Improved Water Sound Synthesis using Coupled Bubbles

KANGRUI XUE, Stanford University, USA
RYAN M. ARONSON, Stanford University, USA
JUI-HSIEN WANG, Adobe Research, USA
TIMOTHY R. LANGLOIS, Adobe Research, USA
DOUG L. JAMES, Stanford University, USA

We introduce a practical framework for synthesizing bubble-based water sounds that captures the rich inter-bubble coupling effects responsible for low-frequency acoustic emissions from bubble clouds. We propose coupled-bubble oscillator models with regularized singularities, and techniques to reduce the computational cost of time stepping with dense, time-varying mass matrices. Airborne acoustic emissions are estimated using finite-difference time-domain (FDTD) methods. We propose a simple, analytical surface-acceleration model, and a sample-and-hold GPU wavesolver that is simple and faster than prior CPU wavesolvers.

Sound synthesis results are demonstrated using bubbly flows from incompressible, two-phase simulations, as well as procedurally generated examples using single-phase FLIP fluid animations. Our results demonstrate sound simulations with hundreds of thousands of bubbles, and perceptually significant frequency transformations with fuller low-frequency content.

Additional Key Words and Phrases: Fluid animation, sound synthesis, acoustic bubbles

ACM Reference Format:
Kangrui Xue, Ryan M. Aronson, Jui-Hsien Wang, Timothy R. Langlois, and Doug L. James. 2023. Improved Water Sound Synthesis using Coupled Bubbles. *ACM Trans. Graph.* 42, 4 (August 2023), 13 pages. https://doi.org/10.1145/3592424

1 INTRODUCTION

Water is ubiquitous in our daily lives. Whether pouring a cup of tea, taking a bath, or sitting near a stream or ocean, we see, and *hear* water daily. Much research has gone into the realistic visual simulation and rendering of fluids. However, it is only within the last decade or so that efficient methods for the simulation of water sound have been explored.

The majority of water sound is the result of volumetric oscillation of bubbles. Once excited, bubbles vibrate (harmonically, in the case of a spherical, isolated bubble), giving off pressure waves that propagate through the fluid and cause the water surface to vibrate (as if it is a shape-changing loudspeaker). Previous simulation methods have explored this phenomenon and some of its complexities: the oscillation frequency is affected by bubble size, shape, and position relative to interfaces; several different mechanisms are responsible for exciting the bubble vibrations; the radiation of pressure fields between the liquid and air is a complex problem that requires time domain methods for realism (transient effects are strong even though ideal bubbles vibrate harmonically). All these mechanisms combine to produce the familiar "blooop" sound of a dripping faucet, and so much more.

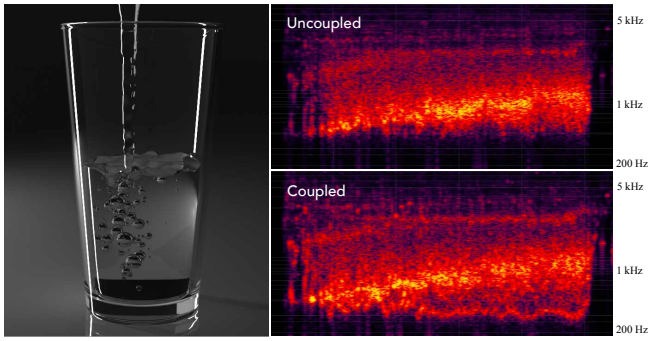


Fig. 1. **Better with *Coupled* Bubbles:** (Left) By simulating the collective oscillations of entrained bubble clouds, richer fluid sounds can be achieved with fuller, low-frequency acoustic emissions. (Right/Top) Spectrograms of the fluid sound before bubble coupling lack low-frequency content, whereas (Right/Bottom) those after bubble coupling exhibit a lower and wider range of frequencies. (Procedural bubbles added to a Houdini FLIP simulation.)

One fluid sound effect that has not yet been synthesized for fluid animations is an emergent phenomenon wherein clouds of tiny bubbles coordinate to produce low-frequency acoustic emissions. While the "bloop" of a single bubble and the low-frequency roar of the ocean sound quite different, they are both produced by bubble vibrations. Previous simulation methods have modeled bubbles as independent, but in reality, the pressure radiated from one bubble has a forcing effect on other bubbles. With small numbers of bubbles, this coupling effect is weak. But as the number of bubbles increases, the effect strengthens, so that even clouds of tiny bubbles can collectively oscillate to produce dramatic high- and low-frequency sounds such as the ocean roar. We present a method to efficiently simulate these collective oscillations for bubble clouds arising in fluid animation.

Our main contribution is a regularized coupled-oscillator model based on a dynamic dense mass matrix, along with methods for efficiently time-stepping the system at audio rates. The coupling forces are modeled assuming spherical bubbles; unfortunately, this

Authors' addresses: Kangrui Xue, Stanford University, USA, kangruix@stanford.edu; Ryan M. Aronson, Stanford University, USA, rmaronso@stanford.edu; Jui-Hsien Wang, Adobe Research, USA, juiwang@adobe.com; Timothy R. Langlois, Adobe Research, USA, tlangloi@adobe.com; Doug L. James, Stanford University, USA, djames@cs.stanford.edu.

0730-0301/2023/8-ART $15.00
https://doi.org/10.1145/3592424

can lead to singularities when applied to realistic bubble data, as nonspherical bubbles can become closer to each other than would otherwise be possible with their spherical equivalents. We introduce a method to regularize these singularities, allowing spherical coupling to be applied robustly to realistic bubble shapes as well as ad hoc procedural bubble animations. Furthermore, our regularization method ensures the positive-definiteness of the mass matrix for numerical stability, which we exploit to speed up numerical integration by interpolating inverses of the mass matrix. For large problems (thousands of bubbles or more per timestep), we also propose an approximate scheme based on low-rank updates to the Cholesky factorization. Finally, we explore both monopole- and dipole-coupling models, where the latter helps reduce over-coupling of bubbles near the fluid-air interface to those deep in the fluid. We demonstrate that our coupled-bubble framework can enhance the sound quality of both (1) expensive two-phase bubbly flows from previous works, and (2) low-cost procedural bubble animations more easily generated using single-phase liquid animation solvers commonly used in the computer animation community.

Finally, the airborne acoustic emissions of the bubbly flows are estimated using a GPU finite-difference time-domain (FDTD) wavesolver, building on prior CPU-based sound-source rendering approaches [Wang et al. 2018]. Such approaches can produce high-quality fluid sounds, but (a) they can require specialized fluid-boundary acceleration data, (b) the cut-cell FDTD CPU computations are slow, and (c) the higher-order cut-cell computations, while capable of low-noise sound synthesis, require the rasterized interfaces to be well-resolved spatiotemporally to avoid numerical artifacts – a challenge for many audio-rate fluid animation examples. To address the first point (a), we introduce a simple analytical surface acceleration shader that can be rapidly computed on the CPU or GPU for arbitrary bubbly flows. Second, we propose an approximate FDTD scheme with sample-and-hold geometry that is (i) simple, fast, and easy to GPU accelerate; (ii) low noise due to piecewise constant geometry; and (iii) robust to rapidly changing and complex, (potentially) under-resolved fluid-air interfaces expected in computer animation workflows. We synthesize compelling fluid sounds at a fraction of the cost, with improved robustness.

