Workshop on Virtual Reality Interaction and Physical Simulation VRIPHYS (2010)
J. Bender, K. Erleben, and M. Teschner (Editors)

Optimization-based Fluid Simulation on Unstructured Meshes

Marek Krzysztof Misztal[†1], Robert Bridson[‡2], Kenny Erleben[§3], Jakob Andreas Bærentzen[¶1] and François Anton[‖1]

[1] Department of Informatics and Mathematical Modelling, Technical University of Denmark
[2] Department of Computer Science, University of British Columbia
[3] Department of Computer Science, University of Copenhagen, Denmark

Abstract

We present a novel approach to fluid simulation, allowing us to take into account the surface energy in a precise manner. This new approach combines a novel, topology-adaptive approach to deformable interface tracking, called the deformable simplicial complexes method *(DSC) with an optimization-based, linear finite element method for solving the incompressible Euler equations. The deformable simplicial complexes track the surface of the fluid: the fluid-air interface is represented explicitly as a piecewise linear surface which is a subset of tetrahedralization of the space, such that the interface can be also represented implicitly as a set of faces separating tetrahedra marked as* inside *from the ones marked as* outside*. This representation introduces insignificant and controllable numerical diffusion, allows robust topological adaptivity and provides both a volumetric finite element mesh for solving the fluid dynamics equations as well as direct access to the interface geometry data, making inclusion of a new surface energy term feasible. Furthermore, using an unstructured mesh makes it straightforward to handle curved solid boundaries and gives us a possibility to explore several fluid-solid interaction scenarios.*

Categories and Subject Descriptors (according to ACM CCS): Computer Graphics [I.3.5]: Computational Geometry and Object Modeling —Physically based modeling Computer Graphics [I.3.7]: Three-Dimensional Graphics and Realism —Animation

1. Introduction

Since the mid-nineties of the previous century, fluid simulation has been extensively used in computer animated sequences of major motion pictures. Moreover, fluid simulation is important in many scientific applications, and simplistic fluid dynamics is even beginning to appear in real-time graphics applications.

Despite this rapid development, some problems remain with existing methods. Most existing methods are based on Eulerian simulations on fixed grids. These are prone to introducing gratuitous artificial viscosity in the simulations, volume loss, grid artifacts and, due to the lack of an explicit surface representation, surface tension is not easy to incorporate.

In the present paper, we propose a novel method for free-surface fluid simulation based on an irregular grid which dynamically adapts to the fluid volume. Our method is based on another recent technique for deformable interface tracking which we call *deformable simplicial complexes* (DSC, [Mis10]). The gist of the DSC method is that the tracked surface is represented as a subcomplex (e.g. triangle mesh) of a simplicial complex (tetrahedral grid) which covers the entire computational domain. Tetrahedra are labelled according to which side of the interface they reside in, and the surface (interface) itself is the set of faces shared by a pair of tetrahedra belonging to opposite sides.

In this paper, we focus on how fluid simulation can be implemented on top of the DSC method, using the tetrahedral grid of the DSC method also as the computational grid for

† mkm@imm.dtu.dk
‡ rbridson@cs.ubc.ca
§ kenny@diku.dk
¶ jab@imm.dtu.dk
‖ fa@imm.dtu.dk

DOI: 10.2312/PE/vriphys/vriphys10/011-020

the fluid simulation. Our new method has at least two important benefits compared to previous methods:

- Because the grid changes only little to adapt to the changes in the water volume, we have very little numerical diffusion.
- Since we have an explicit representation of the water-air interface, it is very easy to add a surface energy term.

2. Related Works on Fluid Solvers

Many works are based on regular grids; as a general reference to grid-based incompressible flow in graphics we refer to the book [Bri08]. Some of the foundations for grid-based works include an Eulerian approach to 3D fluid simulation [FM96] that demonstrated advantages over earlier work using particle systems, and 2D simulations and a semi-Lagrangian implicit time stepping method [Sta99]. Recent work addresses irregular boundaries on grids [BBB10].

Fluid animation on unstructured meshes, like our method, has been gaining popularity in the last five years. [FOK05] simulated gases on static tetrahedral meshes to model the interaction of fluids with irregularly shaped obstacles, based on a finite volume method discretization of the divergence operator with a projection method to enforce incompressibility. On their staggered mesh only normal components of velocities are stored at the face centers, making it easy to apply solid boundary conditions. The main difficulty is the non-trivial reconstruction of the full velocity field from the face normal components. In comparison we use a finite element approach on a moving and deforming tethrehedral mesh and we store the full velocity vector at the vertices.

Deforming unstructured tetrahedral meshes were introduced in [FOKG05]. Here a moving mesh is used where the deformation is limited to preserve mesh quality. Our approach to advection uses the same generalized semi-Lagrangian method from this work. In comparison to our work we emphasize the topological operations needed when deforming the mesh.

Remeshing of the entire computational domain in each simulation step was used in [KFCO06]. The authors addressed two-way coupling of fluids and rigid bodies. Heuristics were used to generate high resolution meshes in visually important regions; our mesh refinement and improvement are based on mesh quality only. (The coupling was later extended to deformable objects as well [CGFO06]; in our paper we do not address two-way coupling.)

Liquid simulation on unstructured tetrahedral meshes is presented in [CFL*07]. A semi-Lagrangian contouring method is used to extract the free surface and rebuild a tetrahedral mesh in every time step, and a body centered cubic lattice is used for the structure of the tetrahedral mesh. The liquid surface is embedded as a discrete submanifold in the tetrahedral mesh as in our case. However, rather than completely rebuilding a new mesh in each time step our approach is based on making local topological changes to remesh and improve mesh quality.

In [ETK*07] a new fluid simulation method is presented. A static staggered grid is used where velocities are stored at vertices and scalar fields at volume centers. The auhors apply a vorticity formulation of the Navier–Stokes equation whereas we use a momentum formulation. The authors employ discrete differential methods to guarantee a circulation-preserving flow, but do not handle liquids/free surfaces.

Another finite volume method is presented in [WBOL07]. Here full velocity vectors are stored at the face centers which add some problems to the pressure correction, necessitating an additional projection each time step.

While not strictly a fluid solver (focusing instead on elastoplastic materials) the work presented in [WT08] combines a highly detailed surface mesh with a non-conforming tetrahedral finite element simulator that makes frequent use of remeshing. In constrast we use a boundary-conforming tetrahedral mesh in our fluid solver.

