates such a circumstance in two dimensions. We use edge, face, and tetrahedron split operations to explicitly remove all elements with four vertices on the boundary, as well as all edges that connect surface vertices but are not themselves on the surface. This procedure introduces additional degrees of freedom where necessary to improve the flow of the material through the object. After each simulation step, we undo these splitting operations. Although this splitting locally reduces the mesh quality, we split edges, faces, and tetrahedra only at their barycenters and therefore still preserve a lower bound on the element quality.

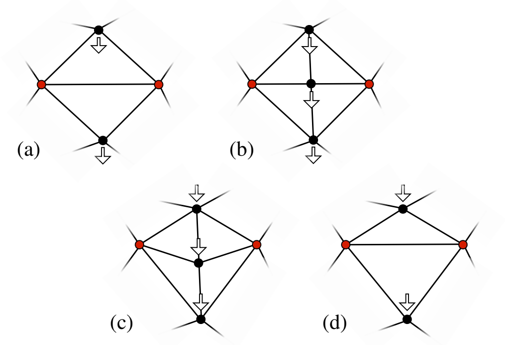

Fig. 3: (a) An example of a locking configuration in a two-dimensional triangular mesh. The red nodes are constrained not to move, and the volume preservation constraints keep the unconstrained nodes from moving vertically. No mass can be transferred between the two elements. (b), (c) After a split, the new node provides the necessary degree of freedom. (d) After the split is undone, volume has been transferred.

6. DYNAMIC LOCAL REMESHING

The main difficulty in Lagrangian fluid simulation is maintaining a high-quality mesh. This problem is significantly easier in two dimensions, where simulations of fluids have been performed with Lagrangian meshes [Cardoze et al. 2004; Cremonesi et al. 2011]. In three dimensions, global remeshing has been used to recreate a good mesh after Lagrangian advection [Bargteil et al. 2007; Chentanez et al. 2007; Wojtan and Turk 2008; Wojtan et al. 2009]. Unfortunately, global remeshing requires resampling of the velocity and strain fields after each time step, negating one of the greatest advantages of Lagrangian methods: their ability to maintain field values without smoothing by repeated interpolation. Local remeshing remedies this problem; Mauch *et al.* [2006] use it to model ballistic penetration in which materials undergo large plastic flows. However, state-of-the-art implementations of local remeshing have not performed well enough to simulate turbulent inviscid flow.

We build on the *Pulsar* implementation of local tetrahedral remeshing [Klingner and Shewchuk 2007; Wicke et al. 2010], and retain its paradigm of only repairing the mesh where it has degraded too much, while changing it as little as possible. To accommodate inviscid flowing liquids, we make a number of improvements, enabling local repair of meshes degraded by flow. In particular, while the previous implementation considered the surface to be piecewise linear, we consider it to be a smooth surface (potentially with sharp features), which the mesh approximates. This change in representation gives us more freedom in remeshing and also better preserves the surface. We also add new local improvement operations: the face and tetrahedron collapse operations.

6.1 Vertex Movement on Smooth Surfaces

The most challenging aspect of local tetrahedral remeshing is repairing the poor-quality elements close to the surface. Because we

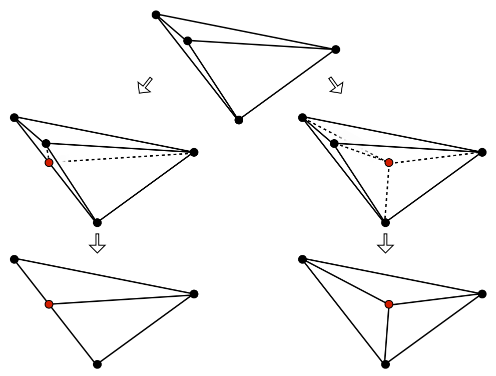

Fig. 4: New operations to remove a bad tetrahedron. Left: A face contraction. A new vertex is inserted on an edge, splitting the tetrahedron in two, and a newly created edge is immediately contracted, eliminating the tetrahedron. Right: A tetrahedron contraction. A new vertex is inserted on a face, splitting the tetrahedron in three, and the new interior edge is contracted.

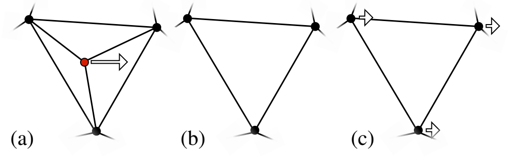

Fig. 5: Resampling can cause loss of momentum. (a) If only the red node has nonzero velocity, its momentum is lost when (b) the node is deleted from the mesh. (c) A compensating force is applied to the nodes for one time step, preserving the total momentum.

would like to change the surface of the object as little as possible, a much smaller set of operations is available to the remesher near the surface. Conversely, the more we allow the surface to change, the easier it is to mesh a domain, or repair a bad mesh.

We assume that the domain boundary is a piecewise smooth surface, and that the vertices of the tetrahedral mesh boundary are samples taken from the true boundary. We reconstruct a piecewise smooth surface from the samples with a method of Guennebaud and Gross [2007]. When surface vertices are moved to improve the quality of adjoining tetrahedra, their movement is constrained to a narrow band near the reconstructed smooth surface. We do not constrain vertices to lie precisely on the surface because a small amount of play sometimes yields significant improvements in the performance of the mesh improvement operations; see Section 4.3 of Wicke *et al.* [2010] for details.

We detect nonsmooth parts (feature edges and points) of the surface with a normal cone criterion (see [Kobbelt et al. 2001]; we use a scalar product threshold of 0.3). A vertex that lies on a feature edge can be moved during mesh improvement, but it is constrained to remain on or near the curve where the two surface patches intersect each other. A vertex that lies on a feature point, where three or more patches meet, is constrained to remain in a small region around its original position.

6.2 Face and Tetrahedron Collapse Operations

To supplement *Pulsar's* mesh improvement operations, we implemented two new compound remeshing operations, illustrated in Fig. 4. These operations consist of an edge split or face split, followed by an edge contraction. In our experiments, these two operations greatly improve the performance of the mesh improvement algorithm; they are often the only operations that can successfully remove bad tetrahedra close to the surface.

The building blocks of these operations, edge/face splits and edge contractions, were available to *Pulsar* before. However, *Pulsar* performs mesh improvement by assigning each mesh a quality score and performing hill-climbing in a search space of meshes. A compound operation that performs two basic operations before demanding a quality improvement can often traverse a valley in the search space to get out of a local maximum and reach a higher peak.

6.3 Volume and Momentum Conservation

After the simulation mesh is repaired, physical field values must be resampled from the old mesh to the new. We transfer properties as Wicke *et al.* [2010] do: variables stored at nodes are reinterpolated with the piecewise linear basis, and piecewise constant quantities stored per element (including the rest volume of each element) are resampled as weighted sums over overlapping elements, with the weights proportional to the overlap volume.

Changes made to the surface by the remesher can add or remove volume. Even in the interior, remeshing can change the total momentum—especially coarsening, as Fig. 5 illustrates. We enforce strict momentum and volume conservation by adding a correction term to the resampled values.

Consider a set E of source elements and a set E' of target elements. Let V_i be the volume of a source element $e_i \in E$, and let V_{ij} be the overlap volume between e_i and a target element $e'_j \in E'$. Given a piecewise constant field C_i defined on the source elements, we resample a value C'_j for a target element e'_j as the weighted sum

$$C'_j = \frac{1}{\sum_i V_{ij}} \sum_i V_{ij} \widetilde{C}_i, \text{ where} \tag{14}$$

$$\widetilde{C}_i = \frac{V_i}{\sum_k V_{ik}} C_i \tag{15}$$

and the summations are over all overlapping elements. Observe that $\widetilde{C}_i = C_i$ if the source tetrahedron is fully covered by target tetrahedra; otherwise, Eq. 15 scales C_i so that $\sum_j C'_j = \sum_i C_i$. One field we treat this way is the rest volume of each element (see Sec. 5.2), so the true total rest volume is conserved.

The same technique preserves momentum, although momentum is a piecewise linear field derived from nodal velocity values. We compute the momentum $\mathbf{m}_i$ of each source element i as its mass times its average nodal velocity, and likewise the momentum $\mathbf{m}'_j$ of each target element j. We resample $\mathbf{m}$ with Eq. 14, giving a desired momentum $\mathbf{m}_j$ for each target element. To compensate for lost momentum, we apply for one time step a force to each node of element j; the force is

$$\mathbf{f}_j = \frac{(\mathbf{m}_j - \mathbf{m}'_j)}{4\,\Delta t}. \tag{16}$$

This correction preserves linear momentum exactly, but not angular momentum. In our experiments, the angular momentum error induced by remeshing is less than the angular momentum error introduced by the corotational formulation, and neither is apparent in our results.