2 RELATED WORK

There are multiple ways that fluids can make sound, but the most prominent in common scenarios is through bubble vibrations. Investigations into bubble sound date back over a century [Bragg 1920; Rayleigh 1917]. Minnaert [1933] calculated the frequency of an isolated, spherical bubble, which vibrates harmonically at a specific frequency dependent on its size. Strasberg [1953] extended the frequency calculation to account for bubble shape and surrounding geometry. Coupled oscillations have long been of interest in oceanography and military studies since the frequencies of independent/isolated bubbles are insufficient to describe the characteristic, low-frequency sound (roar) of ocean waves despite being produced, predominantly, by collections of small bubbles [Bolin and Åbom 2010; Etter 2018; Knudsen et al. 1948; Leighton 2012; Medwin and Beaky 1989]. In their pioneering work, Lu et al. [1990] used normal mode analysis of coupled bubbles to show that significantly lower frequencies could be produced by bubble clouds than by the bubbles in isolation, and these frequencies could account for wind-dependent noise observed in the ocean; later works also analyzed and modeled collective bubble oscillations to explain low-frequency acoustic radiation of breaking waves [Deane and Stokes 2010; Means and Heitmeyer 2001; Oguz 1994]. Many subsequent mathematical models of multiple scattering and self-consistent coupled-bubble dynamics have been studied to understand time-domain bubble phenomena important for sound synthesis, and Feuillade [2001] concluded that so-called self-consistent models, as used herein, are effective for coupled air bubble vibrations in water.

In a fascinating paper by Leroy et al. [2005], both theoretical models and experimental measurements were combined to analyze the vibration properties of instrumented "bubble clouds," including cases where N bubbles were held at fixed locations by an underwater net (see Figure 2 for one such example, with $N = 53$). They concluded that the dynamics of the bubble cloud are found to be well approximated by an N-DoF (degrees of freedom) harmonic oscillator with incompressible inter-bubble coupling forces similar to those used herein, and analyzed the eigenmodes and eigenfrequencies of the coupled system. Interestingly, they show clear evidence of emergent low-frequency phenomena, e.g., the lowest fundamental frequency of a 3D bubble cloud scaled as $f_0/\sqrt[3]{N}$ where f_0 is the uncoupled Minnaert frequency of the identical bubbles, and N is the number of bubbles in the cloud. Therefore, perceptually significant frequency changes can be produced by coupled bubbles, and large N can produce strong low-frequency modal contributions, e.g., a thousandfold increase in bubbles can produce a tenfold decrease in frequency. We later show (in §4.2) that our regularized coupling model's computed eigenfrequencies approximately matches those computed and measured by Leroy et al. [2005].

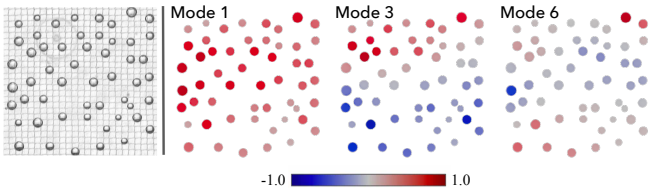


Fig. 2. **Leroy bubble cloud:** (Left) The original photograph of the 53-bubble cloud from Leroy et al. [2005] is shown. (Right) Using image analysis, we estimate bubble position and radii from the photograph and recompute the eigenmodes of the coupled system. Colors indicate signed volume pulsations and show strong collective oscillations.

Fluids are important in graphics, and as such there has been much work on their simulation and visual rendering. Early work focused on free surface methods [Bridson 2008; Enright et al. 2002; Osher and Fedkiw 2006; Stam 1999], which only simulate the fluid volume (treating air as a massless void) and therefore cannot simulate bubbles directly. Single-phase PIC/FLIP methods [Jiang et al. 2015; Zhu and Bridson 2005] are the most common commercial models used in graphics. Efficient, practical, and ad hoc methods have been explored for adding bubbles to fluid simulation [Goldade et al. 2020; Greenwood and House 2004; Kim 2010; Thürey et al. 2007], although they fail to capture the full range of multiphase effects.

Several techniques have been applied to multiphase flows, using SPH [Solenthaler and Pajarola 2008], FLIP [Boyd and Bridson 2012], power particles [de Goes et al. 2015], and level-sets [Kim 2010]. However, the accuracy of these methods has been restricted for multiple reasons: trouble handling large density ratios, blobbiness, and the need for explicit modeling of thin film dynamics. Only very recently have multi-phase Lattice-Boltzmann methods become more efficient [Li et al. 2022]. The Computational Fluid Dynamics (CFD) community has relied on two-phase, variable-density volume-of-fluid (VOF) methods to capture bubble shape, surface tension effects, and high density ratios [Popinet 2003]. We produce results with two methods: accurate two-phase VOF flows by rerunning several examples from Langlois et al. [2016], and buoyant bubble particles in a commercial FLIP solver (Houdini), generated using a stochastic method akin to [Zheng and James 2009].

Despite being the primary source of fluid sound, relatively little work has been done on simulating bubble sounds. Due to the complexity of simulating bubbles accurately, statistical methods have been proposed by [van den Doel 2005], which are fast but lack audio-visual coherency as they do not actually simulate bubble movement. More recent statistical methods [Bolsée and Bolsée 2018] incorporate more detailed models, but still hallucinate bubble movement and size. Database lookup methods [Saito et al. 2021; Wang and Liu 2018] are fast, but their quality depends on how accurately database samples match a given simulation.

To improve quality and audiovisual alignment over statistical methods, more physically based methods have been explored. To avoid the expense of a two-phase fluid simulation, several earlier works [Moss et al. 2010; Zheng and James 2009] augmented single-phase fluid simulations with ad hoc stochastic bubble models. However, this required laborious hand-tuning [Zheng and James 2009] or resulted in unrealistic bubble creation rates [Moss et al. 2010]. Zheng and James [2009] used an acoustic transfer model which could not capture scattering effects of surrounding containers, and Moss et al. [2010] ignored acoustic radiation. While it is possible for bubbles to vibrate at multiple frequencies by considering higher vibration modes (c.f. [Moss et al. 2010]) these modes do not radiate as efficiently, and Moss et al. [2010] only predict higher frequency contributions (whereas coupling accounts for lower-frequency acoustic emissions). However, dipole modes may be important for close bubble-bubble coupling [Leroy et al. 2018]. Langlois et al. [2016] used a two-phase VOF simulator [Popinet 2003] to compute accurate bubble shapes and motion, expanded on Strasberg's [1953] model to compute non-spherical bubble frequencies, and solved the full frequency-domain acoustic radiation problem to account for tank geometry. This came at a cost though, requiring days of simulation time even for small (0.08×0.08 m) scenes. Wang et al. [2018] demonstrated the importance of time-domain acoustic radiation by recomputing examples from [Langlois et al. 2016] using an FDTD (finite-difference time-domain) acoustic wavesolver. We introduce a model to capture coupling effects between bubbles, a strong effect that all previous methods ignore.

Oscillator coupling is also observed in modal sound models, where it can be introduced by contact [Zheng and James 2011] or material nonlinearities [Chadwick et al. 2009]. In the latter case of noisy thin shells, modal coupling can alter frequency responses noticeably.

3 MODELS OF COUPLED-BUBBLE DYNAMICS

3.1 Single-Bubble Dynamics

In the case of an isolated spherical bubble, models for its pulsating volumetric oscillations are well understood. Following [Langlois et al. 2016; Leighton 2012], let the average bubble volume be V_0, while $V(t)$ is the instantaneous volume. The infinitesimal volume pulsations, $v(t) = V(t) - V_0$, are governed by

$$m\,\ddot{v} + \alpha\,\dot{v} + \kappa\,v = p(t) \tag{1}$$

where $p(t)$ is a pressure-based forcing term. An equivalent, but more standard, form of the above equation is

$$\ddot{v} + 2\beta\,\dot{v} + \omega_0^2\,v = \frac{p(t)}{m} \tag{2}$$

where we have divided though by the bubble's effective mass m (in the volume-pressure frame) and denoted the undamped angular frequency as $\omega_0 = \sqrt{\kappa/m}$. For a spherical bubble of radius r in the volume-pressure frame, the effective mass is well approximated by $m = \frac{\rho}{4\pi r}$, and the stiffness is $\kappa = \frac{\gamma p_0}{V_0}$ (assuming adiabatic oscillations, and ignoring surface tension as is suitable for large bubbles [Leighton 2012]); p_0 is the background pressure in the fluid, and we use $\gamma = 1.4$ in air.