In summary, past work is based on staggered meshes using face-centered velocity grid layouts. Most work on unstructured meshes deal with free surfaces using contouring and complete remeshing. Deforming meshes have been considered to control visual quality but in a deformation-limited manner; our approach follows the physical simulation and has no such limits. Further our work uses a finite element method for fluid simulation whereas previous work on fluid simulation on unstructured mehses use finite volume methods.

3. Deformable Interface Tracking

Traditionally, methods for deformable interface tracking fall into two categories: *explicit* (*Lagrangian*) and *implicit* (*Eulerian*). Traditional Lagrangian methods, such as active contours or snakes, use parametrisation of the interface and apply the deforming velocity field ($\mathbf{u}$) directly to the interface points ($\mathbf{x}$):

$$\frac{d\mathbf{x}}{dt} = \mathbf{u}(\mathbf{x}).$$

This approach leads to trouble once the topology of the interface changes. An efficient collision detection mechanism is needed to detect self-intersections of the interface, and once it happens, costly reparametrisation is needed, along with surgical cuts (as in [GGL*95], although in recent work by [BB09] this problem is mitigated by not allowing self-intersections). Those problems do not occur in Eulerian methods, such as the *level set method* (*LSM*, [OF02]). LSM represents a n-dimensional interface as the 0-level set of a $(n+1)$-dimensional function $f(x_1,\ldots,x_n,x_{n+1})$ (signed distance function is usually the choice), defined on the nodes of a regular grid. The evolution of the interface due to the velocity field $\mathbf{u}$ is then described by the following partial dif-

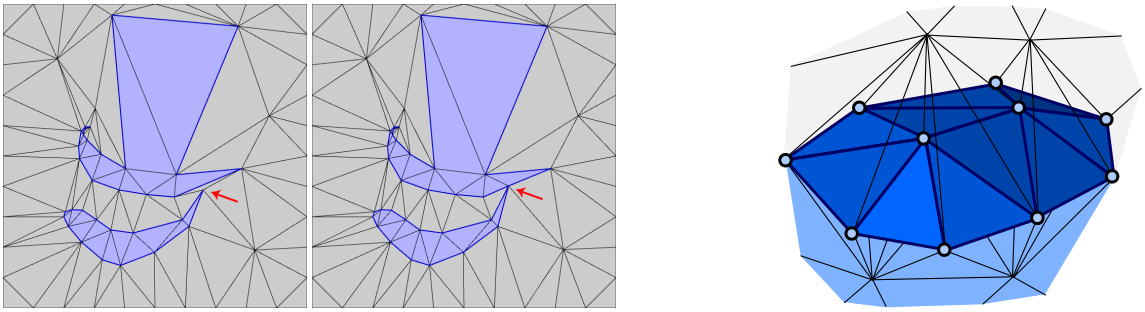

Figure 1: *Interface representation in deformable simplicial complexes (2D on the left, 3D on the right). Exterior triangles (tetrahedra) are light gray, interior – blue. Simplices belonging to the interface (edges and vertices in 2D; faces, edges and vertices in 3D) are highlighted in dark blue. On the left, the red arrow indicates where topology changes take place. Note also the difference in scale between the largest and the smallest triangles.*

ferential equation, also known as that *level set equation*:

$$\frac{\partial f}{\partial t} + \mathbf{u} \cdot \nabla f = 0.$$

This approach provides trivial and robust topological adaptivity. However, the LSM also exhibits several drawbacks: it is bound to a certain scale, it suffers from significant numerical diffusion for features near the sampling rate irrespective of discretization, it does not allow explicit interface representation and it relies on calculations in one dimension greater than the interface itself.

The work most directly related to ours is the method presented in [PB07]. The authors proposed a method which is based on a triangle mesh representation of the interface, but once the vertices have been moved, a restricted Delaunay tetrahedralization of the interface is performed. A test is performed on each of the new tetrahedra in order to label them as interior or exterior. If a vertex is found to be shared only by identically labeled tetrahedra, it is removed. This method shares a number of advantages with our method. In particular, it can be extended to multi-phase simulations, and it suffers only little from numerical diffusion, but there is no detection of what happens between time steps. Arguably a small object could pass through a thin wall if the time step was not properly tuned, and the precise points where interface collisions occur are not detected. Lastly, it would be difficult to extend their method to do topology control which is simple with our approach.

In the *deformable simplicial complexes* (DSC), the interface is represented explicitly as a set of faces of simplices belonging to a simplicial complex one dimension higher. These simplices belong either to the object or the exterior. Simplices never straddle object boundaries. Thus, in 2D, the computational domain is divided into triangles, and the deforming interface is the set of line segments which divide interior triangles from exterior triangles. Similarly, in 3D, the interface is the set of triangles dividing interior tetrahedra from exterior tetrahedra. Both the 2D and 3D case are illustrated in Figure 1.

The interface deformation is performed by moving the vertices, and this means that the method preserves the advantages of the Lagrangian methods: It suffers from little numerical diffusion, and there is an explicit representation of the interface which, furthermore, does not change *gratuitously* between time steps. Moreover, the simplicial complex does not have to be regular meaning that we can allow details of significantly different scale in the same grid (c.f. Figure 1 left).

On the other hand, our approach also shares what we perceive as the biggest advantage of the Eulerian methods. Whenever the interface moves, the triangulation is updated to accommodate the change. If two different interface components collide, this change causes them to merge. Thus, topology is allowed to change transparently to the user—although with our method it would also be possible to disallow topological changes.

The DSC in described in detail, together with some of its other applications in [Mis10].

4. Fluid Simulation

In DSC we attempt to keep the quality of the volume mesh high for the finite element computations, so it is natural to use it directly in the incompressible Euler equations solver. The fluid mass can be represented as the set of *interior* simplices, which can be treated as first order, *conforming* linear elements (meaning that velocity values in a vertex agree for each element sharing that vertex). These are subject to *locking* in the incompressible limit [IGLF06, EB08, ZTT05]. Locking means inability of a given finite element space to offer good approximate solutions, due to the fact that volume constraint on each tetrahedron may leave us with a solution space of very low dimension, or even an overconstrained problem (depending on the boundary conditions).