7. TOPOLOGICAL CHANGES

One of the main reasons why implicit representations are popular for surfaces of liquids is that splitting and merging operations require little or no extra implementation effort. Our Lagrangian meshes require explicit treatment of these operations, but the effort is compensated by reduced numerical viscosity, better volume preservation, and greater accuracy of surface evolution and collision detection. We use a local tetrahedron splitting operation modeled after fracture algorithms for solids [O'Brien and Hodgins 1999] and merging operations based on tetrahedron subdivision. Our method supports all the operations necessary to represent a flowing liquid undergoing topological changes.

7.1 Mesh Splitting and Material Fracture

Splitting has been extensively studied in the context of fracture and cutting of elastic materials [Bielser et al. 1999; Smith et al. 2001; Müller et al. 2001; O'Brien et al. 2002; Molino et al. 2004; Steinemann et al. 2006]. In fracture simulations, a material is typically split where the stress exceeds a specified threshold.

We use a combination of physical and geometric criteria to determine whether a material can split. For liquid elements, we initiate topological separation only at surface elements and only in concave regions—where at least one of the two surface curvatures is negative. Moreover, we require that there be a tensile principal stress whose magnitude is higher than a threshold $\gamma_{lg} \cdot \zeta$ where ζ is the capillary threshold and γ_{lg} the surface tension. Where both criteria are met, we fracture the material with a splitting plane perpendicular to the direction of highest principal stress.

Fracture can be triggered in inappropriate locations by field fluctuations caused by discretization, especially when it is driven by viscous stresses in turbulent liquids; these stresses can vary wildly between elements. To obtain physically plausible splitting, we have found it necessary to compute viscous stresses from a velocity field that has been smoothed over the 2-ring neighborhood of each vertex by a moving least squares approximation.

If the largest principal stress in an element e exceeds a specified threshold, we split it in three stages. 1) We compute a *splitting plane* from the stress field and split e with a sequence of edge split operations, thereby creating triangular *split faces* lying in the splitting plane. 2) We turn the split faces into boundary faces by duplicating vertices and split faces where appropriate, thereby initiating or propagating a crack through e. 3) Finally, we ensure that the mesh is manifold. Discussion of steps 1 and 2 follows; step 3 is described in Sec. 7.3.

Computing split faces. We choose a splitting plane that passes through the barycenter of e and is normal to the direction of largest principal stress. If any edge of e is on the material surface and the surface is highly concave at the edge, we assume that such an edge is the tip of a fracture surface that is propagating through the material, and call it a *snap edge*. How split faces are created depends on how many snap edges e has.

If e has no snap edge, we split each edge of e where it intersects the splitting plane, one edge at a time. An edge split cuts every adjoining tetrahedron into two. Afterward, the new vertices are connected by one or two split faces inside e, as illustrated in Fig. 6(a).

If e has exactly one snap edge, we modify the splitting plane so that it contains the edge, while changing the normal of the splitting plane as little as possible. A single edge split introduces a new node where the modified splitting plane intersects the edge of e opposite the snap edge, and cuts e into two elements that share a split face illustrated in Fig. 6(b).

Two snap edges fully determine a splitting plane that usually coincides with an existing face of the tetrahedron, which becomes the split face, as illustrated in Fig. 6(c). If e has more than two snap edges, or two disjoint snap edges, we choose as split faces two faces of e which together contain all the snap edges, and whose normals deviate least from the original principal stress direction.

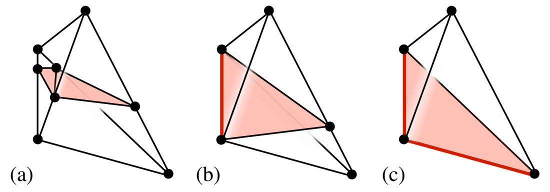

Fig. 6: Creating split faces (red) in the splitting plane. (a) Two split faces propagate through an element having no snap edge. (b) This element has one snap edge, from which a split face propagates. (c) Two snap edges. A preexisting face becomes a split face.

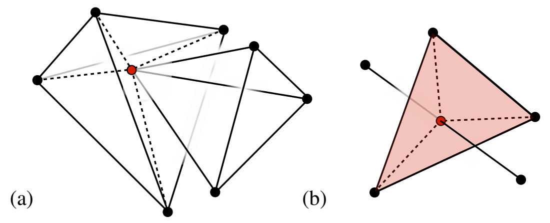

Fig. 7: Merging operations. (a) A tetrahedron split inserts a node of one mesh (red) into an element of the other. (b) An intersection between an edge of one mesh and a face of the other is treated by inserting a new node (red) into both.

Creating surfaces. To propagate the fracture, the split faces must become surface faces. For each node, we check whether the split faces separate its adjoining elements into two or more face-connected components. If so, we duplicate the node, assigning one copy to each face-connected component. The adjoining split faces are also duplicated, become boundary faces, and are no longer considered split faces.

After treating the nodes this way, we perform a similar test for all the edges of the split faces that remain. If split faces separate the elements adjoining an edge's interior into two or more face-connected components, we insert a new node at the midpoint of the edge, splitting the adjoining faces and elements. As above, this new node and the split faces that adjoin it are duplicated, and those split faces become boundary faces.

If any split face survives after these operations, we trisect it with a new node at its barycenter and treat it as above. Now, every split face has been transformed into two surface faces.

To ensure that cracks continue to propagate, we distribute residual stresses to every element that shares an edge with a split face and intersects the splitting plane. Following O'Brien and Hodgins [1999], the residual is added to the stress when determining which element to split next.

7.2 Merging Materials and Meshes

When collisions and self-collisions occur in a material that permits merging, as most fluids do, we locally stitch tetrahedra together in a manner that minimizes the changes. Previous work has used global remeshing to treat colliding tetrahedral meshes [Bargteil et al. 2007; Brochu and Bridson 2009; Wojtan et al. 2010], but, to the best of our knowledge, nobody has attempted conservative local stitching of tetrahedra.

Our merging algorithm identifies overlapping tetrahedra; subdivides them by inserting new nodes at edge-face intersections and inserting duplicate nodes in overlapping tetrahedra, as illustrated in Fig. 7, until the boundary of the overlap region is represented by mesh faces; deletes one of the two layers of tetrahedra covering the overlap region; and merges colocated nodes. The merging algorithm proceeds as follows.

1. Compute for each element e a set $\mathcal{E}(e)$ of elements that overlap e. Let $\mathcal{T} = \{e : \mathcal{E}(e) \neq \emptyset\}$ be the set of elements that overlap at least one other element.
2. We perform a *contraction pass* that contracts every edge of an element in $\mathcal{T}$ shorter than a threshold h, which is initialized to one fifth of the mean edge length in the mesh. We also contract any triangular face or tetrahedral element that has an altitude less than h as described in Sec. 6.2. If this pass changes the mesh, we recompute the overlap sets for the changed elements.
3. We perform an *element splitting pass*: if any element $e \in \mathcal{T}$ contains a node v of an overlapping element, we split e into four new tetrahedra at vertex v as illustrated in Fig. 7(a). Afterward, there are overlapping tetrahedra that share the vertex v. When an element is split, we restart the algorithm and return to the contraction pass. If v is close to a node, edge, or face of e, the split creates a short edge or a face or element with a short altitude that is removed in the subsequent contraction pass, moving v in the process.
4. When no nodes remain inside elements, we perform an *edge-face splitting pass*. If we find an edge-face intersection, we split both the edge and the face with a new node at the intersection point; see Fig. 7(b). After each edge-face split, we restart the algorithm and return to the contraction pass.

Edge-face splits can create new edge-face intersections, so it is not obvious whether this procedure terminates. A packing argument and our experience show that it always does. Each edge-face split inserts a new node which is common to the two overlapping regions. Intersections are only possible between an edge and a face that together have at least one non-common node. Because we enforce a minimum distance between nodes, only a finite number of nodes can be in the overlap region. Common nodes are never deleted: an edge contraction between a common node and a non-common node yields a common node, and face and element collapses leave the number of common nodes unchanged. Therefore, the overlapping region will eventually be packed with common nodes, removing the possibility of further edge-face intersections.

Once the unified mesh is created, quantities from the old mesh are resampled onto the new discretization. Piecewise linear field variables, which are defined per node, are taken to be the average of the linearly interpolated fields. We use the procedure described in Sec. 6.3 to transfer piecewise constant field variables, which are defined per element. This method accurately preserves the rest volume of the mesh. Consider a new element fully inside the overlap region. It overlaps exactly twice its own volume in old elements. If the rest volume of every source element is equal to its current volume, the new element is assigned a rest volume of exactly twice its current volume. Over time, the elements in the overlap region will expand and the original volume will be recovered. Typically the overlap region is thin and this adjustment happens quickly.

