Simulation of Bubbles and Liquid Films

ByungMoon Kim* Yingjie Liu† Ignacio Llamas‡ Jarek Rossignac§
Georgia Institute of Technology

Abstract

Liquid and gas interactions often contain bubbles surrounded by thin liquid films. Simulation of these liquid films is challenging since they quickly become thinner than the grid resolution, which leads to premature bursting or merging of the bubbles. We prevent this thinning process by applying a disjoining force to the film, obtaining bubbles that last much longer without bursting or merging. The surface tension on the liquid film is the next diffuculty. Since the level set is not differentiable at the center of the thin liquid film, the curvature computed from the level set gradient is noisy, and the thin liquid film ruptures quickly. To prevent this, we compute the surface tension from the local isosurface, obtaining long-lasting liquid films. However, since bubbles stay longer without bursting or merging, the volume loss of each bubble is noticeable. To solve this problem, we modify the pressure projection to produce a velocity field whose divergence is controlled by the proportional and integral feedback. This allows us to preserve the volume or, if desired, to inflate or deflate the bubbles. In addition to premature bursting and volume change, another difficulty is the complicated liquid surface, which increases memory and computational costs. To reduce storage requirement, we collocate the velocity and pressure to simplify the octree mesh. To reduce the computational complexity of the pressure projection, we use a multigrid method.

1 Introduction

In real fluids, we often observe a bubble rising to surfaces, floating around for a while, and then bursting or merging. In soap water, the bubble will last much longer than in pure water. When multiple bubbles are rising, the bubbles will interact with other bubbles and liquids. If a large number of bubbles are rising, and if they do not burst, they will stack, forming the *wet foam*. The water between those stacked bubbles will drain, leaving micrometer-thin films of liquid between bubbles. This is called the *dry foam*. The micrometer-thin film maintains its thickness due to the disjoining pressure, which is a result of various molecular interactions. The simulation of liquids with thin film is still a challenging open problem.

On the Eulerian grid, bubbles are hard to simulate, since bubbles are made of thin liquid films, which are too thin to be represented on a regular or even an adaptive grid. Films thinner than grid resolution quickly break, making bubbles burst earlier than desired. This premature bursting hampers simulations of bubbles with thin liquid films. Since films are broken too early, bubbles are few and interactions are rare. Moreover, the bubbles may break even before they become smooth and round, since it takes time for the surface tension to balance the curvature of the bubble. Therefore, preventing thin films from bursting and making them last longer is essential in the simulation of bubble and thin film.

Another difficulty in the simulation of a thin film is the computation of the surface tension. When the level set method is used to represent liquid and gas domains, surface tension is computed by differentiating the level set function, which is differentiable near the interface. However, in the thin film, the level set is locally singular and hence not differentiable. Curvature computed by differentiating the level set function near the thin film is noisy. When this noisy curvature is used to compute the surface tension, the thin film breaks quickly.

Various chemical reactions may result in the volume change of bubbles. For example, bubbles are inflated in boiling water. An animation of such phenomena would require a method to control the volume of bubbles. In addition, the simulated bubble tends to lose or gain volume due to various reasons, for example, numerically inaccurate pressure projection and level set advection. We propose a volume control method using divergence as the control input. This allows us to arbitrarily change the volume of fluid without visual artifacts. The volume loss or gain of fluids can also be remedied by this method.

Finally, bubbles with thin films have a very complex interface, which can lead to high memory and computation costs. We propose methods that can reduce those costs. First, to reduce memory cost, we collocate velocity, pressure, and level set variables at the octree center. Second, to reduce the computational cost, we apply a multigrid method, which greatly reduces the computation time when the interface is extremely complex. In addition, thanks to the volume control, volume of fluid is well preserved or controlled. Therefore, we do not use the particle level set method, which requires significant amount of memory and computational costs.

2 Prior Arts

Significant progress has been made since the pioneering work on fluid simulation for computer graphics [KM90; OH95; FM96]. The stability problem of earlier work was addressed in [Sta99], in which semi-Lagrangian advection and pressure projection were introduced. This solution became popular for the simulation of incompressible fluids, such as smoke [FSJ01] and also for free surface flows [FF01; EMF02]. A well known drawback of the semi-Lagrangian approach [Sta99] is the excessive diffusion and dissipation. Several solutions were proposed such as vorticity confinement method [FSJ01], vortex particles [SRF05], vortex fluid [PK05], and higher-order advection methods such as [SSK05; KLLR05].

In contrast to the simulation of gaseous phenomena such as smoke, the simulation of liquids contains complicated changes of the liquid surface. Therefore, a method suitable to represent the liquid volume and surface and capable of handling changes of the liquid surface is needed. One such method that has been widely used recently is the level set method [OS88; OF02; Set99]. In [FF01], the level set method was used in fluid simulation to create a realistic liquid simulation, and in subsequent work [EMF02], the volume loss of the level set method was addressed by the particle level set method. This particle level set method has been broadly used in recent fluid simulation approaches [CMT04; GBO04; SRF05; ELF05; HK05; WMT05].

Even though the level set method allows simulation of a liquid's surface, the complexity of the surface that can be simulated is limited by the grid resolution. A significant improvement can be obtained by an adaptive grid such as an octree [LGF04; SY04]. In particular, [LGF04] showed that detailed liquid surfaces can be

*e-mail: bmkim@cc.gatech.edu
†e-mail: yingjie@math.gatech.edu
‡e-mail: llamasi@math.gatech.edu
§e-mail: jarek@cc.gatech.edu

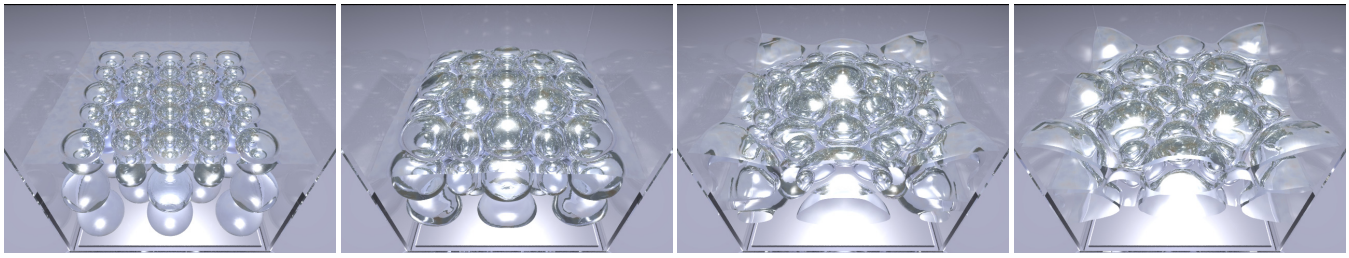

Figure 1: Simulation of 63 interacting bubbles. A disjoining force is applied to preserve the thin liquid film between bubbles, preventing unrealistic bubble merging and premature bursting. Physical realism is increased by computing surface tension from the liquid/air interface. A proportional and integral controller is used to maintain the volume of each bubble.

simulated by using an octree grid.