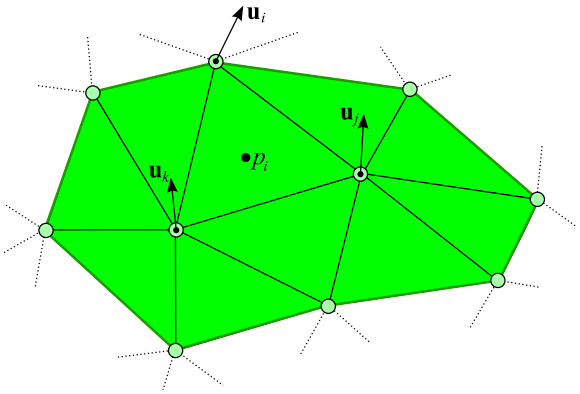

Figure 2: *DSC setup for fluid simulation (observe this is analogical to a staggered mesh). The velocity values* $\mathbf{u}$ *are sampled at the vertices of the DSC mesh and the pressure values* p *are sampled per element (triangle in 2D, tetrahedron in 3D).*

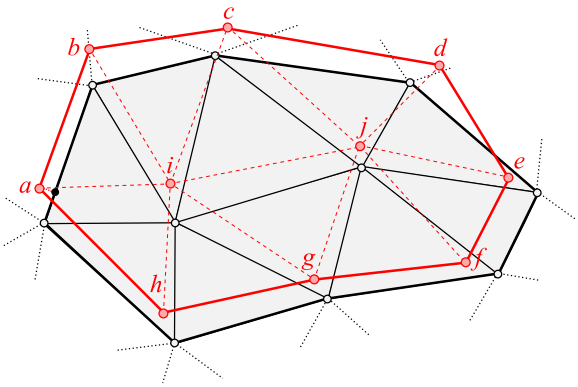

Figure 3: *Advection of the velocity field. If the new vertex position is inside the old fluid volume (vertices e, f, g, h, i and j), we find its new velocity as the linear interpolation of old vertex velocities at this point. In order to find new velocity values at the vertices a, b, c and d, we find their projections onto the interface and sample the velocity there.*

This can manifest itself by, for example: only allowing globally affine divergence-free deformations of the fluid volume. However, locking can be avoided by using pressure stabilization [FP02], as presented in Section 4.4, in exchange for slighly violating the incompressibility constraint. Meanwhile, the simplicity of the linear elements facilitates easy implementation of advection and optimization-based implicit surface tension.

In such a setup, presented in the Figure 2, fluid velocity values are sampled in the vertices (both interface and interior ones) and pressure values are sampled in the centers of volume elements (triangles in the 2D case and tetrahedra in the 3D case). The velocity field is then defined as:

$$\mathbf{u}(\mathbf{x}) = \sum_{i=1}^{N_V} \mathbf{u}_i \varphi_i(\mathbf{x}), \tag{1}$$

where N_V is the number of vertices in the mesh and φ_i is the linear interpolant (*hat* function defined on the star of vertex v_i).

Our method loosely follows the steps of a *fractional step method*, known from the regular-grid based fluid solvers [Bri08].

4.1. Advection

In a Lagrangian setup (such as DSC) advection of the mesh vertices is trivial. Having vertex positions $\{\mathbf{v}_i^t\}_{i=1}^{N_V}$ and velocities $\{\mathbf{u}_i^t\}_{i=1}^{N_V}$ at the time-step t, one can compute the positions at the next time-step $t+\Delta t$ using simple forward Euler integration:

$$\mathbf{v}_i^{t+\Delta t} = \mathbf{v}_i^t + \mathbf{u}_i^t \Delta t.$$

One could also try to use a simple, Lagrangian approach in order to advect the velocity field. However, since we additionally perform smoothing on the mesh vertices, we have to use a slightly more complex, semi-Lagrangian method.

In order to advect the velocity field (or any other quantity sampled at the vertices) we interpolate or extrapolate the values from the previous time-step at the new vertex positions (see Figure 3). If the point $\mathbf{v}_i^{t+\Delta t}$ lies inside the fluid volume at the time-step t, we localize the element σ inside which it lies and compute the new velocity value as the linear combination of the velocities in the vertices of σ with barycentric coordinates of $\mathbf{v}_i^{t+\Delta t}$ as linear coefficients:

$$\mathbf{u}_i^{t+\Delta t} = \mathbf{u}^t(\mathbf{v}_i^{t+\Delta t}).$$

If $\mathbf{v}_i^{t+\Delta t}$ lies outside the fluid volume at the time-step t, we find its projection $\bar{\mathbf{v}}_i^{t+\Delta t}$ onto the interface and sample the velocity at this point:

$$\mathbf{u}_i^{t+\Delta t} = \mathbf{u}^t(\bar{\mathbf{v}}_i^{t+\Delta t}). \tag{2}$$

4.2. Enforcing Incompressibility

Incompressibility of the fluid yields that the divergence of the velocity field vanishes everywhere:

$$\nabla \cdot \mathbf{u} = 0.$$

In our setup (see equation 1):

$$\nabla \cdot \mathbf{u} = \sum_{i=1}^{N_V} \mathbf{u}_i \cdot \nabla \varphi_i.$$

The gradient $\nabla\varphi_i$ is constant over every element (triangle in 2D, tetrahedron in 3D). Let us denote it by:

$$\nabla\varphi_i \equiv \mathbf{d}_{j,i} \text{ over element } \sigma_j.$$

The incompressibility condition is then fulfilled iff:

$$\sum_{i=1}^{N_V} \mathbf{d}_{j,i} \cdot \mathbf{u}_i = 0 \quad \text{for } j = 1,\ldots,N_T,$$

where N_T is the number of elements (triangles in 2D, tetrahedra in 3D) in the mesh. The last equation can be written in matrix form:

$$\mathbf{D}\mathbf{u} = \mathbf{0},$$

where $\mathbf{u}$ is a size $d \cdot N_V$ (where the dimension $d = 2$ or 3) vector containing the coordinates of the vertex velocities and $\mathbf{D}$ is an $N_T \times d \cdot N_V$ sparse matrix.