7.3 Enforcing Manifold Mesh Boundaries

After splitting or merging, the mesh might have a nonmanifold boundary. Abusing terminology, we call an edge or node *nonmanifold* if the tetrahedra having it for an edge or node do not form a single face-connected component. We also call a node nonmanifold if the adjoining boundary faces form more than one edge-connected component. The former kind of node we call *face-type nonmanifold*, and the latter kind we call *edge-type nonmanifold*. Fig. 8 illustrates the three kinds of nonmanifold simplex and the procedures for treating them. The example in Fig. 8(c) is difficult to visualize; think of it as the complement of the circumstance in Fig. 8(a), with empty space replaced by material and material replaced by empty space.

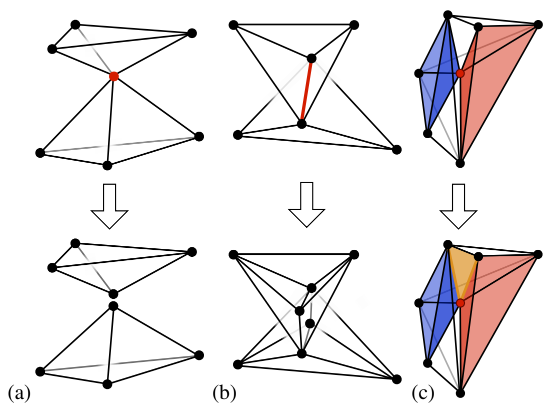

Fig. 8: (a) Removing a face-type nonmanifold vertex. The vertex is split into one vertex for each face-connected component of the adjoining elements. (b) Removing a nonmanifold edge by splitting it. The new vertex is a face-type nonmanifold vertex, and can be removed. (c) Removing an edge-type nonmanifold vertex. The shortest face-path (yellow) connecting the two separate surface regions (red and blue) is found. These faces are designated as split faces and are forced onto the surface as in Sec. 7.1 (in this case, splitting the yellow edges), connecting the red and blue surfaces.

A face-type nonmanifold vertex is easily fixed. For each face-connected component of the elements adjoining it, we create a copy of the vertex, which we assign to the elements of that component.

Once the vertices are fixed, we fix each nonmanifold edge by splitting it with a new vertex at its midpoint. This vertex is a face-type nonmanifold vertex, which we remove as described above. The new edges are no longer nonmanifold.

Edge-type nonmanifold vertices are the hardest to remove. They are pinch vertices where two surfaces meet—red and blue in Fig. 8(c). To remove them, we search for the shortest path of internal faces connecting an edge of the red surface to an edge of the blue surface. This path is yellow in Fig. 8(c). We use the method of Sec. 7.1 to turn these faces into boundary faces, thereby creating a tunnel connecting the two surfaces and removing the pinch. In Fig. 8(c), forcing the yellow face onto the surface involves splitting each yellow edge with a new vertex at its midpoint, duplicating the new midpoint vertices, and creating a tunnel connecting the blue and red surfaces.

8. SIMULATION RESULTS

To demonstrate the capabilities of our simulator, we ran several numerical experiments and example simulations. We first discuss measurements of numerical viscosity and comparisons with theory; then we turn to more complex, phenomenologically interesting examples. Table I shows information and timing data for all the simulations in the paper. Please also refer to the accompanying video for a better impression of the dynamic behavior.

Figure	Δt	τ_{total}	τ_{simulate}	$\tau_{\text{plasticity}}$	τ_{remesh}	τ_{subdiv}	τ_{solve}	V	N_t	N_n	h_{avg}
	4.97		12.6	0.55	4.64	0.56	0.76	520.47	6,425	1,826	0.879
Hydrophilic Drip	1.25	14	6.7	0.32	2.52	0.31	0.26	510.85	5,201	1,464	0.789
	5.00		69.7	1.02	57.48	1.12	3.14	523.74	8,104	2,304	0.953
	4.96		12.6	0.56	4.73	0.51	0.84	520.28	6,628	1,853	0.87
Hydrophobic Drip	1.25	14	6.7	0.33	2.53	0.31	0.26	510.70	5,201	1,464	0.78
	5.00		39.4	1.17	25.92	1.05	4.90	523.74	8,668	2,403	0.95
	0.99		40.1	2.21	1.36	1.74	3.53	48.518	21,412	6,923	0.27
Droplet Pinch-Off	0.12	55	29.3	1.61	1.07	1.36	1.45	37.215	17,175	5,505	0.25
	1.00		118.8	2.72	1.57	2.07	15.48	54.162	24,919	7,946	0.30
	0.98		49.3	2.53	21.05	2.06	4.73	51.639	25,518	8,074	0.26
Stream Breakup	0.50	34	29.2	1.59	12.24	1.44	1.46	43.800	19,167	6,579	0.22
	1.00		197.9	3.28	133.28	2.65	34.23	54.161	30,477	9,284	0.30
	10		17.1	0.39	5.74	5.16	0.81	0.0948	5,965	1,854	0.0507
Dam Break	10	48	4.2	0.18	1.35	0.23	0.20	0.0924	3,365	976	0.0367
	10		171.2	0.94	95.58	73.53	5.67	0.1060	11,828	4,279	0.0620
	49.9		285.3	4.71	213.62	4.1	22.59	0.0240	54,340	9,864	0.0139
Pipe	12.5	129	74.8	2.10	32.73	2.48	11.91	0.0231	42,498	8,063	0.0132
	50.0		4,888.3	7.33	4,819.80	6.88	55.62	0.0250	64,074	11,528	0.0162
	48.60		30.6	4.76	10.63	2.03	1.27	48.32	10,005	2,876	0.347
Melting Bunny	6.25	84	14.9	0.57	4.37	0.71	0.81	48.17	7,140	2,062	0.299
	50.00		138.0	18.82	86.49	5.97	7.25	48.58	13,266	3,880	0.402
	0.001		290.2	46.19	112.57	20.77	37.45	13.14	48,616	15,090	0.135
Lead Bullet	0.001	191	19.4	3.33	5.55	0.66	0.56	12.92	10,943	2,650	0.095
	0.001		1,105.3	289.53	555.57	80.69	218.10	13.41	127,904	41,410	0.214

Table I. : Timings and statistics for the simulations shown in this paper. All entries are mean/min/max except total simulation time. Total time is expressed in hours, time steps in 10^{-4} s, other times in s. Δt: time step; τ_{total}: total simulation time; τ_{simulate}: total simulation time per step; $\tau_{\text{plasticity}}$: time per step for plasticity simulation; τ_{remesh}: time per step for remeshing; τ_{subdiv}: time per step for surface element subdivision; τ_{solve}: time per step to solve the linear system. Additional statistics: V: total volume in world space; N_t: number of tetrahedra; N_n: number of nodes; h_{avg}: average edge length. Lengths and volumes are in mm and mm^3 for the drip examples, cm and cm^3 for the melting bunny and the lead bullet, and m and m^3 for the dam break and the pipe.

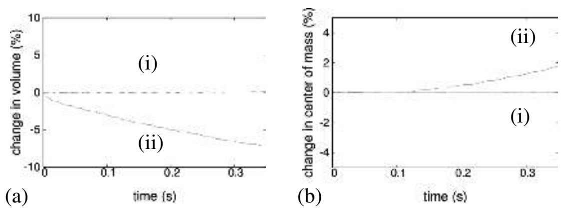

Fig. 9: Conservation of volume and momentum for an oscillating droplet. (a) Relative change in volume with respect to the rest volume, with (i) and without (ii) volume compensation. (b) Relative distance from center of mass to initial center of mass with respect to the equivalent radius R_0, with (i) and without (ii) linear momentum compensation. Time step was $\Delta t = 0.1$ ms; average element size was $h = 0.5$ mm.

8.1 Comparisons with Theory: Oscillating Droplets

To measure the numerical viscosity intrinsic to our method, we simulated a single oscillating inviscid spherical droplet for a number of element sizes and time steps. Because the experiment has a known analytic solution, this allows us to calculate the numerical viscosity of our method. The measured numerical viscosity is around $\mu = 10^{-5}$ Pa·s, or two orders of magnitude below the viscosity of water. The numerical viscosity of an Eulerian simulation (based on Caboussat *et al.* [2010], with 2.5 times smaller mesh elements) is $\mu = 6 \times 10^{-5}$ Pa·s.

The relative change in total volume with respect to the rest volume over the course of the simulations never exceeds 0.5% (Fig. 9(a)). Similarly, the relative distance from the center of mass to the initial center of mass with respect to the drop's nominal radius stays below 0.5%, which shows good conservation of momentum (Fig. 9(b)).

For these experiments we varied the time-step size and mesh resolution, and also compared with results from an Eulerian grid-based simulator. Detailed discussion and measurement data can be found in Appendix A.

8.2 Liquid Behavior at Small and Large Scales

Here we present two simulations of liquid behavior on the scale of millimeters and meters, respectively. Both examples demonstrate our method's handling of topological changes and ability to resolve fine details such as thin sheets, tendrils, and small droplets.