Prior liquid simulation researches [EMF02; LGF04; CMT04; GBO04; SRF05; ELF05] simulated liquids only, ignoring air. Therefore, liquid and air interactions, such as bubbles, requires an extension. In general, liquid/air phenomena can be simulated by the variable density pressure projection method that has been broadly studied in mathematics and fluid mechanics [SSO94; OKBG00; KFL00; HKLS04]. This variable density pressure projection has been used in graphics applications by [SSK05; KLLR05], in which splash and bubbles are simulated. We also notice the bubble simulation using Lattice-Boltzmann-Method(LBM) is studied in [NT04; PDT*04].

Although the two phase flow method can simulate bubbles, the practical simulation of foam, in particular, dry foam, is more challenging since the micrometer-thin liquid film is not captured by an Eulerian grid. Consequently most foam simulations have been performed by Lagrangian methods. For example, the authors of [KHVG02] approximated foam bubbles as spheres and then studied interactions between them. The interaction between bubbles and liquid is not studied. In contrast, the authors of [TFK*03] used particle to represent foams and simulated interactions with liquids. However, the formation, bursting, or merging of foam bubbles have not been addressed.

Thin liquid films are governed by molecular interactions rather than by the Navier-Stokes equations. An interesting outcome of such molecular interactions is the disjoining pressure, which acts along the direction normal to the film surface. This disjoining pressure causes a capillary suction, slowing down the drainage of liquid in the film. Thus, the thin film is maintained. This disjoining pressure, caused by molecular interactions such as electrostatic, van der Waals, steric, and adsorptional interactions, is the subject of numerous engineering and scientific studies [PK96; EK98; WH99; SvK04]. Besides those efforts to understand thin film, its simulation is rather rare. The authors of [WH99] constructed piecewise curves and patches to simulate a dry foam. The author of [Dur97] simulated 2D dry foam using circular bubbles. Because circular bubbles avoid the difficulties related to the thin films, it greatly simplifies the simulation of foam. The drawback of these methods would be the lack of bursting, merging, and interactions with liquid. The authors of [BvdVM01] simulated the behavior of a foam drop that is made of a few bubbles. They used a Lagrangian mesh to represent the liquid/gas interface. Simulations of various phenomena such as wet foam, formation of dry foam, bubble merging, and bursting were not attempted.

3 Fluid Simulation on an Octree Grid

We use the following Navier-Stokes equation for the fluid simulation.

$$\frac{\partial \mathbf{u}}{\partial t} = -\mathbf{u}\cdot\nabla\mathbf{u} + \nu\nabla\cdot(\nabla\mathbf{u}) - \frac{1}{\rho}\nabla P + \frac{\mathbf{f}}{\rho}. \qquad (1)$$

We follow the operator-splitting steps proposed in [Sta99] except for the projection step, where we use the variable density pressure projection. We use the octree grid with both the velocity and pressure located at the center of the cell.

3.1 Nonstaggered Octree Grid

Simulation of bubbles with thin films requires a high resolution mesh to represent thin films. In addition, liquid with multiple bubbles has a complex interface. Since a high-resolution octree grid containing a complex interface has a high memory cost, a memory-efficient implementation of an adaptive grid is desirable. Towards this goal, we simplify the octree grid representation used in [LGF04], by storing pressure and velocity at the center of the octree cell. Since all values are stored at the center of the octree cell, we do not need a parallel data structure for the cell corner, where velocities and level set values were stored in [LGF04]. In addition, velocity transfer [GSLF05] to the cell face is not needed. Therefore, implementation is simpler and memory-efficient. In our experiment, the 512^3 grid with a flat water surface and 63 bubbles underneath (see Fig. 1) has 14 million octree leaves. In this case, 2.5G bytes of memory was used. The 1024^3 grid with five bubbles under a water surface requires 2.2G bytes of memory. About 65% of this memory cost is used for the octree. The remaining 35% is used for the symmetric pressure projection matrix and for temporary vectors needed for conjugate gradient iteration. This 35% cost can be saved when the pressure projection is implemented directly on the octree. In our experiments, this implementation was about twice slower during the first few steps and then quickly become more than five times slower, because of the cache misses resulting from the address fragmentation produced during the dynamic allocation of memory for the octree nodes.

One drawback of our collocation is the complexity of the interpolation in semi-Lagrangian advection. However, we found that a continuous interpolation is not necessary, but a simple interpolation with neighboring cells suffices. We interpolate values with neighboring cells assuming that the cells are in the same tree depth, even when they are not. Since we do not allow more than a two-to-one depth ratio between adjacent cells, the largest tree depth difference ignored would be one. Suppose that we are performing an interpolation of ϕ at a point (x,y,z), which is contained in a leaf cell A. Assume that cell A has tree depth d. Interpolation involves the ϕ values of the neighboring nodes of A. First assume that a particular node B incident upon a face of A has depth d. If B is a leaf, we use its ϕ. If not, we use the average of its children. Now assume that the depth of B is less than d. It can only be $d-1$, since we ensure a 2-to-1 ratio between neighboring cells. We will use its ϕ.

Following [LGF04], we perturb the sampling points to obtain a symmetric pressure projection matrix. The two phase flow formulation is similar to [SSK05], except for the differentiation operator across different grid resolution. Notice that all our variables are

defined at the cell center. Therefore, the differentiations of these variables are the same as the differentiation of pressure in [LGF04].

3.2 Multigrid Solver

A high resolution mesh with a complex interface makes the pressure projection step very slow even in an adaptive grid. In addition, large surface tensions require small time steps. Therefore, simulation may be very slow. To reduce this computational cost, we use a multigrid solver. We applied the simple nested iteration discussed in [BHM00].

1. We first build a pressure projection operator and compute the divergence of the velocity at a coarse level.
2. We then solve the pressure projection equation using the conjugate gradient (CG) method with the Jacobi preconditioner. This will produce the pressure solution in a coarse grid.
3. We use the computed pressure as a starting estimate for the child-nodes at the next finer resolution.
4. Repeat steps 1, 2, and 3 until the maximum depth is reached.

We tested the effect of this approach in several experiments. As shown in Fig. 2, CG iteration tends to vary greatly when we do not use the nested iteration. This often occurs when the time step is small and the CG error bound is large. However, as shown in Fig. 2, when we used the nested iteration, the computation time become uniform. When the complexity of surface is high the benefit is significant. We conclude that the nested iteration is beneficial, especially when the interface is complex or is in high-resolution. When nested iteration is used, the pressure projection takes about $40 \sim 50\%$ of the total computation time.