To enforce incompressibility of the velocity field $\{\tilde{\mathbf{u}}_i\}_{i=1}^{N_V}$ after advection, we introduce a pressure field $\{p_i\}_{i=1}^{N_T}$, such that:

$$\mathbf{u} = \tilde{\mathbf{u}} - \mathbf{M}^{-1}\mathbf{D}^{\mathsf{T}}\mathbf{p}, \tag{3}$$

where $\mathbf{u}$ is divergence free, $\mathbf{p}$ is a size N_T vector containing the pressure values in each face and $\mathbf{M}$ is a size $d \cdot N_V \times d \cdot N_V$ diagonal mass matrix, with diagonal values:

$$\mathbf{M}_{d\cdot i-d+1,d\cdot i-d+1} = \ldots = \mathbf{M}_{d\cdot i,d\cdot i} = m_i,$$

for $i = 1,\ldots,N_V$, where:

$$m_i = \frac{1}{3}\rho \sum_{\sigma \in \mathrm{star}(v_i)} \mathrm{volume}(\sigma),$$

with ρ the fluid density. The incompressibility condition yields:

$$\mathbf{D}\mathbf{u} = \mathbf{D}\tilde{\mathbf{u}} - \mathbf{D}\mathbf{M}^{-1}\mathbf{D}^{\mathsf{T}}\mathbf{p} = \mathbf{0}.$$

Therefore:

$$\begin{aligned} \mathbf{D}\mathbf{M}^{-1}\mathbf{D}^{\mathsf{T}}\mathbf{p} &= \mathbf{D}\tilde{\mathbf{u}}, \\ \mathbf{A}\mathbf{p} &= \mathbf{b}, \end{aligned}$$

where $\mathbf{A} = \mathbf{D}\mathbf{M}^{-1}\mathbf{D}^{\mathsf{T}}$ and $\mathbf{b} = \mathbf{D}\tilde{\mathbf{u}}$. By solving this linear system, we can compute the pressure field and then, by using equation 3, the divergence-free velocity field $\mathbf{u}$.

Solid Boundaries Solid boundaries put extra constraints on vertex velocity values. If the vertex v_i is in contact with the solid (see Figure 4), we force the projection of the vertex's velocity onto the solid normal at the point of collision to match the projection of the solids own velocity onto its normal:

$$\langle \mathbf{u}_i, \mathbf{n}(\mathbf{p}_i) \rangle = \langle \mathbf{u}_{solid}, \mathbf{n}(\mathbf{p}_i) \rangle, \tag{4}$$

while the tangent coordinates of v_i remain unconstrained. In order to compute the new divergence-free velocity field $\{\mathbf{u}\}_{i=0}^{N_V}$ we first need to express the global velocity vector $\tilde{\mathbf{u}}$ and the matrix $\mathbf{D}$ in new coordinates ($\mathbf{n}$ and $\mathbf{t}$ in 2D or $\mathbf{n}$, $\mathbf{t}_1$ and $\mathbf{t}_2$ in 3D, whenever a vertex is in contact with the solid). Then we permute the rows of $\tilde{\mathbf{u}}$ and the columns of $\mathbf{D}$, so that:

$$\tilde{\mathbf{u}} = \begin{bmatrix} \tilde{\mathbf{u}}_f \\ \hline \tilde{\mathbf{u}}_c \end{bmatrix}, \qquad \mathbf{D} = \left[\begin{array}{c|c} \mathbf{D}_f & \mathbf{D}_c \end{array}\right] \tag{5}$$

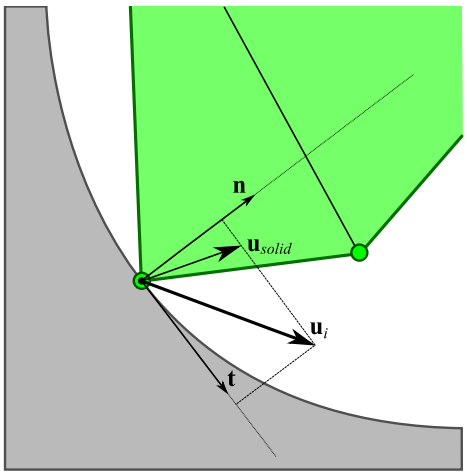

Figure 4: *Collision of the fluid with the solid boundary.* $\mathbf{n}$ *is the normal and* $\mathbf{t}$ *is the tangent vector to the solid boundary at the point of collision.* $\mathbf{u}_{solid}$ *is the velocity of the solid boundary and* $\mathbf{u}_i$ *is constrained in the normal direction:* $\langle \mathbf{u}_i, \mathbf{n} \rangle = \langle \mathbf{u}_{solid}, \mathbf{n} \rangle$.

where $\tilde{\mathbf{u}}_f$ contains the free and $\tilde{\mathbf{u}}_c$ the constrained coordinates of $\tilde{\mathbf{u}}$. The incompressibility condition 3 can be then written as:

$$\begin{aligned} \mathbf{D}\mathbf{u} &= \mathbf{0}, \\ \left[\begin{array}{c|c} \mathbf{D}_f & \mathbf{D}_c \end{array}\right] \begin{bmatrix} \mathbf{u}_f \\ \hline \mathbf{u}_c \end{bmatrix} &= \mathbf{0}, \\ \mathbf{D}_f\mathbf{u}_f + \mathbf{D}_c\mathbf{u}_c &= \mathbf{0}, \end{aligned}$$

but since $\mathbf{u}_c = \tilde{\mathbf{u}}_c$ (as they are forced to match the velocity of the solid projected onto the solid normal), we obtain:

$$\mathbf{D}_f\mathbf{u}_f = -\mathbf{D}_c\tilde{\mathbf{u}}_c. \tag{6}$$

In order to ensure incompressibility of the velocity field $\mathbf{u}$, we again introduce a pressure field $\mathbf{p}$:

$$\mathbf{u}_f = \tilde{\mathbf{u}}_f - \mathbf{M}_f^{-1}\mathbf{D}_f^{\mathsf{T}}\mathbf{p}.$$

Multiplying both sides by $\mathbf{D}_f$ and using eq. 6 gives:

$$\begin{aligned} \mathbf{D}_f\mathbf{u}_f &= \mathbf{D}_f\tilde{\mathbf{u}}_f - \mathbf{D}_f\mathbf{M}_f^{-1}\mathbf{D}_f^{\mathsf{T}}\mathbf{p}, \\ -\mathbf{D}_c\tilde{\mathbf{u}}_c &= \mathbf{D}_f\tilde{\mathbf{u}}_f - \mathbf{D}_f\mathbf{M}_f^{-1}\mathbf{D}_f^{\mathsf{T}}\mathbf{p}, \end{aligned}$$

which can be used to evaluate $\mathbf{p}$ by solving a linear system:

$$\begin{aligned} \mathbf{D}_f\mathbf{M}_f^{-1}\mathbf{D}_f^{\mathsf{T}}\mathbf{p} &= \mathbf{D}_f\tilde{\mathbf{u}}_f + \mathbf{D}_c\tilde{\mathbf{u}}_c, \\ \mathbf{A}_f\mathbf{p} &= \mathbf{D}_f\tilde{\mathbf{u}}_f + \mathbf{D}_c\tilde{\mathbf{u}}_c. \end{aligned}$$

Gravity Including gravity in our fluid dynamics solver is trivial and can be performed by adding $\mathbf{g}\Delta t$ to the velocity field in every time step, where $\mathbf{g}$ is the gravitational acceleration.