The damping coefficient β can be approximated as

$$\beta = \frac{\omega_0\,\delta}{\sqrt{\delta^2+4}} \overset{\delta\ll 1}{\approx} \frac{\omega_0\,\delta}{2} \tag{3}$$

(typically $\delta < 0.10$ for bubbles of interest) and involves radiation, viscosity and thermal contributions, $\delta = \delta_{rad} + \delta_{vis} + \delta_{th}$, whose formulae are evaluated at the resonant bubble frequency,

$$\delta_{rad} = \frac{\omega_0 r}{c}, \quad \delta_{vis} = \frac{4\mu}{\rho\omega_0 r^2}, \quad \delta_{th} = 2\frac{\sqrt{\psi-3} - \frac{3\gamma-1}{3(\gamma-1)}}{\psi-4}, \tag{4}$$

with

$$\psi = \frac{16}{9(\gamma-1)^2}\frac{G_{th}}{f_0}, \qquad G_{th} = \frac{3\,\gamma\,p_f}{4\pi\rho D_g}, \tag{5}$$

and c is the speed of sound in the fluid; ρ is the fluid density; γ is the heat capacity ratio of the air; G_{th} is the thermal damping constant at resonance; D_g is the thermal diffusivity of the gas; $f_0 = \omega_0/2\pi$ is the bubble's natural frequency; and p_f is the hydrostatic pressure of the fluid. In our implementation, for water at STP, we use the following values (in MKS units): $\gamma = 1.4$, $\rho = 1000$, $p_f = 101325$, $D_g = 2.12 \times 10^{-5}$, $G_{th} = 1.6 \times 10^6$ and $c = 1497$.

Finally, the computation of pressure forcing, $p(t)$, is handled as in [Langlois et al. 2016], which was based on models developed in [Czerski 2011; Czerski and Deane 2010, 2011; Deane and Czerski 2008]. In short, these forcing models are able to account for bubble entrainment, merging, and splitting events where surface tension effects are the primary drivers of bubble vibration.

3.2 A Simple Model of Coupled-Bubble Dynamics

Up to this point, however, the bubble oscillations have been considered to be independent. That is to say that the oscillations from one bubble within a bubble cloud do not affect any others. But intuitively, the pressure waves radiating from one bubble should have a forcing effect on the vibrations of any other bubble. When this is allowed

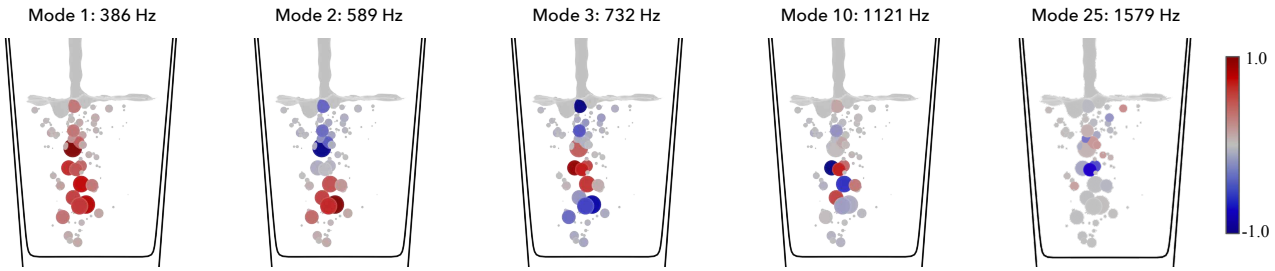


Fig. 3. **Bubble cloud vibration modes:** Through modal analysis, we can decompose collective bubble vibrations into instantaneous eigenfrequencies and volume velocity oscillation modes. Here, we consider the Glass Pour example at $t = 2.54$ seconds. Colors indicate signed volume pulsations (i.e., red for volume expansion, blue for volume contraction). At low eigenfrequencies, the entire bubble cloud tends to oscillate in-phase to produce a low rumble, while at higher eigenfrequencies, oscillation patterns become more localized, and individual bubbles dominate.

to happen, the bubble vibrations can couple together and produce collective volumetric oscillations which can change drastically from the independent, uncoupled case (as illustrated in Figure 3).

Coupling pressures. Consider the vibrations of N bubbles in a cloud to be described by a set of N coupled harmonic oscillators, each similar in form to (2). To couple the equations together, the pressure from the j^{th} bubble affects every other bubble through a coupling pressure added to the right-hand side of the equations:

$$\ddot{v}_i + 2\beta_i \dot{v}_i + \omega_i^2 v_i = \frac{p(t)}{m_i} - \sum_{j \neq i} \frac{p_j(\boldsymbol{x}_i, t)}{m_i} \tag{6}$$

where $p_j(\boldsymbol{x}_i, t)$ is the pressure produced by the j^{th} bubble at the centroid of the i^{th} bubble, and the minus sign implies that a positive coupling pressure acts to compress bubble i, as expected.

Monopole pressure field approximation. For spherical bubbles in isolation, the radiated pressure of bubble j is given by

$$p_j(\boldsymbol{x}, t) = \frac{\rho\, \ddot{v}_j(t)}{4\pi \|\boldsymbol{x} - \boldsymbol{x}_j\|} \tag{7}$$

where $\ddot{v}_j(t)$ is the instantaneous volume acceleration of bubble j (ignoring time delays). While exact for spherical bubbles, this pressure formula also provides a far-field monopole pressure field approximation for well-separated bubbles.

Spherical-bubble coupling model. For spherical bubbles, it is standard to substitute the (volume-pressure frame) mass, $m_i = \frac{\rho}{4\pi r_i}$, into the coupling term to obtain the coupled-bubble vibration equations [Feuillade 2001; Manasseh and Ooi 2009]:

$$\ddot{v}_i + 2\beta_i \dot{v}_i + \omega_i^2 v_i = \frac{p(t)}{m_i} - \sum_{j \neq i} \frac{r_i}{d_{ij}} \ddot{v}_j \tag{8}$$

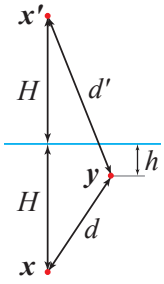
where $d_{ij} = \|\boldsymbol{x}_i(t) - \boldsymbol{x}_j(t)\|$ is the distance between bubbles i and j at time t. Numerically approximating the solutions to this coupled system of ordinary differential equations is the main task of §4.

Discussion. In utilizing these representations of the coupled system, we are making some important assumptions. First, the coupling term assumes incompressible flow, so that any bubble feels the disturbances from other bubbles instantaneously, which is a standard assumption in bubble cloud models where time-delays are small [Feuillade 2001; Leroy et al. 2005]. Second, the model is defined by assuming that each bubble is forced by a $1/d$ pressure field created by an isolated spherical monopole source, thus implying that these equations are only valid when the bubbles are well separated from one another ($d_{ij} \gg r_i$) [Feuillade 2001]. As we shall see, the bubbles in our simulation do not always satisfy this requirement, and the model must be altered in practice for stability (as described later in §4.2) or high-accuracy applications [Leroy et al. 2018].

Furthermore, we note that we have adopted the so-called *self-consistent approach*. In essence this means that the dependent variable v_i (the volume oscillation of bubble i) is defined to be the result once all of the infinitely many acoustic scattering processes have been accounted for. This contrasts with the *multiple scattering approach*, in which the effect of pressure waves that have been reflected back at a bubble are more explicitly accounted for. We found the self-consistent approach to be easier to implement for large bubble clouds, and it has been shown that, in some cases, the self-consistent approach is more robust at capturing strong coupling effects [Feuillade 2001]. We also believe the self-consistent approach is more practical for computer animation, where it is important to avoid instabilities and difficulties associated with integrating time-delay differential equations on dynamically changing oscillator banks.