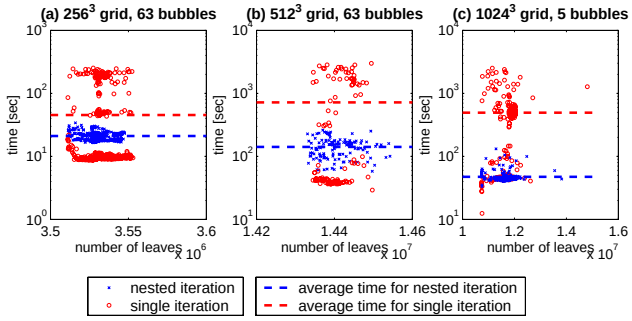

Figure 2: The computation time for the pressure projection step is reduced by nested iteration. The benefit increases as the grid resolution and number of leaves increases. Average pressure projection times are (a) nested:21.1sec, single:45.3 sec, (b) nested:141.2sec, single:718.6sec, (c) nested:47.5sec, single:494.4sec.

3.3 Level Set Advection

We use the level set method to trace the interface between liquid and gas. We use the BFECC (Back and Forth Error Compensation and Correction) and simple redistancing methods [DL03] that significantly reduce the volume loss of fluid thanks to increased second-order accuracies in space and time. Although this approach reduces the volume loss significantly, the volume loss is still noticeable in fluid simulation [KLLR05]. To solve this problem, we propose to use the new divergence control method discussed in a later section.

The BFECC method applied to level set advection tends to induce high-frequency noises on the interface wherever the velocity

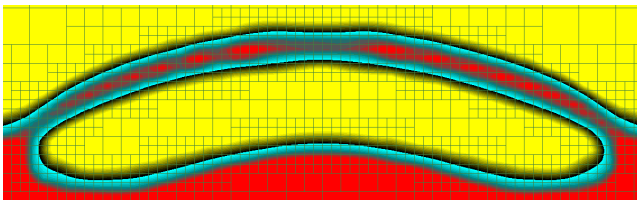

Figure 3: Contour of the level set function ϕ. Thin films contain series of local minima, and therefore, the gradient $\nabla\phi$ is not always perpendicular to the film surface. Therefore, the curvature vector computed from $\nabla\phi$ can be inaccurate.

field is not smooth. Therefore, the authors of [DL04] proposed a remedy to use the following semi-Lagrangian advection

$$\phi^{n+1} = \frac{1}{2}\left[\ \phi^n(-\mathbf{u}\Delta t + \varepsilon\mathbf{e}) + \phi^n(-\mathbf{u}\Delta t - \varepsilon\mathbf{e})\ \right], \qquad (2)$$

where $\mathbf{e} = (1,1,1)$ and $\varepsilon = 0.2\Delta x$. BFECC is implemented by performing this step three times per each time step. This oversampling and averaging adds small amount of diffusion, making the surface smooth. Since this oversampling point is always displaced along $\pm\mathbf{e}$, it may cause some artifacts along $\pm\mathbf{e}$. A possible solution would be oversampling in more directions, but it would be slow. Our solution is to randomly choose $\mathbf{e}$ from the four directions $\{(1,1,1),(1,-1,1),(-1,1,1),(-1,-1,1)\}$ in each time step.

3.4 Surface Tension

Real world bubbles have round and smooth shapes because the surface tension flattens the liquid/gas interface. In thin liquid films, surface tension can be a dominating force. Therefore, implementing surface tension is important in the simulation of thin films. Since the surface tension is proportional to the curvature normal vector of the interface, computation of the curvature normal vector is necessary. In the level set framework, this curvature normal is computed as $\nabla\frac{\nabla\phi}{|\nabla\phi|}$. However, along the center of the thin film, the level set function has a singularity, as shown in Fig. 3, and hence the gradient computed would not be accurate. Therefore, the curvature computed from the gradient of ϕ is noisy. In our experiments, the surface tension computed from the derivatives of ϕ produced artifacts on surface and an early breaking of thin films, even with a large disjoining force (described later) was active. See Fig. 5.

Therefore, a new approach is needed in order to simulate surface tension of the thin film. First, notice that even though the gradient of ϕ is inaccurate inside a thin film, the liquid surface of the film is smooth as shown in Fig. 3. This observation led us to the idea of computing the surface tension from the liquid/gas interface. Assume that we have a sufficient octree refinement near the interface so that all the interfaces are contained in cells of maximum depth. Consider two octree cells whose centers are A and B, as shown in Fig. 4. Suppose that ϕ has different signs at A and B and that the line $\overline{AB}$ is parallel to x-axis. The interface point P can be computed by linear interpolation of ϕ. The neighboring interface points Q_0, Q_1, Q_2, Q_3 can be easily computed. The local surface neighborhood around P contains more neighbors, but computing those additional neighbors and their connectivity would require iso-surface extraction algorithm. Since extracting the iso-surface at every time step is expensive, we developed a method to compute the curvature normal vector on P using the four neighborhood points $\{Q_0, Q_1, Q_2, Q_3\}$.

In estimating curvature normal, the discrete Laplace-Beltrami operator [DMSB99] would be a good candidate. However, the vertex P has only four neighbors $\{Q_0, Q_1, Q_2, Q_3\}$, and moreover, those neighboring vertices are not always evenly distributed, as

shown in the right image of Fig. 4. By the Taylor series expansion [Xu04] of the surface around P, it can be shown that the accuracy of discrete Laplace-Beltrami operator is low. Therefore, we develop a method that first computes a good normal direction and then a reasonable mean curvature value, using only $\{P, Q_0, Q_1, Q_2, Q_3\}$.