The first simulation models water dripping from a vertical pipette with a 4 mm inner diameter in the presence of gravity and surface tension, as illustrated in Figs. 1 and 10. The fluid descends through the pipette with a fixed velocity of 5 mm/s and drips onto a surface 20 mm below the pipette tip. The water has density $\rho = 997$ kg/m^3, surface tension $\gamma = 70.38 \times 10^{-3}$ N/m, viscosity $\mu = 10^{-3}$ Pa·s, and capillary fracture threshold $\zeta = 0.003$. The two figures show simulations with hydrophobic and hydrophilic surfaces, respectively. The hydrophilic surface has a liquid-solid surface tension energy of $\gamma_{ls} = 0$ N/m, inducing a rest contact angle of 90°. The hydrophobic surface has an adhesion energy of $\gamma_{ls} = 70.38$ N/m, inducing a rest contact angle of 0°. We model drag friction on the surface, with a friction coefficient of 10^{-5} kg/s. The time step is $\Delta t = 5 \times 10^{-4}$ s.

Both simulations exhibit fracture and merging events. The contact angles at which the liquid meets the surface are 90° and 0° for the hydrophilic and hydrophobic surfaces, as expected. These an-

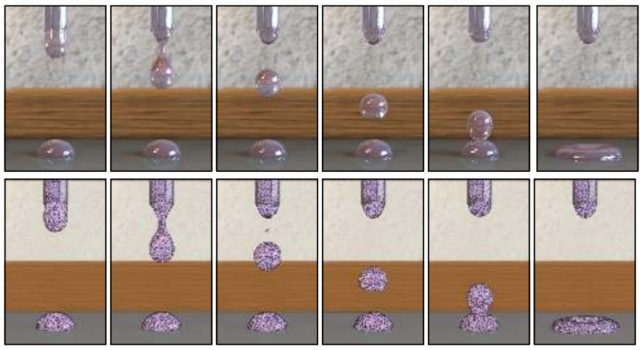

Fig. 10: A simulation of liquid dripping onto a hydrophilic surface.

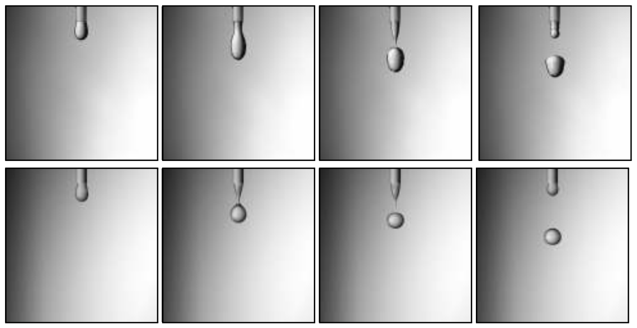

Fig. 11: Frames from a droplet pinch-off. We compare real video (top) with a simulated flow velocity of 0.11 mm/s (bottom).

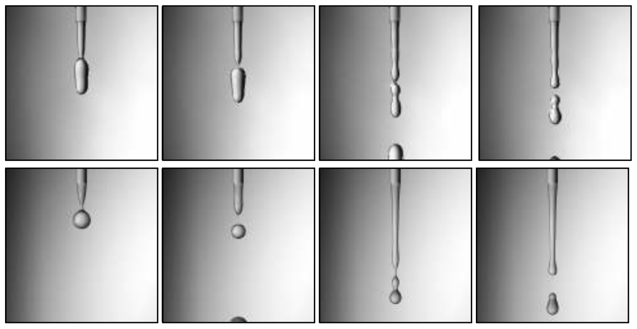

Fig. 12: Frames from a stream breaking up. We compare real video (top) with a simulated flow velocity of 0.6 mm/s (bottom).

gles arise naturally as consequences of the surface tension energy; we did not impose any explicit constraints on contact angles.

We compared two simulations with real video of dripping water. We tuned the material parameters to visually match the video, but we do not know the flow rate and our simulation neglects some important effects such as adhesive forces perpendicular to the flow due to finite wall thickness of the pipette, so the material parameters we chose are not close to those of water. The tube diameter is 1 mm, the liquid density is $\rho = 1.46$ kg/m^3, the surface tension is $\gamma = 13.6 \times 10^{-3}$ N/m, and the viscosity is $\mu = 3.6 \times 10^{-3}$ Pa·s. The first simulation was performed with a tube flow velocity of 0.11 mm/s. There are observable similarities between the real video and the simulation. The simulated drops have a shape similar to the real drops at the tube outlet. The effect of surface tension produces a growing droplet with a reduction of the radius of the upper part until breaking occurs. Afterward, a wave bounces back in the direction of the tube outlet. However, the real and simulated drips have different shapes when breaking (Fig. 11). In particular, the simulated drops are nearly spherical upon breaking, whereas in the real video the drops are more oblong. Another marked difference is that the real liquid oscillates at the surface of the liquid remaining at the tube outlet and on the surface of the detached droplet after breaking. Even with a very fine mesh, we could not reproduce these oscillations. We speculate that this difference might be related to our use of slip boundary conditions inside the simulated tube. Similar differences and similarities appear between the real and simulated breakup of a liquid stream with a flow velocity of 0.6 mm/s (Fig. 12).

The macroscopic behavior of water is illustrated by a dam-break simulation which releases a rectangular 1 m $\times$ 0.5 m $\times$ 0.2 m pool of liquid to flow into a rectangular 2 m $\times$ 2 m $\times$ 0.4 m container (Fig. 13). We use the same material parameters for water listed above, and a time step of 1 ms. Observe that the remesher resizes the tetrahedra according to the surface curvature, enabling finely resolved details without unnecessarily many elements. Features such as small drops, tendrils, and thin sheets are well resolved, as are interesting behaviors such as splashing, merging, and sheet breakup. A crucial benefit of our method is that it maintains thin sheets without volume loss, which is difficult for Eulerian methods.

Fig. 14 shows an example of a more turbulent flow. A fluid of density $\rho = 997$ kg/m^3 and viscosity $\mu = 10^{-6}$ Pa·s is injected with a velocity of $\mathrm{v}_{\mathrm{inlet}} = 0.02$ m/s into a cylindrical pipe of radius $\mathrm{r}_h = 0.1$ m and length 0.75 m, whereupon it strikes a spherical obstacle of radius 0.03 m. The Reynolds number for this experiment is 4×10^6. We apply no-slip boundary conditions to the spherical obstacle. As the simulation proceeds, we generate new tetrahedra at the pipe inlet and remove tetrahedra past the pipe outlet. A postprocess uses the computed flow velocities to advect particles for visualization. The motion of the particles helps to reveal that clouds of fluid detach themselves at regular intervals in a manner characteristic of this scenario.

8.3 Melting

Two examples demonstrate our algorithm's ability to cope with liquid and solid elements in a single mesh, as well as phase transitions between them. In Fig. 15, the Stanford Bunny is heated by a ground plate with a temperature of 353.5 K and a heat transfer coefficient of $h_T = 221$ W/(m^{2}K). The bunny is roughly 7 cm tall with an initial temperature of 253.5 K. Below the melting temperature $T_m = 273.5$ K, we model the material as perfectly elastic with an elastic modulus of 10^5 Pa. The plasticity of the bunny increases as the temperature rises. Above T_m, we model the material as a liquid with viscosity $\mu = 30 \times 10^{-3}$ Pa·s and a surface tension of $\gamma = 70.38 \times 10^{-3}$ N/m. We simulate heat diffusion with a heat capacity of $\xi = 1{,}960$ J/K and a thermal conductivity of $\kappa_T = 35$ W/(m·K). The melted liquid fills the square container and naturally settles at a 90° contact angle. All the material in the simulation is part of a single mesh, although the material parameters vary throughout the mesh. No coupling terms or special interface code is needed for the solid-fluid interaction.