3.3 A Dipole Coupling Extension for Surface Effects

Coupling bubbles together with a pressure model proportional to $1/d$ works for an infinite fluid domain, but it can "over couple" bubbles and be a poor approximation when one bubble is much nearer to the surface than the other. For planar fluid interfaces, we can use an image-source approximation to produce a dipole coupling model (see geometry inset) with a desired zero pressure boundary condition on the planar interface (in blue). Mathematically, we replace $1/d$ by $1/d - 1/d'$ where $d = \|\boldsymbol{x} - \boldsymbol{y}\|$ is the distance between the bubbles, and $d' = \|\boldsymbol{x}' - \boldsymbol{y}\|$ is the distance from the receiver at $\boldsymbol{y}$ to the image source bubble at $\boldsymbol{x}'$. It follows that the dipole $0 \leq 1/d - 1/d' \leq 1/d$, and so it reduces the coupling between two bubbles in cases where the distances d and d' are comparable, i.e., when one bubble is near the



interface. Conversely, the dipole is like the monopole when $d \ll d'$. In other words, shallow bubbles are decoupled from deep bubbles thereby allowing them to vibrate freely at the surface, and deep bubbles can still couple strongly when nearby ("deep coupling with shallow sparkling").

Unfortunately, while fluids do try to form planar interfaces, most fluid animations are nonplanar, and so this model would appear less practical. Analogous to the use of the planar dipole light scattering model used to render nonplanar subsurface scattering effects [Jensen et al. 2001], we use our dipole model with nonplanar fluid geometry by performing a change of variables. By Pythagoras we can write $d' = \sqrt{d^2 + (H+h)^2 - (H-h)^2}$, then we simply replace H and h by the distance to the fluid-air interface, which is exact for planar geometry and a useful generalization for other fluid interfaces. Please see our supplemental video for comparisons and evaluation.

4 PRACTICAL SIMULATION OF COUPLED BUBBLES

Now we turn to the details of how we developed practical methods for time stepping the coupled-bubble oscillators at audio rates. We address the formulation of an ODE system with symmetric mass matrix (§4.1) and the regularization of mass-matrix singularities (§4.2); the tracking of bubble creation and deletion events which change the oscillator bank size (§4.3); the approximation of mass-matrix factorizations for audio-rate integration (§4.4), and low-rank Cholesky updates for approximating the dynamics of very large systems (§4.5).

4.1 Matrix Symmetrization

When the equilibrium volume of the bubbles is constant (implied by the incompressibility assumption) we can symmetrize the oscillator matrix model (8) for numerical efficiency. Substituting $v_i = \sqrt{r_i} y_i$ into (8), we obtain

$$\sqrt{r_i}\,\ddot{y}_i + 2\beta_i \sqrt{r_i}\,\dot{y}_i + \sqrt{r_i}\,\omega_i^2\, y_i = \frac{p(t)}{m_i} - \sum_{j \neq i} \frac{r_i}{d_{ij}} \sqrt{r_j} \ddot{y}_j. \tag{9}$$

Dividing by the factor $\sqrt{r_i}$ and moving the coupling term to the left side of the equation, we get

$$\ddot{y}_i + \sum_{j \neq i} \frac{\sqrt{r_i r_j}}{d_{ij}} \ddot{y}_j + 2\beta_i\, \dot{y}_i + \omega_i^2\, y_i = \frac{p(t)}{\sqrt{r_i}\, m_i}. \tag{10}$$

Thus, written in matrix form the coupled system of equations is

$$\mathbf{M}\,\ddot{\mathbf{y}} + \mathbf{C}\,\dot{\mathbf{y}} + \mathbf{K}\,\mathbf{y} = \mathbf{F} \tag{11}$$

where $\mathbf{K}$ is the diagonal stiffness matrix, $\mathbf{C}$ is the diagonal damping matrix, and $\mathbf{M}$ is the dense, symmetric mass matrix with

$$M_{ij} = \delta_{ij} + (1 - \delta_{ij}) \frac{\sqrt{r_i r_j}}{d_{ij}} \tag{12}$$

where δ_{ij} indicates the Kronecker delta function.

4.2 Regularized Coupling

While the model described above has many advantages (e.g. simplicity, tested on small examples), it does have a major shortcoming that must be resolved before it can be used effectively in graphics applications. In the literature, this model was derived for bubbles that are spherical and well separated. However, as seen in [Langlois et al. 2016], accurate tracking of bubbles, including their deviations from spherical shapes, can be important in determining the produced sound. In addition, non-spherical bubbles can move closer together without touching or merging than spherical bubbles (distance measured between their centers of mass). Since this simple coupling model has no knowledge of actual bubble shape, the results produced with data sets such as the ones in [Langlois et al. 2016] can be nonphysical and unstable. Furthermore, we should expect that artist-generated procedural bubble models may generate overlapping or densely packed spherical bubbles, and we would still like the solver to tolerate imperfect bubble positions.

4.2.1 Coupling instability for non-spherical bubbles. To illustrate this problem more clearly, consider an artificial system of two elliptical bubbles (see Figure 4 (Left)) which can move very close together without merging. Such slender bubble shapes can occur, for example, as bubbles pass by one another or rise to the surface. Under the spherical bubble assumption, the coupling model interprets these as spherical bubbles with the same volume and centroid (see Figure 4 (Right)). Unfortunately the well-separated-bubble assumption of the coupling model is certainly violated, as the bubbles are overlapping, which manifests itself as a loss of positive definiteness of the mass matrix. The latter means that there are unstable modes within the dynamical system, which can lead to inaccurate and unstable results when integrating bubble dynamics. Specifically, since the eigenvalues of the 2x2 $\mathbf{M}$ are $\lambda = 1 \pm M_{12} = 1 \pm \sqrt{r_1 r_2}/d_{12}$, the dynamics are unstable when $\lambda \leq 0$, or $d_{12} \leq \sqrt{r_1 r_2}$. While this does not affect non-overlapping spherical bubbles (since $r_1 + r_2 > \sqrt{r_1 r_2}$), it does affect non-spherical bubbles, as described.

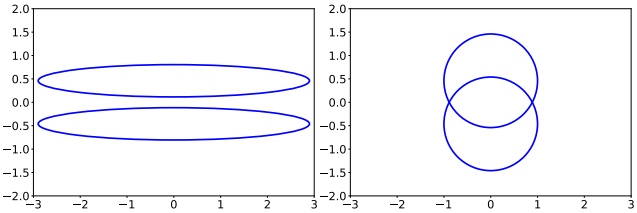


Fig. 4. **Non-spherical bubble interactions:** (Left) Two elliptical bubbles remain separated, but (Right) can overlap when interpreted as spherical bubbles of equal volumes and centroids, and can result in numerically unstable coupled-bubble dynamics.

4.2.2 Regularizing the singularity. Rather than attempt to define a fully general coupling model that works for arbitrary bubble shapes despite increased costs, we can stabilize the simple spherical-bubble coupling model to tolerate non-spherical bubbles in close proximity. The problem terms are the coefficients $\frac{\sqrt{r_i r_j}}{d_{ij}}$ which appear in the off-diagonal elements of the mass matrix $\mathbf{M}$. As bubbles move close together, these terms can become large, possibly even greater than one as in the above example. To regularize this instability, we replace these off-diagonal terms with

$$\tilde{\mathbf{M}}_{ij} = \frac{1}{\sqrt{\frac{1}{M_{ij}^2} + \varepsilon^2}} = \frac{1}{\sqrt{\frac{d_{ij}^2}{r_i r_j} + \varepsilon^2}}, \quad i \neq j, \tag{13}$$

Table 1. **Effect of regularization on coupling frequencies:** Using the 53-bubble cloud example from Figure 2, we compute the eigenfrequencies of the coupled system for varying ϵ and compare them with the four experimentally measured frequencies reported by Leroy et al. [2005]. Using $\epsilon = 0$ (no regularization), our estimated bubble configuration and coupling model is able to closely match the measured and computed frequencies of Leroy et al. [2005]. Using $\epsilon = 2$ (default regularization value), the four eigenfrequencies increase but still remain within 1.8% error on average.