As shown in the left illustration of Fig. 4, the normal vector is computed as a normal to the two difference vectors of the two tangent pairs, i.e., $\mathbf{n} := (\mathbf{t}_{z_1} - \mathbf{t}_{z_0}) \times (\mathbf{t}_{y_1} - \mathbf{t}_{y_0})$, $\mathbf{n} := \mathbf{n}/|\mathbf{n}|$. In the right illustration of Fig. 4, the surface has small curvature but the angle between the two tangents $\mathbf{t}_{y_{0,1}}$ is small. Notice that a good normal vector is produced in this case as well. Once the normal

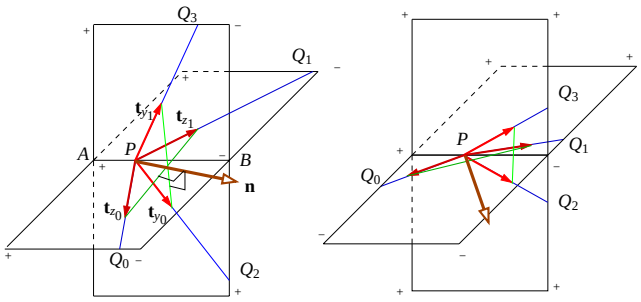

Figure 4: Computing a normal vector on a point where the liquid surface and $\overline{AB}$ are intersecting. In the right, we illustrate the case when the iso-surface has low curvature but the angle $\angle Q_2PQ_3$ is small. Notice that the normal vector $\mathbf{n}$ is computed properly.

vector is computed, we estimate the mean curvature from the angles between $\mathbf{n}$ and each of the tangent vector $\mathbf{t}_{z_0}, \mathbf{t}_{z_1}$, $\mathbf{t}_{y_0}$, and $\mathbf{t}_{y_1}$. We first compute the radius of a circle that encloses $\Delta x\mathbf{n}$ and $\Delta x\mathbf{t}_{z_0}$ by $r_0 = \Delta x\mathbf{t}_{z_0} \cdot \mathbf{n}$. Similarly, we compute $r_1 = \Delta x\mathbf{t}_{z_1} \cdot \mathbf{n}, r_2 = \Delta x\mathbf{t}_{y_0} \cdot \mathbf{n}$ and $r_3 = \Delta x\mathbf{t}_{y_1} \cdot \mathbf{n}$. We then average those radii, computing $r = (r_0 + r_1 + r_2 + r_3)/4$. The mean curvature is estimated as $\kappa = 2/r$, which allows us to compute the surface tension as $\gamma\kappa\mathbf{n}$, where γ is the surface tension coefficient. After $\gamma\kappa\mathbf{n}$ is computed, we multiply it with the weights $|\overline{PB}|/|\overline{AB}|$ and $|\overline{PA}|/|\overline{AB}|$, and apply them to A and B.

When the surface tension is applied to the Navier-Stokes equation, it must be divided by the density. However, in water and air interface, density ratio is near 1,000, causing a large velocity jump across the interface. In our experiment, this induces artifacts. Therefore, we divided $\gamma\kappa\mathbf{n}$ by the density of water uniformly near the interface since the water dominates the behavior near the interface.

4 Making Thin Film Last Longer

In real fluid, thinning of a film slows down due to the disjoining pressure that results from various molecular forces, such as the electrostatic, van der Waals, steric, and adsorptional interactions [PK96; EK98]. This disjoining pressure is active typically when the thickness is smaller than a micrometer, which is too thin to be captured by an Eulerian grid. Therefore, we apply disjoining forces in a film that is thinner than $4\Delta x$. This disjoining force is normal to the film and is applied in both directions to make the film thicker. As a result, a part of a film that became thinner than the threshold recovers its thickness. To this goal, we first detect cells inside the thin film, compute the normal vector and film thickness, compute the disjoining force, and finally, apply a symmetric set of forces that repulse the two film surfaces, making the film thicker.

To determine whether a cell is a thin film cell or not, we check sign changes of ϕ in each x, y, and z directions. We only explore two grid cells in each direction, since our threshold thickness is $4\Delta x$. If

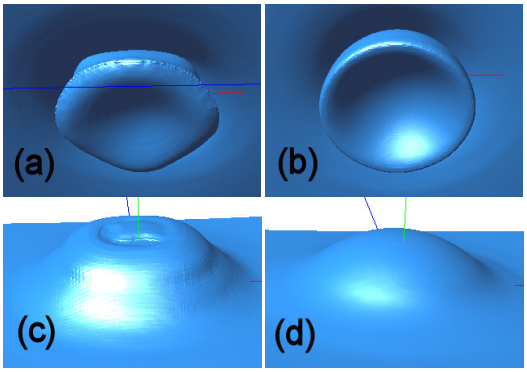

Figure 5: (a) The artifacts near the interface is resulted from errors in ϕ due to the simple redistancing and level set singularities. (b) This artifacts are removed when the curvature is computed from the local interface. (c) The inaccurate ϕ inside the thin film causes artifacts on the bubble surface, and premature bursting follows. (d) When the surface tension is computed from the local interface, the bubble surface is smooth and the bubble does not burst.

sign changes in both $\pm x$ directions are detected, the cell is marked as a thin film cell. Similarly, we apply this to $\pm y$ and $\pm z$ directions.

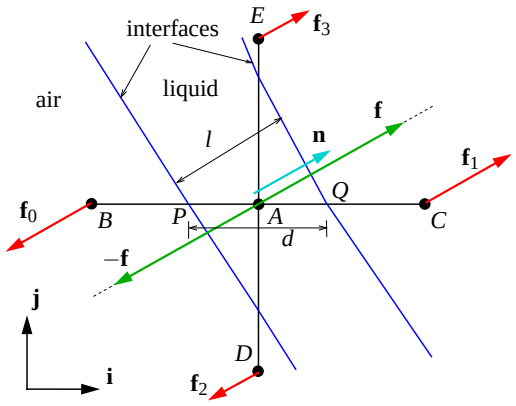

Figure 6: Disjoining force that prevents a thin film from bursting.

Figure 6 shows a fluid cell A that is inside a thin film and is surrounded by four air cells A, B, C, and D. We calculate forces that push the two surrounding interface away from A. In each of such thin film cell, we compute the normal direction to the thin film. The normal $\mathbf{n}$ at A is computed as the orthogonal vector of the average of the two tangents at P and Q. Thus, we avoid using the level set gradient that is not differentiable in the thin film, similarly to the surface tension case. The thickness of the film is computed as a projected length. For example, in Fig. 6, the thickness is computed by $l = d\mathbf{n} \cdot \mathbf{i}$. When sign changes are detected in more than one direction, we perform this operation on each direction and average them. Thus, we computed the normal direction $\mathbf{n}$, and the thickness of the thin film l at each film cells. We then compute the force $\mathbf{f}$ by

$$\mathbf{f} = \eta \left(2\left(\frac{l}{4\Delta x}\right)^3 - 3\left(\frac{l}{4\Delta x}\right)^2 + 1 \right)^{\alpha} \mathbf{n}, \qquad (3)$$

where η is the coefficient that determines the magnitude of force. In our experiments, when $\eta = 5,000$, thin films rupture rarely. The power α determines the thickness of the film. In most case, we use $\alpha = 1$, which produces film slightly thinner than $4\Delta x$. When $\alpha = 2$, the film thickness is reduced below $3\Delta x$.