4.3. Optimization Based Approach

Surface Tension In order to make our fluid simulation more plausible we include *surface tension*. Surface tension is derived from *surface energy* U_γ defined as:

$$U_\gamma = \gamma A,$$

where γ is the surface energy density (material constant) and A is the free surface area. Surface tension forces alone yield a highly divergent velocity field and our experiments have shown that integrating them before enforcing incompressibility step can give very poor results. Instead, we fully couple them with incompressibility by solving the following optimization problem:

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}(\mathbf{u}-\tilde{\mathbf{u}})^\mathsf{T}\mathbf{M}(\mathbf{u}-\tilde{\mathbf{u}}) + U_\gamma(\mathbf{x}+\mathbf{u}\Delta t), \\ \text{subject to} & \mathbf{D}\mathbf{u} = \mathbf{0}, \end{array} \tag{7}$$

which is consistent as the first-order KKT conditions for the optimality of its solution is the backward Euler step:

$$\mathbf{u} = \tilde{\mathbf{u}} - \Delta t \mathbf{M}^{-1}\nabla U_\gamma(\mathbf{u}) - \mathbf{M}^{-1}\mathbf{D}^\mathsf{T}\mathbf{p},$$

where the pressure values $\mathbf{p}$ play the role of Lagrange multipliers (see that for $\gamma = 0$ this is identical with eq. 3). In our work so far we only use first-order approximation of the surface energy function:

$$\begin{aligned} U_\gamma(\mathbf{x}+\mathbf{u}\Delta t) &= \gamma A(\mathbf{x}+\mathbf{u}\Delta t) \approx \\ &\approx \gamma A(\mathbf{x}) + \gamma\Delta t\mathbf{k}^\mathsf{T}\mathbf{u}, \end{aligned}$$

where $\mathbf{k}^\mathsf{T}$ is the area gradient ∇A. Substituting this into optimization problem 7 and dropping constant terms leads to a simple quadratic programming problem with linear equality constraints:

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\mathbf{u}^\mathsf{T}\mathbf{M}\mathbf{u} + (-\mathbf{M}\tilde{\mathbf{u}} + \gamma\Delta t\mathbf{k})^\mathsf{T}\mathbf{u}, \\ \text{subject to} & \mathbf{D}\mathbf{u} = \mathbf{0}. \end{array}$$

In this fashion we avoid having to explicitly estimate surface curvature, automatically conserve linear and angular momentum by virtue of translation and rotation-independence of the objective function and naturally capture minimum-surface-area equilibrium. We do, however, realize that this setup does not allow for non-linear surface phenomena in our simulations.

Solid Boundaries Incorporating solid boundaries into the new setting is relatively straightforward. However, one has to take into account the fact that the surface energy density for the fluid-air surface γ is usually different from the surface energy density for the fluid-solid surface $\alpha_1\gamma$ and from the surface energy density for the air-solid surface $\alpha_2\gamma$. Hence, we multiply the area of the solid-liquid contact surface by $\alpha_1 - \alpha_2$. Using the notation from the previous section, the zero-divergence constraint is decribed by the eq. 6. Also, the surface energy has to be expressed in the new variables:

$$\begin{aligned} U_\gamma(\mathbf{x}+\mathbf{u}\Delta t) &= \gamma A(\mathbf{x}+\mathbf{u}\Delta t) \approx \\ &\approx \gamma A(\mathbf{x}) + \gamma\Delta t\mathbf{k}_f^\mathsf{T}\mathbf{u}_f + \gamma\Delta t\mathbf{k}_c^\mathsf{T}\mathbf{u}_c, \end{aligned}$$

where $\mathbf{k}_f$ is a vector containing those coordinates of the area gradient ∇A, which correspond to the free coordinates of $\mathbf{u}$ (put together in a vector $\mathbf{u}_f$) and $\mathbf{k}_c$ contains those cordinates of ∇A, which correspond to the constrained coordinates of $\mathbf{u}$ (put together in a vector $\mathbf{u}_c$). Finally, after dropping constant terms and terms depending only on $\mathbf{u}_c$, we can state our optimization problem in the following form:

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\mathbf{u}_f^\mathsf{T}\mathbf{M}_f\mathbf{u}_f + (-\mathbf{M}_f\tilde{\mathbf{u}}_f + \gamma\Delta t\mathbf{k}_f)^\mathsf{T}\mathbf{u}_f, \\ \text{subject to} & \mathbf{D}_f\mathbf{u}_f = -\mathbf{D}_c\tilde{\mathbf{u}}_c. \end{array} \tag{8}$$

Solution For sake of simplicity, let us rewrite the optimization problem 8 as:

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\mathbf{u}_f^\mathsf{T}\mathbf{M}_f\mathbf{u}_f - \mathbf{b}\mathbf{u}_f, \\ \text{subject to} & \mathbf{D}_f\mathbf{u}_f = \mathbf{c}, \end{array}$$

where $\mathbf{b} = \mathbf{M}_f\tilde{\mathbf{u}}_f - \gamma\Delta t\mathbf{k}_f$ and $\mathbf{c} = -\mathbf{D}_c\tilde{\mathbf{u}}_c$. The solution of this optimization problem can be found by solving the following linear equation

$$\begin{bmatrix} \mathbf{M}_f & \mathbf{D}_f^\mathsf{T} \\ \mathbf{D}_f & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{u}_f \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} \mathbf{b} \\ \mathbf{c} \end{bmatrix}. \tag{9}$$

We are doing this by applying Schur complement method for solving linear systems with block matrices [Zha05]. It produces the following linear equation:

$$-\mathbf{D}_f\mathbf{M}_f^{-1}\mathbf{D}_f^\mathsf{T}\mathbf{p} = \mathbf{c} - \mathbf{D}_f\mathbf{M}_f^{-1}\mathbf{b}.$$

Then, having found $\mathbf{p}$:

$$\mathbf{u}_f = \mathbf{M}_f^{-1}\mathbf{b} - \mathbf{M}_f^{-1}\mathbf{D}_f^\mathsf{T}\mathbf{p}.$$

This way, we are only required to solve a linear equation with a size $N_T \times N_T$ matrix (instead of size $d \cdot N_V + N_T \times d \cdot N_V + N_T$ original problem), as computing the inverse of the diagonal matrix $\mathbf{M}_f$ is trivial.