Measured		*Computed*				
Peak	Frequency (kHz) [Leroy et al. 2005]	Mode	Frequency (kHz) [Leroy et al. 2005]	Ours ($\epsilon = 0$)	Ours ($\epsilon = 2$)	Ours ($\epsilon = 4$)
1	1.31 ± 0.01	1	1.25	1.25	1.26	1.31
2	1.93 ± 0.01	3	1.93	1.93	1.98	2.08
3	2.32 ± 0.01	6	2.30	2.31	2.36	2.46
4	2.80 ± 0.01	14	2.77	2.78	2.80	2.79

where $\varepsilon \geq 0$ is a regularization parameter.

4.2.3 Modal analysis of collective bubble oscillations. Given an instantaneous configuration of bubbles, we can perform modal analysis to estimate modal frequencies and the associated volume velocities of the bubble cloud. Consider the unforced, undamped, coupled oscillator system, $\tilde{\mathbf{M}}\,\ddot{\mathbf{y}} + \mathbf{K}\,\mathbf{y} = 0$. Assuming $\mathbf{y}(t) = e^{i\omega t}\bar{\mathbf{y}}$, the generalized eigenvalue problem is $\mathbf{K}\bar{\mathbf{y}} = \omega^2\tilde{\mathbf{M}}\bar{\mathbf{y}}$ where ω^2 is the eigenvalue and $\bar{\mathbf{y}}$ is the corresponding eigenvector. Please see Figure 3 for an illustration of the modal analysis from the Glass Pour example showing strong collective oscillations at lower frequencies.

We can also assess the effects of mass-matrix regularization on the cloud's modal frequencies and validate this approximation. When $\varepsilon = 0$ the regularized mass matrix reverts to the original coupling model, whereas as ε is increased, the effect of the coupling terms are weakened. Moreover, notice that this construction will have a larger effect on the coupling between close pairs of bubbles, which are strongly coupled, while having less of an effect on far-field coupling. Please see Table 1 for a comparison to numerical and experimental results from [Leroy et al. 2005]. Conditions on ε to guarantee the positive definiteness of the regularized mass matrix are given in Appendix B. In our examples we use $\varepsilon = 2$ (unless stated otherwise).

4.3 Bubble Tracking

From [Langlois et al. 2016], we have a method and dataset for tracking individual bubble events such as entrain, split, merge, and collapse – along with bubble radii, position, natural frequency, and pressure data at regular time intervals.

At each timestep, we track active bubbles (defined as bubbles that have been added through entrain, split, or merge events and have not yet been removed through collapse, splitting, or merging into other bubbles). All active bubbles are coupled together. For bubbles that are removed while they are still oscillating, care must be taken to avoid discontinuities. In our implementation, we let bubbles "ring out" upon removal, i.e., we uncouple them from other bubbles, freeze their position, radii, and other data, and continue timestepping their oscillations until they decay sufficiently.

We note that when bubbles move, the mass matrix changes, and when the number of bubbles active at any given time changes, the matrix must be resized. Moreover, when bubbles are simultaneously created and destroyed, it is possible that the system has the same size but is representing different bubbles. The dynamic nature of the dense mass matrix makes simulating these coupled ODEs unlike many of the discrete mechanical systems simulated in computer animation, where the mass matrix is often constant in time and sparse (or approximated by a diagonal lumped system).

Naïvely, the coupled-bubble mass matrix would need to be formed and inverted multiple times at each time step, which becomes prohibitively costly when integration is performed at (or near) audio rates. However, we note that the animation timescales of bubble movement are much slower than acoustic time scales for bubble vibration and audio time stepping. Consequently, the mass matrix changes very slowly, so we can devise methods to timestep the coupled-bubble dynamics at fast timescales while only periodically forming and factoring mass matrices.

Once we have the approximate inverse mass matrix $\tilde{\mathbf{M}}^{-1}$, the coupled oscillator equation can be solved in time using any standard numerical time integration scheme; we utilize Runge-Kutta (RK4) like prior work [Langlois et al. 2016; Wang et al. 2018] which aids comparisons.

4.4 Inverse Interpolation for $\tilde{\mathbf{M}}^{-1}$

Our first scheme for computing $\tilde{\mathbf{M}}^{-1}$ is to exploit temporal coherence by linearly interpolating the inverse mass matrix between endpoints. Consider a temporal splitting of the entire simulation time into short time segments with endpoints given by $t_0, t_1, ..., t_n$. Within each interval, $[t_k, t_{k+1}]$, we form the mass matrix at the endpoints, and can then invert (or factor) both matrices. At any intermediate time within a segment we create an approximation to $\tilde{\mathbf{M}}^{-1}$ by linearly interpolating the inverses:

$$\tilde{\mathbf{M}}^{-1}(t) \approx (1-\alpha)\tilde{\mathbf{M}}^{-1}(t_k) + \alpha\tilde{\mathbf{M}}^{-1}(t_{k+1}) \tag{14}$$

where $\alpha = \frac{t-t_k}{t_{k+1}-t_k}$ and $t \in [t_k, t_{k+1}]$. This approximation is then used to step the bubble system forward in time. The main advantage of this method (outside of its simplicity) comes from the convexity of symmetric, positive definite matrices. If we assume that the mass matrices formed at the endpoints of the intervals are positive definite (as guaranteed by regularized coupling in §4.2), then we know that the corresponding inverses are positive definite as well, and thus any approximate inverse formed using (14) will not contain any spurious, unstable modes.

At minimum, the mass matrix must be updated whenever bubbles are added or removed, since it doesn't make sense to interpolate between disjoint bubble clouds. For our many-bubble examples, this

requirement already implies fast updates. For instance, the 2016 Pour and 2016 Step examples from [Langlois et al. 2016] have mean segment times of 0.16 ms and 0.10 ms, or 7.8 and 4.8 timesteps (at 48 kHz), respectively. Nevertheless, interpolation does reduce occasional popping artifacts, e.g., arising with longer segments.

Thus, in all our examples, we define endpoints as times when bubbles are added or removed (which we compute beforehand). Concretely, we form and interpolate the Cholesky factorization of the mass matrices at each endpoint rather than the full inverse.

4.5 Low-Rank Updates

As the size of our bubble system increases, individual bubble details become increasingly difficult to resolve. If bubbles are added or removed at every timestep, even the above linear interpolation scheme would require refactoring the mass matrix each time. Here, we describe an approximate scheme based on stationary bubble positions that avoids full re-factorization of the mass matrix by leveraging low-rank updates to the Cholesky decomposition.

Consider the symmetric mass matrix $\tilde{\mathbf{M}}$ and its Cholesky factorization (in block matrix form) $[M_{11}] = [L_{11}][L_{11}^T]$, where L_{11} is lower-triangular. Upon adding bubbles to the system, we append rows and columns to the end of the matrix, so our updated factorization is given by:

$$\begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} = \begin{bmatrix} L_{11} & \\ L_{21} & L_{22} \end{bmatrix} \begin{bmatrix} L_{11}^T & L_{21}^T \\ & L_{22}^T \end{bmatrix}. \tag{15}$$

The Cholesky updates are thus given by $L_{21} = (L_{11}^{-1} M_{12})^T$ and $L_{22}L_{22}^T = M_{22} - L_{21}L_{21}^T$. Note that L_{21} can be obtained by performing a back-solve with lower-triangular matrix L_{11}, and L_{22} can be obtained from the Cholesky factorization of $M_{22} - L_{21}L_{21}^T$. In computing the M_{12} and M_{22} blocks, we fix all bubbles at their initial positions.