The next step is to apply this force $\pm\mathbf{f}$ to the neighboring cells B, C, D, and E so that the interfaces are repulsed. Since these forces

applied to neighbors should not introduce motions other than thickening, we apply $\pm\mathbf{f}$ to B,C,D, and E so that the force and torque resultants are zero, i.e.,

$$\begin{aligned} \mathbf{f}_0+\mathbf{f}_1+\mathbf{f}_3+\mathbf{f}_3 &= 0 \\ (-\mathbf{i})\times\mathbf{f}_0 \;+\; \mathbf{i}\times\mathbf{f}_1 \;+\; (-\mathbf{j})\times\mathbf{f}_2 \;+\; \mathbf{j}\times\mathbf{f}_3 &= 0\ . \end{aligned} \tag{4}$$

Let $\mathbf{n}=(n_x,n_y)$. The forces $\mathbf{f}_0,\mathbf{f}_1,\mathbf{f}_2$, and $\mathbf{f}_3$ that satisfy (4) are computed as

$$\begin{aligned} \mathbf{f}_0 &= -\mathbf{f}_1 = -n_x\mathbf{f} \\ \mathbf{f}_2 &= -\mathbf{f}_3 = -n_y\mathbf{f}\ . \end{aligned} \tag{5}$$

Finally, $\mathbf{f}_0,\mathbf{f}_1,\mathbf{f}_2$ and $\mathbf{f}_3$ are added to B,C,D and E, respectively.

The extension to 3D is straightforward. Let $\mathbf{n}=(n_x,n_y,n_z)$, and let $\mathbf{f}_4$ and $\mathbf{f}_5$ be the forces on the two neighboring nodes in back and front of A, respectively. Then, the forces $\mathbf{f}_4$ and $\mathbf{f}_5$ are computed by $\mathbf{f}_4=-\mathbf{f}_5=-n_z\mathbf{f}$, while $\mathbf{f}_0,\mathbf{f}_1,\mathbf{f}_2$, and $\mathbf{f}_3$ are computed by (5).

In addition, in our experiments, the first-order level set advection yields early breaking of thin films. Since it is only first-order accurate, it yields fast local volume loss in the thin film, even when large disjoining force is applied. We conclude that a high-order level set advection method such as BFECC is necessary.

5 Controlling Fluid Volume

In this section, we present the volume control idea that allows arbitrary volume change. This tool is also used in correcting undesired volume loss or gain of the liquid drops, bubbles, as well as liquid body.

We begin with the brief overview of volume control. First, we segment the fluid domain into several connected components of liquid or gas. We then compute the volume error as a difference between the desired and current volume per each component. Using this volume error, we compute the divergence to inflate/deflate each component to correct volume error. In computation of the divergence, we use a feedback control strategy. Finally, we project the velocity field to the vector field with the computed divergence, which is performed in the pressure projection step. Notice that we apply divergence to all cells of each component except for the boundary cells. Therefore, added divergence is small and the overall fluid motion is affected little.

5.1 Segmentation and Tracking

A step needed for the volume control is segmenting the connected components of gas and liquids, computing volumes of components, and then tracking components movement, merge, and split in the following time steps. The segmentation is rather trivial in Eulerian grid using level set. We first consider the leaf cells as a graph, where two leaf cells are connected if they share a face. On this graph, we start a component from a leaf cell and grow that component by visiting neighborhood in a breadth first manner. When no connected cell has the same sign of ϕ, we start a new component. Since this requires only a normal queue rather than a priority queue, this is a linear time algorithm. The volume of the component is computed as the sum of the volumes of cubical leaf cells that belong to the component. We ignore the liquid-gas interface cutting the cells. Although this is an approximation, it only adds small amount of error.

The next step is tracking the connected components. Let $\mathscr{S}^n$ be the set of components in previous time step, and let $\mathscr{S}^{n+1}$ be the set of components in current time step. We compute flow from $\mathscr{S}^n$ to $\mathscr{S}^{n+1}$. We ignore the flow that is less than 10% of the initial volume of the component. Otherwise, we consider it as a volume transfer between components, indicating a merge and/or split. Notice that this approach handles simultaneous merges and splits between components.

5.2 Projection to a Controlled Divergence Field

Let $\mathbf{u}^*$ be the velocity computed before the pressure projection is applied. The modified pressure projection projects $\mathbf{u}^*$ to a velocity $\mathbf{u}^{n+1}$ that has the divergence c^n, i.e.,

$$\nabla\cdot\mathbf{u}^{n+1}=c^n \quad\Rightarrow\quad \nabla\cdot\frac{\nabla P}{\rho}=\frac{1}{\Delta t}(\nabla\cdot\mathbf{u}^*-c^n), \tag{6}$$

Since c^n is simply subtracted from the divergence, the complexity of the pressure projection step is not increased. The divergence value c^n is constant at all cells that belong to a connected component. In general, different connected components have different values of c^n.

5.3 A Feedback Controller

A connected component, such as a bubble, gains or loses little volume per time step. It typically takes almost 1,000 time steps until a bubble loses half of its volume even in a relatively coarse 128^3 grid (See Fig. 8). In other words, the volume change is not a high frequency behavior when BFECC is used for level set advection. This indicates that volume control should be relatively easy. In addition, this slow dynamic nature implies that a derivative feedback is not necessary and proportional feedback would suffice. Given this observation, we first designed a proportional controller.

In designing a control system, the first step is to define the state variable x. We use the normalized volume error

$$x_i^n=\frac{V_i^n-\tilde{V}_i}{\tilde{V}_i}, \tag{7}$$

where V_i is the current volume of the i^{th} component, and $\tilde{V}_i$ is its desired volume. Using this state variable x, the divergence of i^{th} component, denoted by c_i, is computed as the proportional feedback

$$c_i^n=-k_P x_i^n, \tag{8}$$

where k_P is the proportional gain.

When we applied this strategy to a rising bubble simulation, we observed that a rising bubble still loses volume until the loss $V_i-\tilde{V}_i$ is large enough for the proportional controller to produce large enough divergence c_i. As a result, the volume of a rising bubble tends to saturate to a slightly smaller volume. This drift error is often negligible. However, the amount of error is hard to expect. Therefore, we design additional control strategy to remove this drift error.

In the classical control strategy, a natural choice for removing drift error is using integral feedback, by which the small drift error will be integrated over time and then used as an additional control input. This is an efficient strategy since the small drift error can be accumulated to produce large control input. The proportional-integral (PI) controller is in the following form.

$$c_i^n \;=\; -k_P x_i^n \;-\; k_I\sum_{m=0}^{n} x_i^m\,\Delta t, \tag{9}$$

where k_P and k_I determines the amount of the proportional and integral feedbacks, respectively. k_P and k_I are constant numbers called the proportional and integral gains, respectively.