Fig. 16 depicts a simulation of a 5 cm long lead bullet striking a perfectly rigid wall with a velocity of 200 m/s. Nodes in contact with the wall have the normal components of their velocities set to zero. As the bullet advances, the large strain rate causes its tip to melt. It has an initial temperature of 298.5 K, a thermal expansion coefficient of $\alpha_T = 29 \times 10^{-6}$ K^{-1}, a heat capacity of

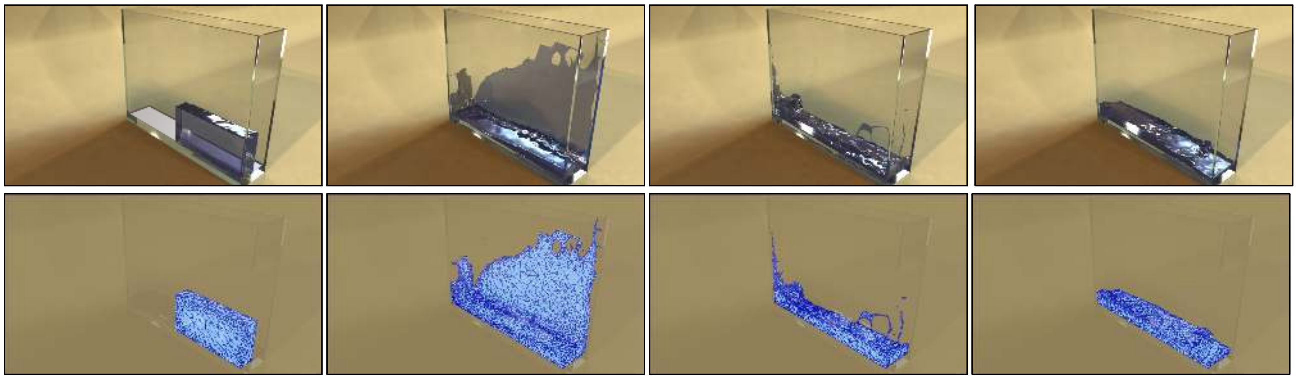

Fig. 13: Frames from a dam-break simulation.

$\xi = 130$ J/K, and a thermal conductivity of $\kappa_T = 35$ W/(m·K). Below the melting temperature of 600 K, we model the material as elastoplastic with a tensile modulus of 16 GPa and a density of $\rho = 11{,}300$ kg/m^3. Above the melting temperature, the material is considered to be liquid with a density of $\rho = 10{,}210$ kg/m^3, a viscosity of $\mu = 20 \times 10^{-3}$ Pa·s, and a surface tension of 430×10^{-3} N/m. The time step is $\Delta t = 10^{-7}$ s. To model the deformation of the steel plate and create a more interesting melt pattern, we placed a small conical dent on the plate, centered on the point of impact. The molten lead is pushed outward and forms a thin sheet.

9. DISCUSSION AND CONCLUSIONS

By using fully Lagrangian tetrahedral meshes, we can simulate materials ranging from stiff solids to viscoplastic metals to inviscid fluids, as well as interactions and phase changes among them, within a single framework and without explicitly coupling different materials or phases.

Our numerical results demonstrate that a Lagrangian discretization of fluids yields high accuracy even in simulations with moderate resolutions. Perhaps the most telling result is that the measured numerical viscosity is two orders of magnitude less than that of water. What little numerical viscosity arises is caused by resampling; our dynamic meshing algorithms help us do as little of that as possible. In contrast, simulators employing Eulerian discretizations, or even Lagrangian discretizations coupled with global remeshing, resample all the velocity values at every time step, introducing high numerical viscosity.

The Lagrangian mesh provides an explicit, physically meaningful, triangulated surface. Its value is demonstrated by the accuracy with which our method can model surface tension effects—as the oscillating droplet simulations show—and our ability to reproduce emergent behavior such as liquid-solid contact angles.

Our method guarantees perfect long-term preservation of volume, locally and globally with only minor short-lived fluctuations, by doing careful bookkeeping of the rest volume, especially during remeshing operations. We are able to do this because our method does not maintain a traditional material-space mesh that dictates the rest volumes; rather, it uses the more flexible representation of Bargteil *et al.* [2007], as modified by Wicke *et al.* [2010], that stores an independent rest configuration for each element. In stark contract to Eulerian approaches, our volume preservation works flawlessly even for large thin liquid sheets. By similar means, we preserve linear momentum.

Our element subdivision scheme removes the threat of element locking from simulations of incompressible liquids, even around thin sheets and tendrils.

9.1 Limitations

Perhaps the biggest limitation of Lagrangian meshes is that the permissible time step is limited by the threat of element inversions. While the restrictions on the time step are not as onerous as those imposed by the Courant–Friedrichs–Lewy condition for explicit time integration in a finite difference setting, they can be significant. We address this problem by occasionally allowing inversions to occur, and fixing them afterward. This procedure surely incurs errors, but the errors have been too small to notice visually. Their numerical effects are hard to quantify, as simulations with known analytical solutions are too simple to cause this problem.

Explicit treatment of splitting and merging events is a burden for the programmer, although their running time is only a small fraction of the total simulation time. We believe that for simulations where accuracy matters, the effort is worth it—especially for simulations where surface effects or preservation of thin features are essential.

The cost and programming effort of maintaining a high-quality tetrahedral mesh are high. We are distributing the updated *Pulsar* dynamic mesher, which we hope will enable many researchers to implement Lagrangian fluid simulations. Yet, even the most advanced remeshing tool cannot repair meshes whose surfaces have dihedral angles close to zero. Such circumstances occur in the simulations that appear in this paper; fortunately, implicit time integration is not sensitive to small angles, and we observed no artifacts because of these problems. Surface tension quickly resolves such surface configurations. However, while they persist, they substantially increase the running time of mesh repair.

Finally, although our method never forgets volume (the rest volume stays constant to machine precision), the current volume can vary somewhat. These fluctuations are greatest following collisions; our collision handler deforms the mesh surfaces to a collision-free state. The element rest volumes do not change, so the mesh gradually recovers the lost volume over time. We could drastically reduce the volume fluctuations by reversing time steps in which collisions occur, then repeating them while enforcing constraints.

We are excited about the new avenues of research opened up by Lagrangian mesh simulations. We believe they create opportunities for more accurate fluid-solid and fluid-fluid interface simulations, because multiple materials and the interfaces between them can be represented within a single mesh. We would like to accurately re-

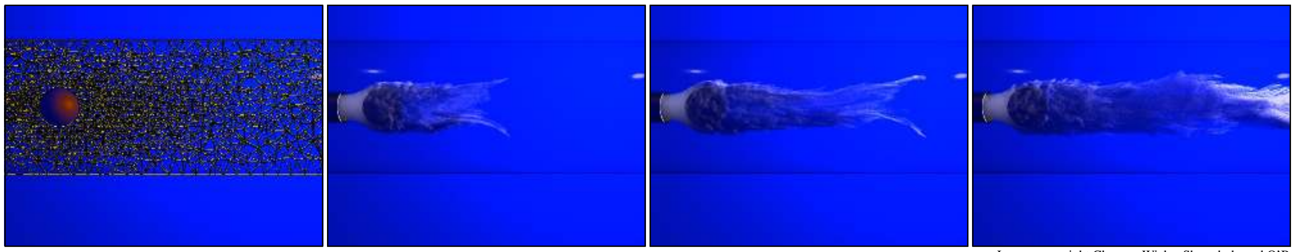

Fig. 14: Liquid is injected with a velocity of 0.02 m/s into a pipe of radius 0.1 m. The flow hits a spherical obstacle (with no-slip boundary conditions) and becomes turbulent. The simulation recreates smoke packets detaching at regular intervals, as expected. The initial mesh is shown at left. The mesh moves with the flow and is adaptively refined to have higher resolution around the obstacle.

produce phenomena such as gases bubbling through fluids and the mixing of petroleum and water, and to simulate interactions between viscous fluids and elastic solids as in a beating heart or a surgical procedure.

REFERENCES

ADAMS, B. AND WICKE, M. 2009. Meshless approximation methods and applications in physics based modeling and animation. In *Eurographics 2009 Tutorials*. 213–239.

BARGTEIL, A. W., GOKTEKIN, T. G., O'BRIEN, J. F., AND STRAIN, J. A. 2006. A semi-Lagrangian contouring method for fluid simulation. *ACM Trans. Graphics 25,* 1, 19–38.

BARGTEIL, A. W., WOJTAN, C., HODGINS, J. K., AND TURK, G. 2007. A finite element method for animating large viscoplastic flow. *ACM Trans. Graphics 26,* 3, 16:1–16:8.

BECKER, M., IHMSEN, M., AND TESCHNER, M. 2009. Corotated SPH for deformable solids. In *Proc. Eurographics Workshop on Natural Phenomena*. 27–34.

BELYTSCHKO, T. AND GLAUM, L. W. 1979. Applications of higher order corotational stretch theories to nonlinear finite element analysis. *Computers & Structures 10,* 1–2, 175–182.

BELYTSCHKO, T., KRONGAUZ, Y., ORGAN, D., AND FLEMING, M. 1996. Meshless methods: An overview and recent developments. In *Comp. Meth. Appl. Mech. Eng.* 3–47.

BIELSER, D., MAIWALD, V. A., AND GROSS, M. H. 1999. Interactive cuts through 3-dimensional soft tissue. *Computer Graphics Forum 18,* 3, 31–38.

BRACKBILL, J., KOTHE, D., AND ZEMACH, C. 1992. A continuum method for modeling surface tension. *J. Comp. Phys. 100*, 335–354.

BRO-NIELSEN, M. AND COTIN, S. 1996. Real-time volumetric deformable models for surgery simulation using finite elements and condensation. *Computer Graphics Forum 15,* 3, 57–66.