When removing bubbles from the system, there is no guarantee that the bubbles lie on consecutive rows and columns of the matrix. This means that instead of performing a single block matrix operation, we must perform a series of incremental updates. Furthermore, the bubbles we wish to remove often correspond to rows and columns in the center of the matrix. In the most general case, consider the block matrix factorization of $\tilde{\mathbf{M}}$ given by:

$$\begin{bmatrix} M_{11} & M_{12} & M_{13} \\ M_{21} & M_{22} & M_{23} \\ M_{31} & M_{32} & M_{33} \end{bmatrix} = \begin{bmatrix} L_{11} & & \\ L_{21} & L_{22} & \\ L_{31} & L_{32} & L_{33} \end{bmatrix} \begin{bmatrix} L_{11}^T & L_{21}^T & L_{31}^T \\ & L_{22}^T & L_{32}^T \\ & & L_{33}^T \end{bmatrix}. \tag{16}$$

Removing the middle row and column of $\tilde{\mathbf{M}}$, we get:

$$\begin{bmatrix} M_{11} & M_{13} \\ M_{31} & M_{33} \end{bmatrix} = \begin{bmatrix} L_{11} & \\ L_{31} & S_{33} \end{bmatrix} \begin{bmatrix} L_{11}^T & L_{31}^T \\ & S_{33}^T \end{bmatrix} \tag{17}$$

where we have defined lower-triangular matrix S_{33} to indicate that the entries change upon removing preceding rows and columns. S_{33} is given by $S_{33}S_{33}^T = L_{33}L_{33}^T + L_{32}L_{32}^T$. Note that since we are removing only one bubble, L_{32} is a column vector. Thus, the expression is of the form $\tilde{A} = A + xx^T$ (where $A = S_{33}S_{33}^T$ and $x = L_{32}$) and can be solved efficiently using rank-1 updates to $L_{33}L_{33}^T$ (as opposed to a full Cholesky factorization of $L_{33}L_{33}^T + L_{32}L_{32}^T$).

By default, bubbles are taken to remain stationary after being added. Periodically, we may update a bubble's position by modifying the corresponding row and column of the mass matrix as $\tilde{\mathbf{M}} := \tilde{\mathbf{M}} + \frac{1}{2}(x+y)^T + \frac{1}{2}(x-y)^T$. Concretely, when bubble i is updated, we set x as the unit basis vector e_i and y as the difference between the updated i-th column $\tilde{\mathbf{M}}$ and its previous value. However, this comes at the cost of two additional rank-1 updates per bubble. The approximation error of $\tilde{\mathbf{M}}$ is directly correlated to the rate at which bubble positions are updated.

In practice, using low-rank updates with stationary bubbles can exploit the temporal coherence in large mass matrices (see Figure 5).

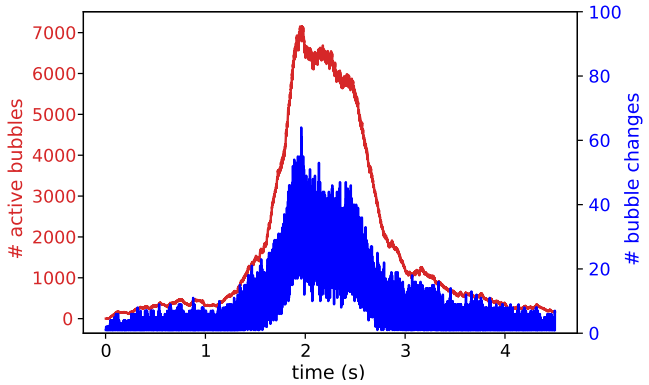


Fig. 5. **Low-rank update statistics:** The large Wave Crash example in Figure 12 has hundreds of thousands of bubbles, whereas the number of active bubbles at any given time is much less and plotted above (in red). Furthermore, the number of bubble changes at each solve is far lower (plotted in blue), which makes low-rank Cholesky updates efficient.

5 PRACTICAL SYNTHESIS OF AIRBORNE SOUNDS

Fluid animations can produce complex, dynamic fluid-air interfaces, posing unique challenges for bubble-based fluid sound synthesis methods that rely on expensive air-domain acoustic wave simulation. On the other hand, Wang et al. [2018] demonstrated that capturing air-domain waves is critical to the quality of the final fluid sound. To improve the speed and the robustness of finite-difference time-domain wavesolvers for airborne synthesis of coupled-bubble sounds, we make two practical contributions in this section: (1) an analytical approximation of normal-acceleration fluid-air boundary data for animation-quality fluid simulations, and (2) a sample-and-hold FDTD GPU wavesolver that enables faster sound generation than the prior high-quality CPU-based method [Wang et al. 2018] with some quality trade-offs but significant improvements in robustness and speed.

5.1 Surface-Acceleration Shader

To compute the wave equation's Neumann boundary condition, we need the normal acceleration of the fluid-air interface, which is computed using a surface-acceleration shader [Wang et al. 2018]. Examples in that paper used a time-harmonic surface vibration model based on prevailing mono-frequency harmonic bubble approximations [Langlois et al. 2016; Wang et al. 2018; Zheng and James 2009] which provide an estimate of the harmonic surface

acceleration function, e.g., from an interior Laplace or Helmholtz bubble-to-surface transfer approximation. These are relatively expensive computations which must be done for each bubble. In our case the bubble is not a harmonic source, but fortunately, the FDTD framework supports general time-varying bubble oscillations. Given the limited size of our fluid domains, and the fact that we already use incompressible fluid approximations for bubble-bubble coupling, we use a simpler approximation for surface accelerations.

Given the large number of bubbles we expect in fluid animation workflows, we use an approximate analytical point source for each bubble, and estimate the surface acceleration at any point using a superposition of spherical sources. Specifically, the velocity potential, velocity, and acceleration equations corresponding to the j^{th} spherical bubble's pressure (7) are

$$\phi(\boldsymbol{x},t) = -\frac{\dot{v}(t)}{4\pi\|\boldsymbol{x}-\boldsymbol{x}_j\|} \tag{18}$$

$$\boldsymbol{u}(\boldsymbol{x},t) = \nabla\phi = \frac{\dot{v}(t)}{4\pi\|\boldsymbol{x}-\boldsymbol{x}_j\|^3}(\boldsymbol{x}-\boldsymbol{x}_j) \tag{19}$$

$$\boldsymbol{a}(\boldsymbol{x},t) = \frac{\partial}{\partial t}\boldsymbol{u}(\boldsymbol{x},t) = \frac{\ddot{v}(t)}{4\pi\|\boldsymbol{x}-\boldsymbol{x}_j\|^3}(\boldsymbol{x}-\boldsymbol{x}_j) \tag{20}$$

$$a_n(\boldsymbol{x},t) = \boldsymbol{a}\cdot\boldsymbol{n} = \frac{\ddot{v}(t)}{4\pi\|\boldsymbol{x}-\boldsymbol{x}_j\|^3}(\boldsymbol{x}-\boldsymbol{x}_j)\cdot\boldsymbol{n} = F_j(\boldsymbol{x})\,\ddot{v}(t). \tag{21}$$

The acceleration shader then accumulates $a_n(\boldsymbol{x},t)$ contributions at each surface point by summing over all currently active bubbles. In practice, the spatial form factor, $F_j(\boldsymbol{x})$, for each bubble need only be updated at a temporal rate lower than the time-stepping rate; in our implementation we use update rates of at most one millisecond. Finally, for each bubble, we normalize its surface acceleration contribution $a_n(\boldsymbol{x},t)$ at each point by dividing by the bubble's net flux through the surface Φ_F, given by $\Phi_j = \sum_k A_k F_j(\boldsymbol{c}_k)$ where the summation is over all surface points, and A_k denotes the area of the k-th surface element, and $\boldsymbol{c}_k$ its centroid. This normalization is important for non-spherical domains, as it ensures the bubble's acceleration contribution is directed entirely through the water surface.