The next problem is to compute k_P and k_I that stably reduce the volume error x_i. Since it is rather complex, we provide the detail in the appendix. The gains k_P and k_I are computed as

$$k_P=\frac{9.2}{n_p\Delta t}\quad,\quad k_I=\frac{1}{4}\left(\frac{k_P}{2\zeta}\right)^2, \tag{10}$$

where n_p and ζ are variables that the user can choose. n_p is the number of time step required to reduce the volume error down to 90%. Suppose that a bubble has a large volume error. Then the error is reduced down to $0.1x$ in approximately n_p steps. Therefore, smaller n_p provides faster correction of error. However, large n_p can cause instability or noisy surface. In our experiment, $n_p = 50$ worked well in most cases. ζ is the damping coefficient of the typical second order equation $\ddot{y} + 2\zeta\omega_n\dot{y} + \omega_n^2 = 0$. In our experiments, the critical damping $\zeta = 1$ worked well.

5.4 Experiments

In Fig. 8, we test various controllers to prevent the volume change. In Fig. 9, we command the desired volume as a function of time, obtaining the simulation of a bubble whose volume is changing over time. Notice that when the desired volume is time-varying, the PI controller does not produce error converging to zero. This could be improved with derivative feedback [Oga90]. Since PI controller produces a small error, we do not use derivative feedback. The volume control tests in Fig. 7 and 8 are performed on 128^3-equilivalent grids.

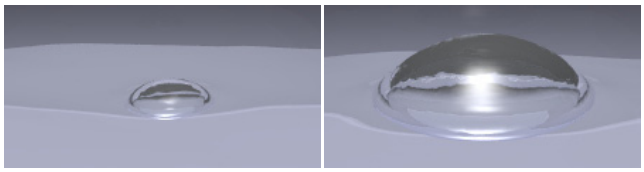

Figure 7: The bubble shrinks when the volume control is not applied (left). When the volume control is applied, the volume is preserved (right). See Fig. 8 for the volume change over time.

6 Results

We simulated bubble interactions in several situations. In Fig. 1, we simulated 63 bubble rising, interacting including merging in a 256^3-equilivalent grid. At the end of the simulation, the bubbles tends to form a structure similar to the wet foam. The average computation is about 50 seconds per time step ($\Delta t = 0.00002 sec$) and the whole simulation took about three days on an Athlon 64 3200 (2GHz) machine. We choose $\eta = 50{,}000$. Densities and viscosities are those of air and water.

In Fig. 10, we create 21 bubbles on a 512^3-equilivalent grid and then inflate them as a linear function of time. The disjoining force prevents bubbles from bursting. Since the surface complexity increases as bubbles grow, the computation time per time step increased from 45sec at the beginning to 140sec at the end on an Athlon 64 3200 (2GHz) machine. The whole simulation took about a week ($\Delta t = 0.00002$, total 6,544 time steps).

7 Conclusion

We have shown that the various bubble interactions such as stacking, merging, splitting, and bursting, can be simulated by using an octree grid, by computing the surface tension from the thin film interface, and by applying a disjoining force that delays the thinning process of the film that separates bubbles. The potentially high memory and computation costs of these simulations can be reduced by using a simplified octree and a multigrid method. In addition, we have shown that the volume of fluid can be preserved or arbitrarily changed using divergence as a control variable. We adopted the PI control strategy, developed a method to compute control gains, and showed its effectiveness on several examples.

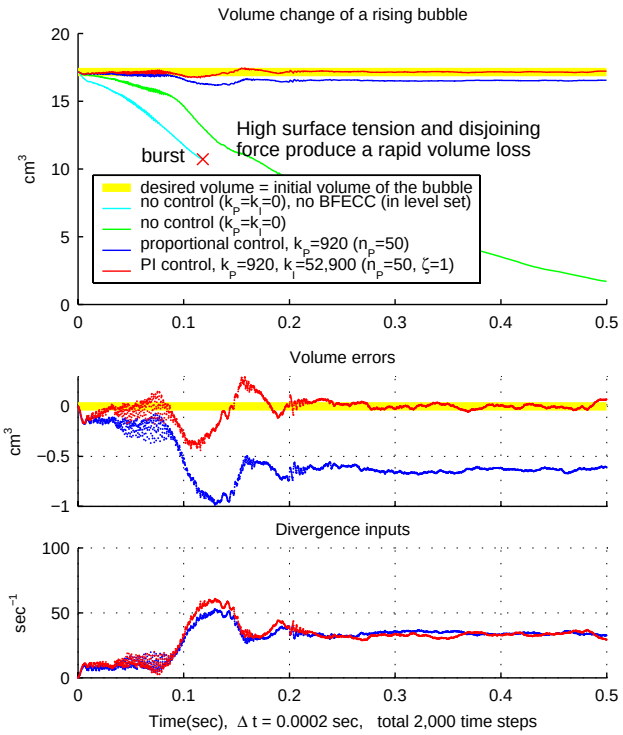

Figure 8: Controlling the volume of a rising air bubble. The proportional controller (blue) can keep the volume error small. The small steady-state error is fixed when the proportional-integral (PI) controller is used (red). As shown in the bottom figure, the PI controller increases the divergence input little.

References

BRIGGS W. L., HENSON V. E., MCCORMICK S. F.: *A Multigrid Tutorial*. Siam, 2000.

BAZHLEKOV I. B., VAN DE VOSSE F. N., MEIJER H. E.: Boundary integral method for 3d simulation of foam dynamics. In *Lecture Notes in Computer Science* (2001), pp. 401–408.

CARLSON M., MUCHA P. J., TURK G.: Rigid fluid: Animating the interplay between rigid bodies and fluid. In *SIGGRAPH* (2004), ACM.

DUPONT T. F., LIU Y.: Back and forth error compensation and correction methods for removing errors induced by uneven gradients of the level set function. *Journal of Computational Physics 190*, 1 (2003), 311–324.

DUPONT T. F., LIU Y.: Back and forth error compensation and correction methods for semi-lagrangian schemes with application to interface computation using level set method. *CDSNS2004-399, School of Mathematics, Georgia Institute of Technology* (2004).

DESBRUN M., MEYER M., SCHRÖDER P., BARR A. H.: Implicit fairing of irregular meshes using diffusion and curvature flow. In *Proceedings of ACM SIGGRAPH* (1999), pp. 317–324.

DURIAN D.: Bubble-scale model of foam mechanics: Melting, nonlinear behavior, and avalanches. *Physical Review 55*, 2 (1997), 1739–1751.

EXEROWA D., KRUGLYAKOV P. M.: *Foams and Foam Films*. Elsevier, 1998.