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

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

BUDD, C. J., HUANG, W., AND RUSSELL, R. D. 2009. Adaptivity with moving grids. In *Acta Numerica 2009*. Vol. 18. 1–131.

CABOUSSAT, A., CLAUSEN, P., AND RAPPAZ, J. 2010. Numerical simulation of two-phase flow with interface tracking by adaptive Eulerian grid subdivision. *Mathematical and Computer Modelling 55*, 490–504.

CAPELL, S., GREEN, S., CURLESS, B., DUCHAMP, T., AND POPOVIĆ, Z. 2002. A multiresolution framework for dynamic deformations. In *Proc. Symp. Comput. Animation*. 41–48.

CARDOZE, D., CUNHA, A., MILLER, G. L., PHILLIPS, T., AND WALKINGTON, N. J. 2004. A Bézier-based approach to unstructured moving meshes. In *Proc. Twentieth Annual Symposium on Computational Geometry*. Brooklyn, New York, 310–319.

CARLSON, M., MUCHA, P. J., AND TURK, G. 2004. Rigid fluid: Animating the interplay between rigid bodies and fluid. *ACM Trans. Graphics 23,* 3, 377–384.

CARLSON, M., MUCHA, P. J., VAN HORN III, R. B., AND TURK, G. 2002. Melting and flowing. In *Proc. Symp. Comput. Animation*. 167–174.

CHEN, D. T. AND ZELTZER, D. 1992. Pump it up: Computer animation of a biomechanically based model of muscle using the finite element method. In *Proc. SIGGRAPH*. 89–98.

CHENTANEZ, N., ALTEROVITZ, R., RITCHIE, D., CHO, L., HAUSER, K. K., GOLDBERG, K., SHEWCHUK, J. R., AND O'BRIEN, J. F. 2009. Interactive simulation of surgical needle insertion and steering. *ACM Trans. Graphics 28,* 3, 88:1–88:10.

CHENTANEZ, N., FELDMAN, B. E., LABELLE, F., O'BRIEN, J. F., AND SHEWCHUK, J. R. 2007. Liquid simulation on lattice-based tetrahedral meshes. In *Proc. Symp. Comput. Animation*. 219–228.

CHENTANEZ, N., GOKTEKIN, T. G., FELDMAN, B. E., AND O'BRIEN, J. F. 2006. Simultaneous coupling of fluids and deformable bodies. In *Proc. Symp. Comput. Animation*. 83–89.

COOK, R. D., MALKUS, D. S., PLESHA, M. E., AND WITT, R. J. 2001. *Concepts and Applications of Finite Element Analysis*, Fourth ed. John Wiley & Sons, New York.

CREMONESI, M., FRANGI, A., AND PEREGO, U. 2011. A Lagrangian finite element approach for the simulation of water-waves induced by landsides. *Computers and Structures 89,* 11–12 (June), 1086–1093.

DAI, M. AND SMITH, D. P. 2005. Adaptive tetrahedral meshing in free-surface flow. *J. Comp. Phys. 208,* 1, 228–252.

DEBUNNE, G., DESBRUN, M., CANI, M.-P., AND BARR, A. H. 2001. Dynamic real-time deformations using space & time adaptive sampling. In *Proc. SIGGRAPH 2001*. 31–36.

DIERKES, U., HILDEBRANDT, S., KUESTER, A., AND WOHLRAB, O. 1992. *Minimal Surfaces (I).* Springer-Verlag.

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

ENRIGHT, D. P., MARSCHNER, S. R., AND FEDKIW, R. P. 2002. Animation and rendering of complex water surfaces. *ACM Trans. Graphics 21,* 3, 736–744.

ERLEBEN, K., MISZTAL, M. K., AND BÆRENTZEN, J. A. 2011. Mathematical foundation of the optimized-based fluid animation method. In *Proc. Symp. Comput. Animation*. 101–110.

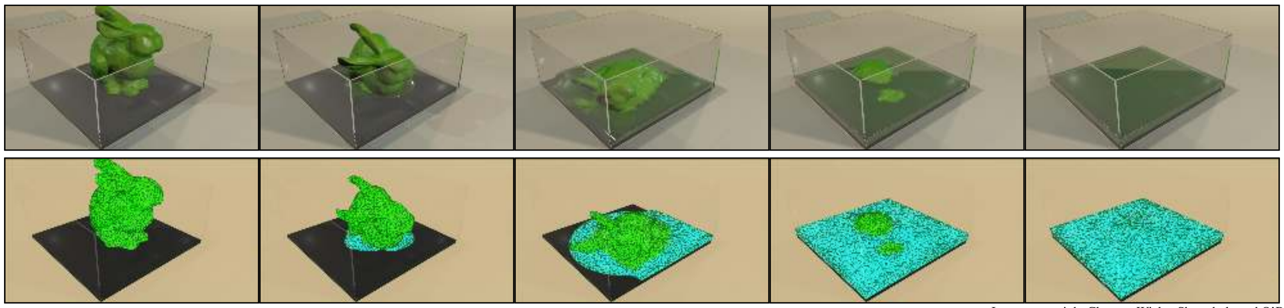

Fig. 15: The Stanford Bunny melts on a hot plate.

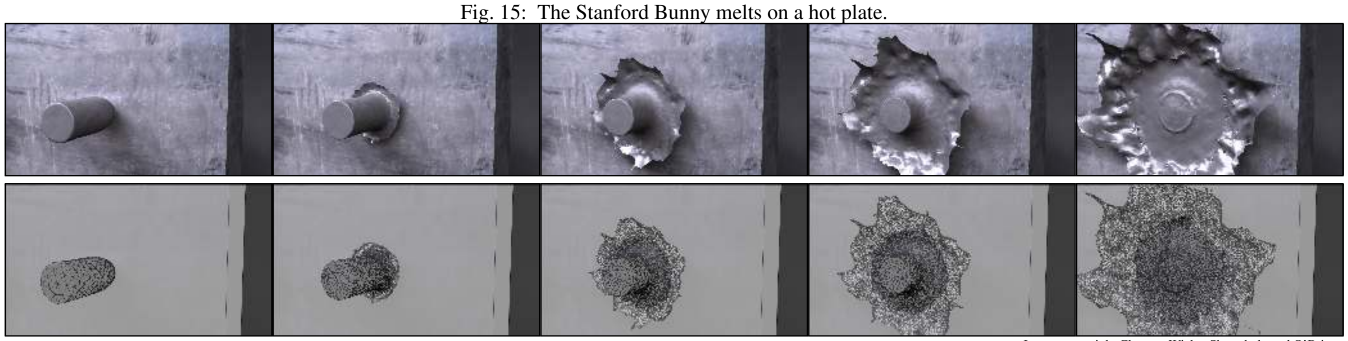

Fig. 16: A simulation of the impact of a lead bullet against a rigid wall. Thermoelastic heating causes the bullet to melt and spread into a thin sheet. The frames in the bottom row show the dynamically changing tetrahedral mesh.

ETZMUSS, O., KECKEISEN, M., AND STRASSER, W. 2003. A fast finite element solution for cloth modelling. In *Pacific Graphics*. 244–251.

FELDMAN, B. E., O'BRIEN, J. F., AND KLINGNER, B. M. 2005. Animating gases with hybrid meshes. *ACM Trans. Graphics 24,* 3 (Aug.), 904–909.

FELDMAN, B. E., O'BRIEN, J. F., KLINGNER, B. M., AND GOKTEKIN, T. G. 2005. Fluids in deforming meshes. In *Proc. Symp. Comput. Animation*. 255–260.

FOSTER, N. AND FEDKIW, R. 2001. Practical animation of liquids. *ACM Trans. Graphics 20,* 3, 23–30.

FOSTER, N. AND METAXAS, D. 1996. Realistic animation of liquids. *Graphical Models and Image Processing 58,* 5 (Sept.), 471–483.

GERSZEWSKI, D., BHATTACHARYA, H., AND BARGTEIL, A. W. 2009. A point-based method for animating elastoplastic solids. In *Proc. Symp. Comput. Animation*. 133–138.

GOKTEKIN, T. G., BARGTEIL, A. W., AND O'BRIEN, J. F. 2004. A method for animating viscoelastic fluids. *ACM Trans. Graphics 23,* 3, 463–468.

GOURRET, J.-P., THALMANN, N. M., AND THALMANN, D. 1989. Simulation of object and human skin deformations in a grasping task. In *Proc. SIGGRAPH*. 21–30.

GRAY, A. 1998. *Modern Differential Geometry of Curves and Surfaces with Mathematica*. CRC Press.

GRINSPUN, E., KRYSL, P., AND SCHRÖDER, P. 2002. CHARMS: A simple framework for adaptive simulation. *ACM Trans. Graphics 21,* 3, 281–290.

GUENDELMAN, E., SELLE, A., LOSASSO, F., AND FEDKIW, R. 2005. Coupling water and smoke to thin deformable and rigid shells. *ACM Trans. Graphics 24,* 3 (Aug.), 973–981.