4.4. Pressure Stabilization

As we previously mentioned, presented finite element setup is subject to *locking*. This problem can be solved by adding a stabilization term $\mathbf{S}$ (size $N_T \times N_T$) to the linear equation 9:

$$\begin{bmatrix} \mathbf{M}_f & \mathbf{D}_f^\mathsf{T} \\ \mathbf{D}_f & -\mathbf{S} \end{bmatrix} \begin{bmatrix} \mathbf{u}_f \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} \mathbf{b} \\ \mathbf{c} \end{bmatrix}. \tag{10}$$

such that:

$$\mathbf{S}_{ij} = \begin{cases} -\delta \cdot a_{ij} & \text{if } i \neq j \\ \delta \cdot \sum_{k \neq i} a_{ik} & \text{if } i = j \end{cases} \tag{11}$$

where δ is a positive stabilization parameter and a_{ij} is the area of the face shared by tetrahedra i and j, if they do have a common face, or otherwise 0. Stabilization term of this form acts like Laplacian smoothing of the pressure field in exchange for slightly violating the incompressibility constraint.

In order to solve equation 10, we again apply Schur complement method and solve the following equation:

$$-(\mathbf{S}+\mathbf{D}_f\mathbf{M}_f^{-1}\mathbf{D}_f^{\mathsf{T}})\mathbf{p} = \mathbf{c}-\mathbf{D}_f\mathbf{M}_f^{-1}\mathbf{b}.$$

Then, having found $\mathbf{p}$:

$$\mathbf{u}_f = \mathbf{M}_f^{-1}\mathbf{b}-\mathbf{M}_f^{-1}\mathbf{D}_f^{\mathsf{T}}\mathbf{p}.$$

Even though the velocity field $\mathbf{u}$ computed this way is not divergence-free in each individual tetrahedron, it is still *globally* volume preserving for a stabilization term $\mathbf{S}$ defined as above. It is easy to notice that the form of stabilization term 11 yields that the sum of all coordinates of the vector $\mathbf{Sp}$ equals 0. Since:

$$\begin{aligned}\mathbf{Sp} &= \mathbf{D}_f\mathbf{u}_f-\mathbf{c}\\ &= \mathbf{D}_f\mathbf{u}_f+\mathbf{D}_c\tilde{\mathbf{u}}_c\\ &= \mathbf{D}_f\mathbf{u}_f+\mathbf{D}_c\mathbf{u}_c = \mathbf{Du},\end{aligned}$$

that means that the integral of the divergence of the velocity field over the fluid volume equals 0, hence it preserves global volume.

4.5. Volume Loss Compensation

If the volume loss due to the truncation errors is visually noticeable, one can compensate for that by adding a constant term:

$$\tau\cdot\frac{V_0-V_c}{\Delta t\cdot V_c}$$

to the desired divergence $\mathbf{c}$, where V_0 is the original volume of the fluid, V_c is the current volume and τ is a relaxation parameter ($\tau = 0.5$ being a good choice).

5. Tests and Results

Stationary Volume of Fluid We first tested our solver on a regular sphere model obtained by subdiving an icosahedron 4 times using $\sqrt{3}$-subdivision scheme [Kob00] and reprojecting the vertices onto a sphere. The initial velocity of all vertices are set to zero and there is no gravity – the only forces in this setup are due to surface tension and incompressibility.

After 10000 iterations the changes in volume and surface area are below floating point truncation error and there is no visible displacement of the mesh vertices, as expected for this symmetric situation.

Droplets Colliding Our next test involves two water droplets in 0-gravity conditions. In the first experiment (see Figure 5) they collide head-on, and in the second (see Figure 6) they collide obliquely. In the first case droplets merge and the resulting volume begins to oscillate between flattened and elongated shape, according to the energy conservation principle. In the second case droplets first merge, but as two big fractions of volume keep moving in original directions they detach, leaving a trace of small droplets, which soon start to oscillating around the spherical equilibrium.

Fluid-Solid Interaction In the first test, we examined the behavior of a droplet of fluid put on a flat surface, subject to gravitational force. Let γ be the surface energy density for the fluid-air interface, $\alpha_1\gamma$ – for the fluid-solid interface and $\alpha_2\gamma$ – for the air-solid interface. Then the contact angle θ between the fluid-air surface and the solid surface equals:

$$\cos\theta = -(\alpha_1-\alpha_2).$$

A concave *meniscus* has contact angle less than 90° (e.g. water on glass) and a convex meniscus has contact angle greater than 90° (e.g. water on paraffin wax or mercury on glass). We ran tests for $\alpha_1-\alpha_2 = 0.75$ and $\alpha_1-\alpha_2 = -0.75$. The results, presented in Figure 7 demonstrate physical soundness of our method.

We also tested our fluid simulation in scenarios involving curved solid boundaries. The results, presented in Figures 8 and 9, demonstrate that the curved boundaries are handled correctly in our setup.

Performance In all of our experiments the number of tetrahedra was on the order of 10000. Simulation time ranged from about 10 to 30 iterations per minute (on 64-bit Intel® Xeon® CPU W5590 @ 3.33 GHz, 6 GB RAM) – not including rendering which was done in a subsequent step.

6. Conclusions and Future Work

In this paper, we have demonstrated the feasibility of fluid simulation in a framework where the computational grid evolves over time, maintaining the fluid interface as a subcomplex of the tetrahedral grid. This is in contrast to the few examples of previous work which used unstructured grids. In these methods, the computational grid is either fixed or, essentially, rebuilt every time step.

Because of this, we have an explicit fluid surface representation in the form of a triangle mesh which is also not rebuilt every time step since it is a subcomplex of the tetrahedral grid. From this we derive one of the big advantages of the method, namely that we can easily formulate surface energy in terms of the surface geometry.

Since, arguably, this method is qualitatively different from previous unstructured mesh based methods for fluid simulation, it is unsurprising that there is room for future improvement.