To validate our approximation, we compared sounds generated from prior work using the BEM-based acceleration shader for the 2016 Pour and 2016 Step example to our simple analytical monopole shader. The results are qualitatively similar (please see the supplemental video).

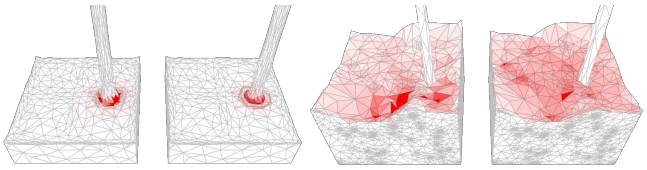


Fig. 6. **BEM vs. monopole shader comparisons:** Normalized acceleration shader values are plotted on meshes from the 2016 Pour example at different times. Darker shades of red indicate larger amplitudes. For the few bubble case, the monopole shader (b) is able to capture the spatial variations of the BEM solution (a). Furthermore, for under-resolved meshes, the monopole shader (d) is spatially smoother than the BEM shader (c).

5.2 Sample-and-Hold GPU Wavesolver

We improve upon the general-purpose CPU wavesolver introduced in [Wang et al. 2018] for our application. The solver in [Wang et al. 2018] generates low-noise rendering of vibrating fluid surfaces by use of a second-order boundary handling scheme. However, we found that this scheme, along with the point-based rasterizer in [Wang et al. 2018], results in a finicky solver that requires sufficiently high spatiotemporal resolution to work robustly, especially for our application where the fluid interface can be highly dynamic (see Figure 7). Moreover, the constantly changing fluid interface and the resulting rasterizations can cause a large performance penalty.

We improve *both* the robustness and speed of the solver by slightly sacrificing the quality using a simple sample-and-hold principle. Inspired by the parallel-in-time algorithm introduced in [Wang et al. 2018], we first segment the animation data (i.e., mesh and boundary conditions) temporally into wavesolve "chunks". We run each chunk on our GPU wavesolver with a fixed water interface (thus the name sample-and-hold). The bubble positions are still varying in time in each chunk and acceleration shaders are resampled accordingly (§5.1). Like [Wang et al. 2018], we continue running the simulation for a short window of time after each chunk boundary ("post-window") to ensure the resonating waves have decayed sufficiently. The solution for each chunk is then superimposed at the listening location to recover the full solution [Wang et al. 2018].

Considerations for selecting chunk sizes. Both the chunk size (10 ms) and the post-window size (10 ms) are chosen to be small. This has two benefits: 1) this ensures all animation data can fit comfortably in the GPU memory, and memory fetch strides will be kept minimal, and 2) the piecewise constant fluid interface change at the chunk boundary might result in frequency distortion; by keeping the chunks small, we minimize this artifact. In most of our examples, fluid interfaces can form intricate cavities which have strong resonance, and therefore the 10 ms post-window size is almost certainly not enough to capture energy decay in a naive implementation. On the other hand, extending the post-window duration so we can capture the full resonance is impractical as this is often on the order of several hundred milliseconds (more than 10 times our original chunk size!). Fortunately, the signal in the post-window is much more structured, consisting mostly of unforced oscillations of resonance (another benefit of the sample-and-hold approach). We leveraged this insight to build an audio-domain spectral decay extension model for each chunk to make the 10 ms post-window choice less noticeable, but the change was barely perceptible for our examples, so we simply play the chunk audio directly.

Stability. The simplicity of the sample-and-hold approach, combined with our use of a more conservative rasterizer, removed any observed instability in the previous solver [Wang et al. 2018]. We also use the first-order boundary handling scheme (i.e., staircasing) because the second-order scheme requires finer spatial resolution. See Table 2 for comparisons on stability and performance.

6 RESULTS

We integrate our coupled-bubble equations on a 13^{th} Gen Intel i9 CPU using a single-threaded implementation. FLIP simulations

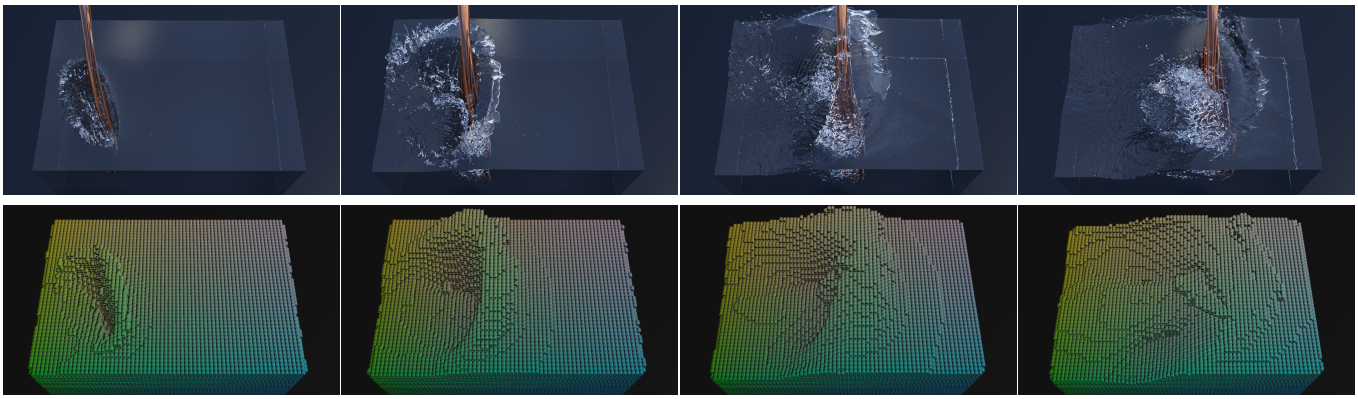


Fig. 7. **Overview of sample-and-hold GPU wavesolver:** The Paddle Splash animation demonstrates (Top) complex, dynamic fluid surfaces with detailed features which make robust, low-noise simulation of acoustic waves challenging. (Bottom) By freezing fluid and rigid scene geometry within short temporal chunks, we can use fast GPU-accelerated FDTD wavesolvers for static geometry, then combine the resulting audio waveforms for each chunk using linear superposition. The approach is approximate, but it is fast and robust for practical fluid animation workflows.

were run in Houdini FX v19.5, with our stochastic bubble model implemented as a Solver node on output FLIP fluid simulation data. The CPU wavesolver was run on a 2.3 GHz Intel Xeon CPU (with 36 cores), while the GPU wavesolver was run on an NVIDIA GeForce RTX 4090. Please see our supplemental video for animation sound results. For oscillator integration, all results use inverse interpolation for $\tilde{\mathbf{M}}^{-1}$ (§ 4.4) except for the Wave Crash example, where we use low-rank updates (§ 4.5) due to the enormous number of bubbles.

6.1 Two-Phase Simulation

We run our coupled-bubble model on prior examples from Langlois et al. [2016]. The datasets for these examples were generated using offline two-phase incompressible fluid simulators (and are included in the dataset of [Wang et al. 2018]). As a result, they exhibit reasonable bubble distributions along with plausible sounds even without coupling. However, we show how our coupled-bubble model can enhance these sounds.

In performing our comparison, we found that the water/air surface acoustic velocity data provided in that dataset was normalized incorrectly. In [Langlois et al. 2016; Wang et al. 2018], it is stated that it should be normalized for unit vibration, i.e., $\dot{v}(t) = \int_{Bubble} \partial_n \phi \, dS = 1$. However, it appears that after the data was initially computed with a unit potential $\phi = 1$ for each bubble, it was not renormalized. To correct this, we use the fact that, due to incompressibility, the dataset velocity simply needs to be normalized by its integral over the surface: $\int_{Bubble} \partial_n \phi \, dS = \int_{Fluid\ Surface} \partial_n \phi \, dS$. *Unfortunately, this implies that previous results from those papers were missing an ω proportional factor (causing higher frequencies to be suppressed)*, since for a harmonic spherical bubble with radius r, we have $\phi(r,t) = \frac{\dot{v}(t)}{4\pi r} = \frac{\dot{v}_0 e^{i\omega t}}{4\pi r} \implies |\dot{v}_0| = 4\pi r$ when $\phi = 1$, and the Minnaert frequency is $\omega_0 \approx \frac{3}{2\pi r}$.