GUENNEBAUD, G. AND GROSS, M. 2007. Algebraic point set surfaces. *ACM Trans. Graphics 26,* 3, 23.1–23.9.

HARLOW, F. H. AND WELCH, J. E. 1965. Numerical calculation of time-dependent viscous incompressible flow of fluid with a free surface. *The Physics of Fluids 8,* 12, 2182–2189.

HONG, J.-M. AND KIM, C.-H. 2003. Animation of bubbles in liquid. *Computer Graphics Forum 22,* 3 (Sept.), 253–262.

HONG, J.-M. AND KIM, C.-H. 2005. Discontinuous fluids. *ACM Trans. Graphics 24,* 3 (Aug.), 915–920.

IRVING, G., SCHROEDER, C., AND FEDKIW, R. 2007. Volume conserving finite element simulations of deformable models. *ACM Trans. Graphics 26,* 3, 13:1–13:6.

IRVING, G., TERAN, J., AND FEDKIW, R. 2004. Invertible finite elements for robust simulation of large deformation. In *Proc. Symp. Comput. Animation*. 131–140.

JONES, M. T. AND PLASSMANN, P. E. 1997. Adaptive refinement of unstructured finite-element meshes. *Finite Elements in Analysis and Design 25*, 41–60.

KAMRIN, K. AND NAVE, J.-C. 2009. An Eulerian approach to the simulation of deformable solids: Application to finite-strain elasticity. arXiv:0901.3799v2 [cond-mat.soft].

KAUFMANN, P., MARTIN, S., BOTSCH, M., GRINSPUN, E., AND GROSS, M. 2009. Enrichment textures for detailed cutting of shells. *ACM Trans. Graphics 28,* 3 (July), 50:1–50:10.

KEISER, R., ADAMS, B., GASER, D., BAZZI, P., DUTRÉ, P., AND GROSS, M. 2005. A unified Lagrangian approach to solid-fluid animation. In *Eurographics Symp. Point-Based Graphics*. 125–133.

KHAREVYCH, L., MULLEN, P., OWHADI, H., AND DESBRUN, M. 2009. Numerical coarsening of inhomogeneous elastic materials. *ACM Trans. Graphics 28,* 3 (Aug.), 51:1–51:8.

KLINGNER, B. M., FELDMAN, B. E., CHENTANEZ, N., AND O'BRIEN, J. F. 2006. Fluid animation with dynamic meshes. *ACM Trans. Graphics 25,* 3, 820–825.

KLINGNER, B. M. AND SHEWCHUK, J. R. 2007. Aggressive tetrahedral mesh improvement. In *Proc. 16th International Meshing Roundtable*. 3–23.

KOBBELT, L. P., BOTSCH, M., SCHWANECKE, U., AND SEIDEL, H.-P. 2001. Feature sensitive surface extraction from volume data. In *Proc. SIGGRAPH*. 57–66.

KUCHARIK, M., GARIMELLA, R. V., SCHOFIELD, S. P., AND SHASHKOV, M. J. 2010. A comparative study of interface reconstruction methods for multi-material ALE simulations. *J. Comp. Phys. 229,* 7, 2432–2452.

LAMB, H. 1932. *Hydrodynamics*. Cambridge University Press.

LANDAU, L. AND LIFSHITZ, E. 1970. *Theory of Elasticity*. Pergamon Press, New York.

LEVIN, D. I. W., LITVEN, J., JONES, G. L., SUEDA, S., AND PAI, D. K. 2011. Eulerian solid simulation with contact. *ACM Trans. Graphics 30,* 4 (Aug), 36:1–36:10.

LOSASSO, F., GIBOU, F., AND FEDKIW, R. 2004. Simulating water and smoke with an octree data structure. *ACM Trans. Graphics 23,* 3 (Aug.), 457–462.

LOSASSO, F., IRVING, G., GUENDELMAN, E., AND FEDKIW, R. 2006. Melting and burning solids into liquids and gases. *IEEE Trans. Vis. Comp. Graphics 12,* 3 (May/June), 343–352.

LOSASSO, F., SHINAR, T., SELLE, A., AND FEDKIW, R. 2006. Multiple interacting liquids. *ACM Trans. Graphics 25,* 3 (July), 812–819.

LOSASSO, F., TALTON, J., KWATRA, N., AND FEDKIW, R. 2008. Two-way coupled SPH and particle level set fluid simulation. *IEEE Trans. Vis. Comp. Graphics 14,* 4 (July/Aug.), 797–804.

MARTIN, S., KAUFMANN, P., BOTSCH, M., WICKE, M., AND GROSS, M. 2008. Polyhedral finite elements using harmonic basis functions. *Computer Graphics Forum 27,* 5, 1521–1529.

MAUCH, S., NOELS, L., ZHAO, Z., AND RADOVITZKY, R. A. 2006. Lagrangian simulation of penetration environments via mesh healing and adaptive optimization. In *Proc. 25th Army Science Conference*.

MILLER, G. AND PEARCE, A. 1989. Globular dynamics: A connected particle system for animating viscous fluids. *Computers & Graphics 13,* 3, 305–309.

MISZTAL, M. K., BRIDSON, R., ERLEBEN, K., BÆRENTZEN, J. A., AND ANTON, F. 2010. Optimization-based fluid simulation on unstructured meshes. In *Proc. 7th Workshop on Virtual Reality Interaction and Physical Simulation*. 11–20.

MISZTAL, M. K., ERLEBEN, K., BARGTEIL, A. W., FURSUND, J., CHRISTENSEN, B. B., BÆRENTZEN, J. A., AND BRIDSON, R. 2012. Multiphase flow of immiscible fluids on unstructured moving meshes. In *Proc. Symp. Comput. Animation*. 97–106.

MOLINO, N., BAO, Z., AND FEDKIW, R. 2004. A virtual node algorithm for changing mesh topology during simulation. *ACM Trans. Graphics 23,* 3, 385–392.

MÜLLER, M., DORSEY, J., MCMILLAN, L., JAGNOW, R., AND CUTLER, B. 2002. Stable real-time deformations. In *Proc. Symp. Comput. Animation*. 49–54.

MÜLLER, M. AND GROSS, M. H. 2004. Interactive virtual materials. In *Proc. Graphics Interface*. 239–246.

MÜLLER, M., KEISER, R., NEALEN, A., PAULY, M., GROSS, M., AND ALEXA, M. 2004. Point based animation of elastic, plastic and melting objects. In *Proc. Symp. Comput. Animation*. 141–151.

MÜLLER, M., MCMILLAN, L., DORSEY, J., AND JAGNOW, R. 2001. Real-time simulation of deformation and fracture of stiff materials. In *Proc. Eurographics Workshop on Computer Animation and Simulation*. 113–124.

NARAIN, R., SAMII, A., AND O'BRIEN, J. F. 2012. Adaptive anisotropic remeshing for cloth simulation. *ACM Trans. Graphics 31,* 6 (Nov.), 152:1–152:10.

NESME, M., KRY, P. G., JEŘÁBKOVÁ, L., AND FAURE, F. 2009. Preserving topology and elasticity for embedded deformable models. *ACM Trans. Graphics 28,* 3 (July), 52:1–52:9.

NOUR-OMID, B. AND RANKIN, C. 1991. Finite rotation analysis and consistent linearization using projectors. *Comp. Meth. Appl. Mech. Eng. 93,* 3, 353–384.

O'BRIEN, J. F., BARGTEIL, A. W., AND HODGINS, J. K. 2002. Graphical modeling and animation of ductile fracture. *ACM Trans. Graphics 21,* 3, 291–294.

O'BRIEN, J. F. AND HODGINS, J. K. 1999. Graphical modeling and animation of brittle fracture. In *Proc. SIGGRAPH*. 137–146.

ODEN, J. T. AND DEMKOWICZ, L. F. 1989. Advances in adaptive improvements: A survey of adaptive finite element methods in computational mechanics. In *State-of-the-Art Surveys on Computational Mechanics*. The American Society of Mechanical Engineers, 441–467.

OSHER, S. AND FEDKIW, R. 2003. *The Level Set Method and Dynamic Implicit Surfaces*. Springer-Verlag.

OTADUY, M. A., GERMANN, D., REDON, S., AND GROSS, M. 2007. Adaptive deformations with fast tight bounds. In *Proc. Symp. Comput. Animation*. 181–190.

PARKER, E. G. AND O'BRIEN, J. F. 2009. Real-time deformation and fracture in a game environment. In *Proc. Symp. Comput. Animation*. 156–166.

PAULY, M., KEISER, R., ADAMS, B., DUTRÉ, P., GROSS, M., AND GUIBAS, L. J. 2005. Meshless animation of fracturing solids. *ACM Trans. Graphics 24,* 3 (Aug.), 957–964.