It is clear from our screenshots and animations that the fluid surface is quite rough in some cases. This is due to the lack of a viscosity term which should be straightforward to add and make it faster to reach the equilibrium state. Note also that in grid based methods, unintentional viscosity is quite common due to numerical diffusion. In fact, one of the strengths of our method is that there is very little numerical

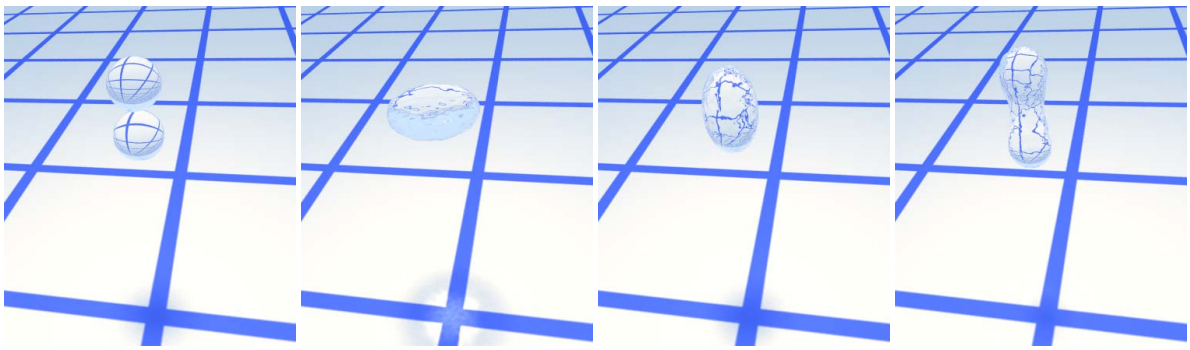

Figure 5: *Head-on collision of two water droplets in 0-gravity conditions.*

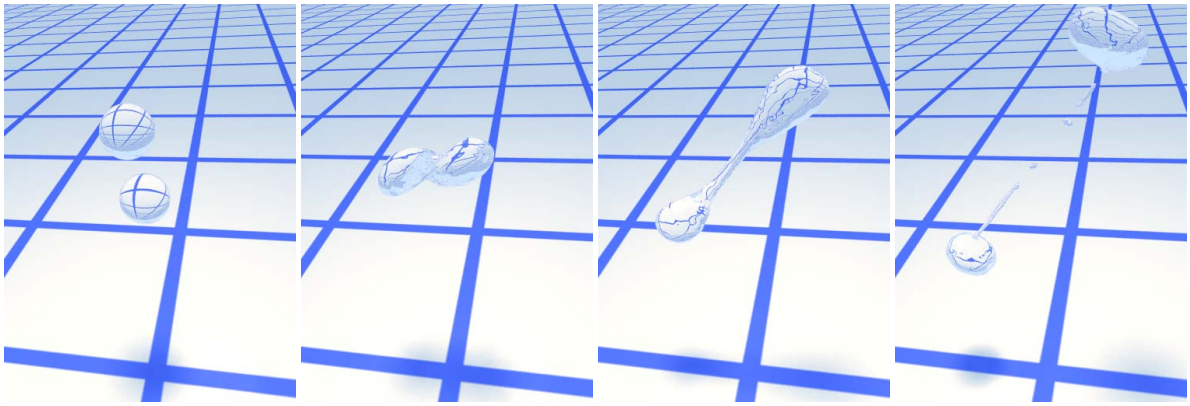

Figure 6: *Oblique collision of two water droplets in 0-gravity conditions.*

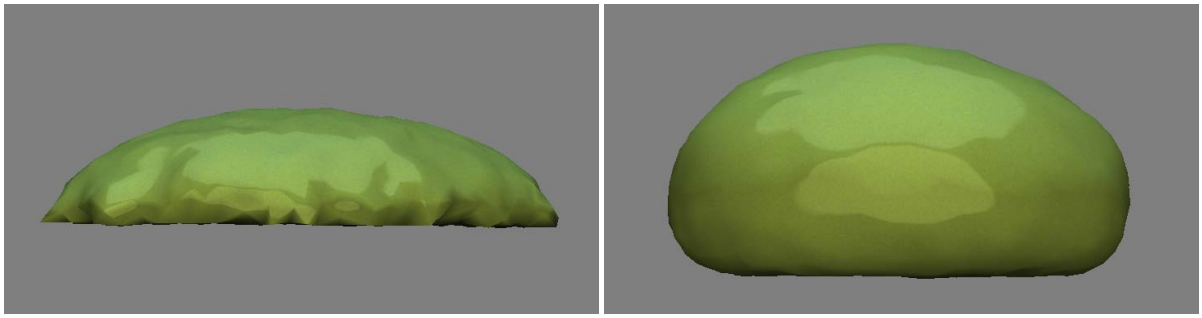

Figure 7: *Shape aquired by a droplet of liquid put on a flat surface. On the left-hand side:* $\alpha_1 - \alpha_2 = -0.75$*, and indeed the contact angle* $\theta < 90^\circ$*. On the right-hand side:* $\alpha_1 - \alpha_2 = 0.75$*, and indeed the contact angle* $\theta > 90^\circ$*. Note that the pond shape is more irregular when* $\theta < 90^\circ$*. This is the case also in the physical world, e.g.: ponds of water on the glass usually acquire quite irregular shapes, while ponds of mercury on the glass are usually round. This is due to the fact, that increasing the contact surface between water and glass decreases the total energy of the system, unlike in the latter case.*