ENRIGHT D., LOSASSO F., FEDKIW R.: A fast and accurate semi-lagrangian particle level set method. *Computers and Structures 83* (2005), 479–490.

ENRIGHT D., MARSCHNER S., FEDKIW R.: Animation and rendering of complex water surfaces. In *SIGGRAPH* (2002), ACM.

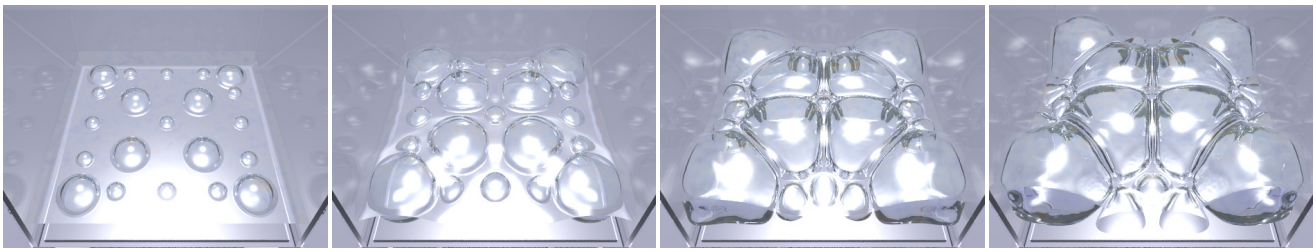

Figure 10: Inflating bubbles on a 512^3-equilivalent mesh. The volume of bubble is commanded to increase linearly in time.

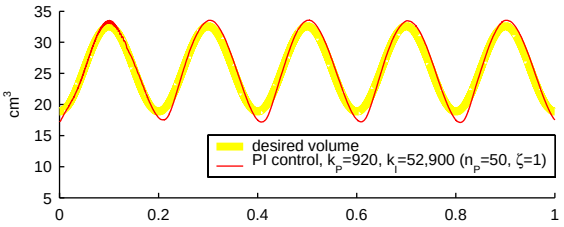

Figure 9: Controlling the volume of a rising air bubble whose commanded volume is defined as $\tilde{V}(t) = 0.4\sin(31.4t - 1.571)\tilde{V}^0$.

FOSTER N., FEDKIW R.: Practical animation of liquids. In *SIGGRAPH* (2001), ACM, pp. 15–22.

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

FEDKIW R., STAM J., JENSEN H.: Visual simulation of smoke. In *SIGGRAPH* (2001), ACM, pp. 23–30.

GOKTEKIN T. G., BARGTEIL A. W., O'BRIEN J. F.: A method for animating viscoelastic fluids. In *SIGGRAPH* (2004), ACM, pp. 463–468.

GUENDELMAN E., SELLE A., LOSASSO F., FEDKIW R.: Coupling water and smoke to thin deformable and rigid shells. In *SIGGRAPH* (2005), ACM.

HONG J.-M., KIM C.-H.: Discontinuous fluids. In *SIGGRAPH* (2005), ACM.

HAARIO H., KOROTKAYA Z., LUUKKA P., SMOLIANSKI A.: Computational modelling of complex phenomena in bubble dynamics: Vortex shedding and bubble swarms. In *Proceedings of ECCOMAS 2004* (2004).

KANG M., FEDKIW R. P., LIU X.-D.: A boundary condition capturing method for multiphase incompressible flow. *Journal of Scientific Computing 15*, 3 (Sep 2000).

KHALIL H. K.: *Nonlinear Systems*. Prentice Hall, 1996.

KÜCK, H. VOGELGSANG C., GREINER G.: Simulation and rendering of liquid foams. In *Proceedings of Graphics Interface* (2002), pp. 81–88.

KIM B., LIU Y., LLAMAS I., ROSSIGNAC J.: Flowfixer: Using bfecc for fluid simulation. In *Eurographics Workshop on Natural Phenomena* (2005).

KASS M., MILLER G.: Rapis stable fluid dynamics for computer graphics. In *SIGGRAPH* (1990), ACM, pp. 49–57.

LOSASSO F., GIBOU F., FEDKIW R.: Simulating water and smoke with an octree data structure. In *SIGGRAPH* (2004), ACM, pp. 457–462.

NILS THUREY U. R.: Free surface lattice-boltzmann fluid simulations with and without level sets. In *Workshop on Vision, Modeling, and Visualization* (2004).

OSHER S. J., FEDKIW R. P.: *Level Set Methods and Dynamic Implicit Surfaces*. Springer-Verlag, 2002.

OGATA K.: *Modern Control Engineering*. Prentice Hall, 1990.

O'BRIEN J., HODGINS J.: Dynamic simulation of splashing fluids. In *SIGGRAPH* (1995), ACM, pp. 198–205.

OEVERMANN M., KLEIN R., BERGER M., GOODMAN J.: *A Projection Method for Two-Phase Incompressible Flow with Surface Tension and Sharp Interface Resolution*. Tech. Rep. ZIB-Report 00-17, Konrad-Zuse-Zentrum für Informationstechnik Berlin, 2000.

OSHER S., SETHIAN J. A.: Fronts propagating with curvature-dependent speed: Algorithms based on hamilton-jacobi formulations. *Journal of Computational Physics 79* (1988), 12–49.

POHL T., DESERNO F., THUREY N., RUDE U., LAMMERS P., WELLEIN G., ZEISER T.: Performance evaluation of parallel large-scale lattice boltzmann applications on three supercomputing architectures. In *Supercomputing, 2004. Proceedings of the ACM/IEEE SC2004 Conference* (2004).

PRUD'HOME R. K., KHAN S. A.: *Foams: Theory, Measurements, and Applications*. Marcel Dekker, Inc, 1996.

PARK S. I., KIM M. J.: Vortex fluid for gaseous phenomena. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2005).

SETHIAN J. A.: *Level Set Methods and Fast Marching Methods*. Cambridge University Press, 1999.

SELLE A., RASMUSSEN N., FEDKIW R.: A vortex particle method for smoke, water and explosions. In *SIGGRAPH* (2005), ACM.

SONG O., SHIN H., KO H.: Stable but nondissipative water. *ACM Transactions on Graphics 24*, 1 (2005), 81–97.

SUSSMAN M., SMEREKA P., OSHER S.: A levelset approach for computing solutions to incompressible two-phase flow. *Journal of Computational Physics 114*, 1 (1994), 146–159.

STAM J.: Stable fluids. In *SIGGRAPH* (1999), ACM, pp. 121–128.

STUBENRAUCH C., VON KLITZING R.: Disjoining pressure in thin liquid foam and emulsion films - new concepts and perspective. *Journal of Physics Condensed Matter 15*, 3-4 (2004), 173–181.