RAYLEIGH, J. 1879. On the capillary phenomena of jets. *Proc. Royal Society London 29*, 71–97.

SCARDOVELLI, R. AND ZALESKI, S. 1999. Direct numerical simulation of free surface and interfacial flows. *Annual Review of Fluid Mechanics 31*, 567–603.

SIFAKIS, E., SHINAR, T., IRVING, G., AND FEDKIW, R. 2007. Hybrid simulation of deformable solids. In *Proc. Symp. Comput. Animation*. 81–90.

SIN, F., BARGTEIL, A. W., AND HODGINS, J. K. 2009. A point-based method for animating incompressible flow. In *Proc. Symp. Comput. Animation*. 247–255.

SMITH, J., WITKIN, A., AND BARAFF, D. 2001. Fast and controllable simulation of the shattering of brittle objects. *Computer Graphics Forum 20,* 2, 81–91.

STAM, J. 1999. Stable fluids. In *Proc. SIGGRAPH*. 121–128.

STEINEMANN, D., OTADUY, M. A., AND GROSS, M. 2006. Fast arbitrary splitting of deforming objects. In *Proc. Symp. Comput. Animation*. 63–72.

THÜREY, N., WOJTAN, C., GROSS, M., AND TURK, G. 2010. A multi-scale approach to mesh-based surface tension flows. *ACM Trans. Graphics 29,* 4, 48:1–48:10.

WANG, H., MUCHA, P. J., AND TURK, G. 2005. Water drops on surfaces. *ACM Trans. Graphics 24,* 3 (Aug.), 921–929.

WANG, H., O'BRIEN, J., AND RAMAMOORTHI, R. 2010. Multi-resolution isotropic strain limiting. *ACM Trans. Graphics 29,* 6, 156:1–156:10.

WICKE, M., RITCHIE, D., KLINGNER, B. M., BURKE, S., SHEWCHUK, J. R., AND O'BRIEN, J. F. 2010. Dynamic local remeshing for elastoplastic simulation. *ACM Trans. Graphics 29,* 4 (July), 49.1–49.11.

WOJTAN, C., THÜREY, N., GROSS, M., AND TURK, G. 2009. Deforming meshes that split and merge. *ACM Trans. Graphics 28,* 3, 76:1–76:10.

WOJTAN, C., THÜREY, N., GROSS, M., AND TURK, G. 2010. Physics-inspired topology changes for thin fluid features. *ACM Trans. Graphics 29,* 4 (July), 50:1–50:8.

WOJTAN, C. AND TURK, G. 2008. Fast viscoelastic behavior with thin features. *ACM Trans. Graphics 27,* 3, 47:1–47:8.

ZHANG, Y., WANG, H., WANG, S., TONG, Y., AND ZHOU, K. 2012. A deformable surface model for real-time water drop animation. *IEEE Trans. Vis. Comp. Graphics 18*, 1281–1289.

ZHU, Q.-H., CHEN, Y., AND KAUFMAN, A. 1998. Real-time biomechanically-based muscle volume deformation using FEM. *Computer Graphics Forum 17,* 3, 275–284.

APPENDIX

A. OSCILLATING DROPLETS

We simulated a single oscillating inviscid spherical droplet. We use a surface tension coefficient of $\gamma = 70.38 \times 10^{-3}$ N/m and density $\rho = 997$ kg/m^3 in the absence of gravity and viscosity. This test is one way to quantify the physical accuracy of our simulation, because we can compare it with known theory. It also allows us to measure the numerical viscosity introduced by resampling errors. We performed the simulation for three different time steps ($\Delta t = 0.1$ ms, $\Delta t = 0.05$ ms, $\Delta t = 0.01$ ms) and two different meshes (272–667 tetrahedra and 2,918–6,801 tetrahedra). The initial liquid droplet is an ellipsoid with radius 4 mm in the x-direction and a cross-sectional radius of 2 mm in the y-z plane. The radius of the sphere of equal volume at equilibrium is $R_0 = 2.5198$ mm.

For low-viscosity liquids (*e. g.* water, ethanol) and relatively large droplets ($R_0 \geq 50$ μm) in quiescent surroundings, the description derived by Rayleigh [1879] simplifies and the so-called irrotational approximation holds [Lamb 1932]. We consider only the fundamental mode. For a three-dimensional liquid droplet in vacuum, the period is $\Omega_2 = 2\pi((\rho R_0^3)/(8\gamma))^{1/2}$ and the decay time $\Gamma_2 = (\rho R_0^3)/(5\mu)$, which allows us to infer the magnitude of the numerical viscosity produced by interpolation errors.

Surface tension produces oscillations of the droplet with decaying amplitude. Fig. 17 charts the maximum elongation as a function of time. We fitted the oscillation maxima to obtain the oscillation period and decay time. The simulated oscillation period is about 5% longer than the theoretical value (Table II). The decay of the oscillations implies a viscosity of roughly $\mu = 10^{-5}$ Pa·s, which we take to be the baseline numerical viscosity inherent in the method. It is two orders of magnitude less than the viscosity of water, 10^{-3} Pa·s. As expected, a decrease of the time step or element size induces a decrease of the numerical viscosity (Fig. 18).

We conducted the same experiment with an Eulerian method based on Caboussat *et al.* [2010] and a time step of $\Delta t = 0.05$ ms. At 6×10^{-5} Pa·s, numerical viscosity is roughly six times higher in this setting, as compared to the Lagrangian simulation with 668–1,385 nodes, even though the mesh of the Eulerian simulation is finer (3,492 liquid nodes on average, element size of $h = 0.17$ mm). The calculated oscillating period is 35.3 ms; compare with the exact (analytical) oscillation period of 33.44 ms and the periods for our simulations in Table II. While the details may differ for different Eulerian simulators, the general finding should hold irrespective of implementation: because only a small fraction of variables are reinterpolated in each time step, numerical viscosity will be lower in Lagrangian methods than it is in Eulerian ones.

Received December 2011; revised September 2012; accepted November 2012

Δt (ms)	N_n (N_t)	R_0 (mm)	τ_{comp} (ms)
0.1	85–155 (272–607)	2.443 ± 0.001	34.6 ± 0.6
0.05	87–166 (272–660)	2.444 ± 0.002	34.9 ± 0.7
0.01	87–168 (272–667)	2.447 ± 0.003	35.0 ± 0.4
0.1	727–1298 (3281–6304)	2.5055 ± 0.0004	36.0 ± 0.4
0.05	668–1385 (2918–6798)	2.5057 ± 0.0003	36.3 ± 0.4
0.01	703–1392 (3107–6801)	2.5072 ± 0.0007	36.2 ± 0.7

Table II. : Time step Δt, number of vertices N_n, number of tetrahedra N_t, equilibrium radius R_0, and computed period of oscillation τ_{comp} for the simulation of an oscillating water droplet. The theoretical oscillation period for an equivalent radius of $R_0 = 2.5198$ mm is $\tau_2 = 33.44$ ms.

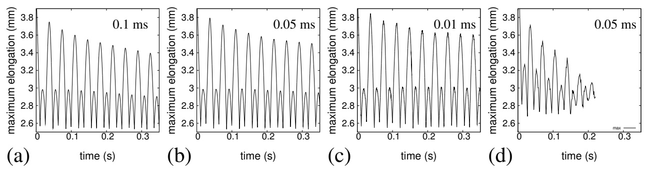

Fig. 17: Oscillations of a droplet. The mesh has 668–1392 nodes (average element size $h = 0.5$ mm). The time steps are (a) $\Delta t = 0.1$ ms, (b) $\Delta t = 0.05$ ms, (c) $\Delta t = 0.01$ ms. (d) Corresponding Eulerian simulation with time step $\Delta t = 0.05$ ms. The x-axis is time, and the y-axis graphs the maximum radial distance from the center of mass to the surface of the droplet. The small peaks represent the maximal length of the y-, and z-axes when the droplet is shortened in the x-direction.

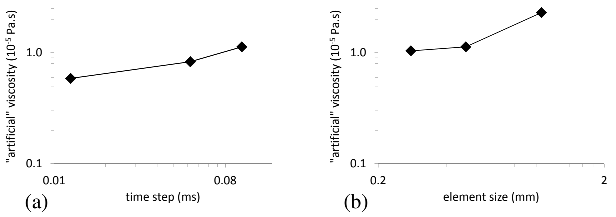

Fig. 18: (a) Numerical viscosity as a function of the time step for a mesh with 668–1,392 nodes (average element size $h = 0.5$ mm), computed from the oscillations' decay. Values are 0.56, 0.83, and 1.13 ($\times 10^{-5}$ Pa·s). (b) Numerical viscosity as a function of the element size for a time step of $\Delta t = 0.1$ ms. Values are 1.10, 1.13, and 2.3 ($\times 10^{-5}$ Pa·s).