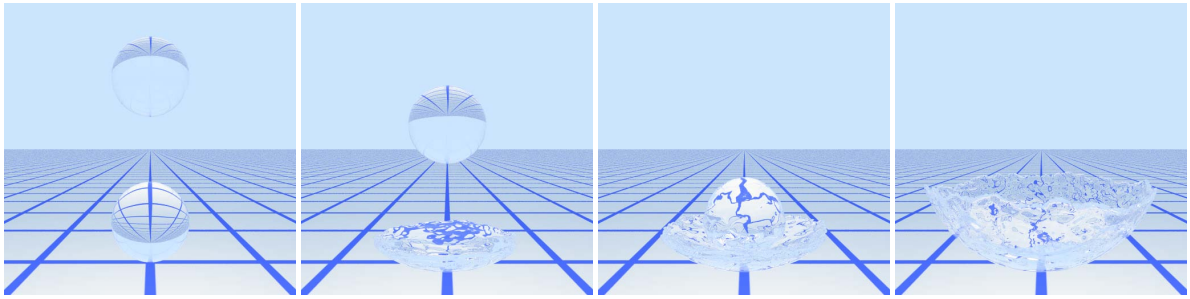

Figure 8: *Two drops of water splashing inside a solid sphere.*

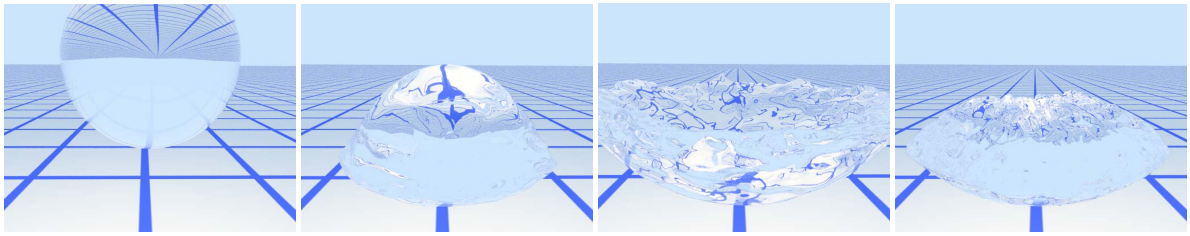

Figure 9: *Large drop of water splashing inside a solid sphere.* $\alpha_1 - \alpha_2 = 0$ *(corresponding to contact angle* $\theta = 90^\circ$*, characteristic for e.g. water on silver), surface energy density exagerrated.*

diffusion since we only change the mesh when parts of the surface collide (to change topology of the fluid volume) or when we need to remove poor quality tetrahedra.

One of the direct and straight-forward short term goals is investigating the influence of using second-order surface energy approximation, which can easily be included in the existing framework. We are also planning to try using Sequential Quadratic Programming in order to investigate the influence of higher-order terms (although we believe this might require using more elaborate mesh refinement schemes).

The other way of improving our method could be by using more sophisticated linear elements.

Furthermore, our method can be extended by adding viscosity, allowing compressible fluids and multiple phases (supported naturally by the DSC). Ultimately, we would like to include solid (rigid, ellastic and deformable) objects in an unified physis simulation setup.

Acknowledgements

We would like to thank Jeppe Revall Frisvad (DTU Informatics) for providing us with the water rendering software.

References

[BB09] BROCHU T., BRIDSON R.: Robust topological operations for dynamic explicit surfaces. *SIAM Journal on Scientific Computing 31*, 4 (2009), 2472–2493. 2

[BBB10] BROCHU T., BATTY C., BRIDSON R.: Matching fluid simulation elements to surface geometry and topology. In *ACM SIGGRAPH 2010 papers* (2010), ACM, p. XX. 2

[Bri08] BRIDSON R.: *Fluid Simulation*. A. K. Peters, Ltd., Natick, MA, USA, 2008. 2, 4

[CFL*07] CHENTANEZ N., FELDMAN B. E., LABELLE F., O'BRIEN J. F., SHEWCHUK J. R.: Liquid simulation on lattice-based tetrahedral meshes. In *SCA '07: Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation* (Aire-la-Ville, Switzerland, Switzerland, 2007), Eurographics Association, pp. 219–228. 2

[CGFO06] CHENTANEZ N., GOKTEKIN T. G., FELDMAN B. E., O'BRIEN J. F.: Simultaneous coupling of fluids and deformable bodies. In *SCA '06: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation* (Aire-la-Ville, Switzerland, Switzerland, 2006), Eurographics Association, pp. 83–89. 2

[EB08] ENGLISH E., BRIDSON R.: Animating developable surfaces using nonconforming elements. In *ACM SIGGRAPH 2008 papers* (2008), ACM, p. 66. 3

[ETK*07] ELCOTT S., TONG Y., KANSO E., SCHRÖDER P., DESBRUN M.: Stable, circulation-preserving, simplicial fluids. *ACM Trans. Graph. 26*, 1 (2007), 4. 2

[FM96] FOSTER N., METAXAS D.: Realistic animation of liquids. *Graph. Models Image Process. 58*, 5 (1996), 471–483. 2

[FOK05] FELDMAN B. E., O'BRIEN J. F., KLINGNER B. M.: Animating gases with hybrid meshes. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers* (New York, NY, USA, 2005), ACM, pp. 904–909. 2

[FOKG05] FELDMAN B. E., O'BRIEN J. F., KLINGNER B. M., GOKTEKIN T. G.: Fluids in deforming meshes. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics sym-*

posium on Computer animation (New York, NY, USA, 2005), ACM, pp. 255–259. 2

[FP02] FERZIGER J. H., PERIC M.: *Computational Methods for Fluid Dynamics*, 3rd ed. Springer, 2002. 4

[GGL*95] GLIMM J., GROVE J. W., LI X. L., SHYUE K.-M., ZENG Y., ZHANG Q.: Three dimensional front tracking. *SIAM J. Sci. Comp 19* (1995), 703–727. 2

[IGLF06] IRVING G., GUENDELMAN E., LOSASSO F., FEDKIW R.: Efficient simulation of large bodies of water by coupling two and three dimensional techniques. *ACM Transactions on Graphics (TOG) 25*, 3 (2006), 811. 3

[KFCO06] KLINGNER B. M., FELDMAN B. E., CHENTANEZ N., O'BRIEN J. F.: Fluid animation with dynamic meshes. *ACM Trans. Graph. 25*, 3 (2006), 820–825. 2

[Kob00] KOBBELT L.: $\sqrt{3}$-subdivision. In *Proceedings of the 27th annual conference on computer graphics and interactive techniques* (2000). 7

[Mis10] MISZTAL M. K.: Deformable simplicial complexes. Unpublished manuscript, 2010. 1, 3

[OF02] OSHER S. J., FEDKIW R. P.: *Level Set Methods and Dynamic Implicit Surfaces*, 1 ed. Springer, October 2002. 2

[PB07] PONS J.-P., BOISSONNAT J.-D.: Delaunay deformable models: Topology-adaptive meshes based on the restricted delaunay triangulation. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on 0* (2007), 1–8. 3

[Sta99] STAM J.: Stable fluids. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1999), ACM Press/Addison-Wesley Publishing Co., pp. 121–128. 2

[WBOL07] WENDT J. D., BAXTER W., OGUZ I., LIN M. C.: Finite volume flow simulations on arbitrary domains. *Graph. Models 69*, 1 (2007), 19–32. 2

[WT08] WOJTAN C., TURK G.: Fast viscoelastic behavior with thin features. In *SIGGRAPH '08: ACM SIGGRAPH 2008 papers* (New York, NY, USA, 2008), ACM, pp. 1–8. 2

[Zha05] ZHANG F.: *The Schur Complement and Its Applications*. Springer, 2005. 6

[ZTT05] ZIENKIEWICZ O., TAYLOR R., TAYLOR R.: *The finite element method for solid and structural mechanics*. Butterworth-Heinemann, 2005. 3