SHI L., YU Y.: Visual smoke simulation with adaptive octree refinement. In *Computer Graphics and Imaging* (2004).

TAKAHASHI T., FUJII H., KUNIMATSU A., HIWADA K., SAITO T., TANAKA K., UEKI H.: Realistic animation of fluid with splash and foam. In *EUROGRAPHICS* (2003), vol. 22.

WEAIRE D., HUTZLER S.: *The physics of foams*. Oxford, 1999.

WANG H., MUCHA P. J., TURK G.: Water drops on surfaces. In *SIGGRAPH* (2005), ACM.

XU G.: The convergent discrete laplace-beltrami operator over triangular surfaces. In *Proceedings of Geometric Modelling and Processing (GMP2004)* (2004), pp. 195–204.

8 Appendix

In this section, we show the derivation of the control gains given in (10). We first start by constructing a mathematical model of the volume change.

8.1 Building a Volume Change Equation

The volume change could be due to the inaccuracies in several steps such as the pressure projection and level set advection. We also obtain error in the volume computation. We can assume that this volume change error per time step due to the inaccuracies of the simulation is small and that large enough divergence can dominate the volume change.

To make the analysis easy, we use the continuous system, i.e., we write $x_i = x_t(t), c_i = c_i(t)$ as functions of time. From the divergence theorem, the volume rate is $\dot{V}_i = V_i c_i$. However, since we do not apply the divergence to cells near the interface, the effective volume is slightly smaller. In addition, the inaccuracies in the level set advection and the pressure projection exist. We amortize those unknown factors in b. Then, the volume rate would be $\dot{V}_i = b\, V_i c_i$, which yields

$$\dot{x}_i \;=\; \frac{\dot{V}_i}{\tilde{V}_i} \;=\; b\,\frac{V_i c_i}{\tilde{V}_i} \;=\; b\,(x_i+1)c_i. \tag{11}$$

When the proportional feedback $c_i = -k_P x_i$ is used, $x_i = 0$ is a locally asymptotically stable equilibrium [Kha96] and x_i converges to 0 for any initial value in $(-1,\infty)$, since the function x_i^2 is strictly decreasing, i.e., $\frac{dx_i^2}{dt} = -bk_P x_i^2(x_i+1) < 0$ on $(-1,\infty)-\{0\}$. However, developing a systematic strategy to compute k_P that provides fast and stable error reduction is much harder with nonlinear equations. Therefore, to facilitate further analysis, we linearize the equation by assuming a small volume error, i.e., $|x_i| \ll 1$, yielding

$$\dot{x}_i \;=\; b\,c_i. \tag{12}$$

Now, we can develop a strategy to compute the gains k_P and k_I. We first compute k_P and then compute k_I.

8.2 Computing Proportional Gain k_P

In a high resolution mesh with small surface tension, since volume loss or gain occur slowly, the proportional control may keep the volume loss unnoticeable. Indeed, in most case, the proportional control seems sufficient. Therefore, we first develop a simpler proportional controller, i.e., we provide a method to compute k_P when $k_I = 0$.

Using the proportional feedback $c_i = -k_P xi$, the system equation (12) is written as

$$\dot{x}_i = -bk_P x_i, \tag{13}$$

and its discrete form is $x_i^n = \left(e^{-bk_P\Delta t}\right)^n x_i^0$. Let n_P be the number of time steps required to reduce the initial error x_i^0 down to 10%. Then n_P is computed from the condition $\left(e^{-bk_P\Delta t}\right)^{n_P} = 0.1$ as

$$k_P = \frac{-\ln 0.1}{n_P b\Delta t} = \frac{2.3}{n_P b\Delta t}, \tag{14}$$

where n_P can be translated as the number of time steps required to reduce error down to 10%. We estimate b from the response of a step input. We set the volume error $x_i = 0.5$ and then apply the volume control. The error x_i will quickly decrease towards zero. The time taken until the error is reduced by 90% of initial error is considered as the rising time. We measure this rising time, and then from the condition $\left(e^{-bk_P\Delta t}\right)^{n_P} = 0.1$, we can estimate b. We observed that $b \approx 0.25$ for a single rising bubble on a 128^3 grid. The value of b will change depending on different situations, but it is hard to predict and may become noticeable occasionally. Therefore, we propose to choose $b = 0.25$. Then the proportional gain is computed as

$$k_P \;\approx\; \frac{9.2}{n_P\Delta t} \tag{15}$$

Even though this is obtained with a number of assumptions, it provides a reasonable guideline in choosing the gain k_P. In our experiments, $n_P = 50$ worked well.

We would like to notice that the unknown factors, such as changing values of b, can be tracked by inserting an estimator. However, in Fig. 8, notice that the proportional gain k_P provides an already good result and that the integral control provides further improvements as discussed in the next section. Since these simple controllers leave little room for improvement, we do not peruse more advanced controllers.

8.3 Computing Integral Gain k_I

The drift error of proportional control can be removed by adding integral feedback. However, improperly chosen integral gain k_I can cause undesired oscillations in volume, which indeed occurred in our simulation of stacked bubbles. This oscillation was removed when we increased the damping. Therefore, we propose a method to compute a gain k_I that provides a good damping.

By substituting c_i in (12) by the PI-controller, we obtain

$$\dot{x}_i \;=\; -\; bk_P x_i \;-\; bk_I \int_0^t x_i. \tag{16}$$

Let $y = \int_0^t x_i dt$. Then, we obtain a second order system

$$\ddot{y} \;+\; bk_P\dot{y} \;+\; bk_I y = 0, \tag{17}$$

whose natural frequency is $\omega_n = \sqrt{bk_I}$ and the damping coefficient is $\zeta = \frac{bk_P}{2\sqrt{bk_I}} = \frac{k_P\sqrt{b}}{2\sqrt{k_I}}$. Our goal is now choosing good values of ζ that provides enough damping. By the classical control theory, a system that has a good balance between fast regulation and damping would have $\zeta = 0.7$, which contains a small amount of oscillation that settles down quickly. When $\zeta \geq 1$, the system is critically damped or overdamped, and therefore, oscillation in x_i does not exist (an overshoot is possible). Notice that equation (16) is obtained by making a number of assumptions. Therefore, it would be safe to choose ζ slightly larger than 0.7. In our experiments, $\zeta = 1$ worked well. After ζ is chosen, the integral gain k_I is computed as

$$k_I = \left(\frac{k_P\sqrt{b}}{2\zeta}\right)^2 = b\left(\frac{k_P}{2\zeta}\right)^2 \;\approx\; \frac{1}{4}\left(\frac{k_P}{2\zeta}\right)^2 = \frac{k_P^2}{16}, \tag{18}$$

where the proportional gain k_P is computed from (15).