2016 Pour: For smaller domains, our coupling model produces subtle but perceptually significant frequency transformations. The coupled audio exhibits the characteristic bimodal spectra of a pouring glass: a falling pitch from bubbles coupling as they move deeper into the fluid, along with a rising pitch from the amplification of frequencies (due to the container cavity resonance) that go up as the water level rises.

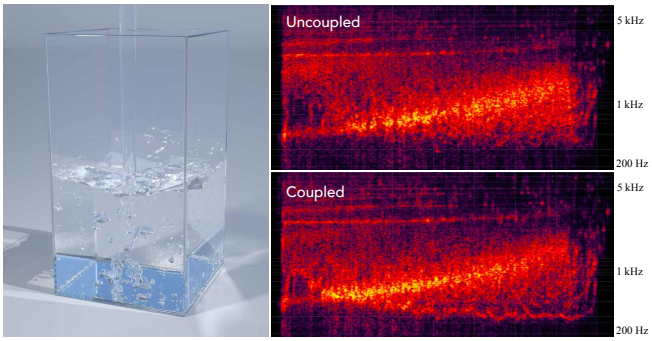


Fig. 8. **2016 Pour**

2016 Step: The 2016 Step example features a denser bubble distribution and steadier bubble generation rates compared to the 2016 Pour. We use a stronger regularization value of $\epsilon = 2\sqrt{2}$ (instead of the default $\epsilon = 2$) to achieve smoother audio results. The coupled audio is less shrill and livelier (with more frequency variations).

6.2 FLIP Simulation

In addition to enhancing the sound of high-fidelity bubble simulations, we test our method on procedurally-generated FLIP animations from Houdini, with ad hoc bubble generation processes and spherical bubbles. Except for the Glass Pour example, which takes minutes for the fluid and bubble solvers to run, our larger examples can be simulated overnight. Here, the effect of coupling itself can

Table 2. **Results:** We provide timings, domain/grid sizes, and oscillator statistics. Oscillator integration and wavesolver time-stepping are performed at the same rate. The CPU wavesolver corresponds to [Wang et al. 2018], whereas the GPU wavesolver is our novel implementation. Wavesolver domains are cubic. The Paddle Splash example fails on the CPU due to rapid, noisy fluid interface movement causing the simulation to crash. Likewise, the Forest Creek and Wave Crash examples run to completion on the CPU but exhibit popping artifacts (from under-resolved geometry) larger than the audio signal of interest.

					Oscillators			*Wavesolver*		*Time (hr)*	
	Example	*Length (s)*	*Dimensions (m)*	*Rate (kHz)*	*Total*	*Per Sec*	*Time (hr)*	*h (mm)*	*# cells*	*CPU*	*GPU*
VOF	2016 Pour	8.3	$0.08 \times 0.08 \times 0.24$	192	51598	6216	0.05	5.0	80^3	29.7	0.42
VOF	2016 Step	4.2	$0.08 \times 0.24 \times 0.24$	192	53357	12704	0.09	5.0	80^3	30.7	0.34
FLIP	Glass Pour	5.3	$0.08 \times 0.08 \times 0.2$	48	22591	4262	0.01	12.5	40^3	1.4	0.22
FLIP	Paddle Splash	1.9	$0.45 \times 0.66 \times 0.5$	96	11967	6283	0.20	10.0	80^3	–	0.61
FLIP	Forest Creek	6.0	$1.0 \times 0.5 \times 0.4$	48	133014	22169	1.78	12.5	120^3	42.4	3.12
FLIP	Wave Crash	4.5	$1.5 \times 0.75 \times 0.25$	48	306713	68158	0.91	12.5	120^3	31.6	1.54

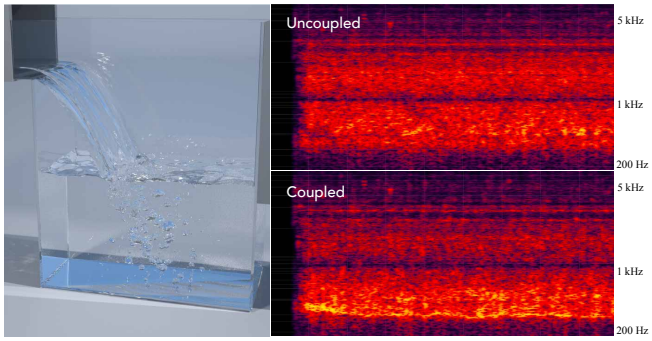


Fig. 9. **2016 Step**

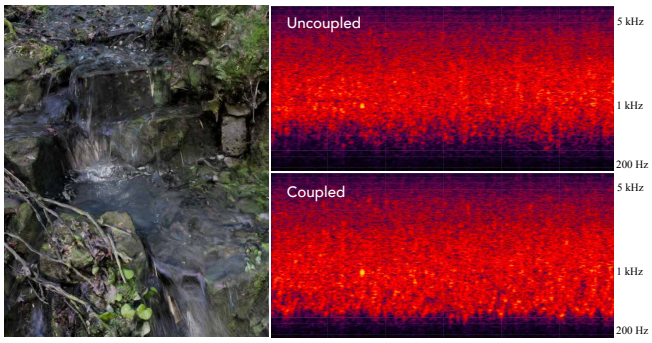


Fig. 10. **Forest Creek**

account for diverse pitch variations, producing rich sounds without needing to resolve non-spherical bubble natural frequencies.

Glass Pour: Shown previously in Figure 1, this example emulates the dimensions and setup of the 2016 Pour example while only taking 1.4 hours to run on the CPU wavesolver (compared to 29.7 hours for the 2016 Pour). We include a comparison with a real recording of a pouring glass in Figure 13 and show that our simulated audio exhibits matching spectral features – namely, the aforementioned bimodal spectra.

Forest Creek: Like the 2016 Step example, the Forest Creek example (shown in Figure 10) features relatively constant bubble generation rates. Key differences include a larger domain (1.0 m × 0.5 m for the Forest Creek, compared to 0.24 m × 0.08 m for the 2016 Step), as well as the most drastic deviation from planar water surface geometry among all of our examples.

Paddle Splash: Shown in Figure 11 (as well as previously in Figure 7), the Paddle Splash example features dynamic and abrupt geometry changes. Our GPU wavesolver can handle these without artifacts. Another benefit of FLIP simulators is the ease of handling dynamic geometry, which was difficult with the higher quality CFD VOF method [Popinet 2003].

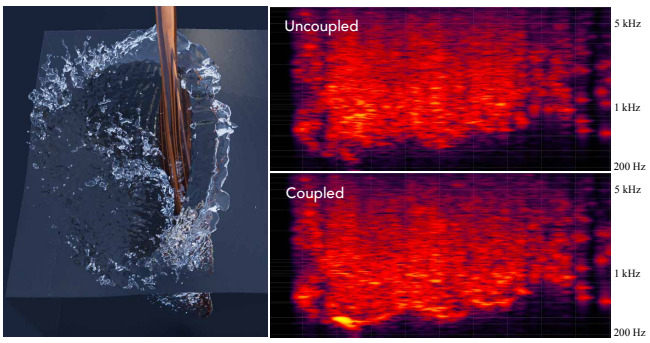


Fig. 11. **Paddle Splash**

Wave Crash: Shown in Figure 12, the largest of our examples is a breaking wave in a 1.5 m × 0.5 m domain. It generates over 300,000 bubbles in 4.5 seconds of simulation and produces the low-frequency (200 Hz) "roar" expected of a crashing wave.

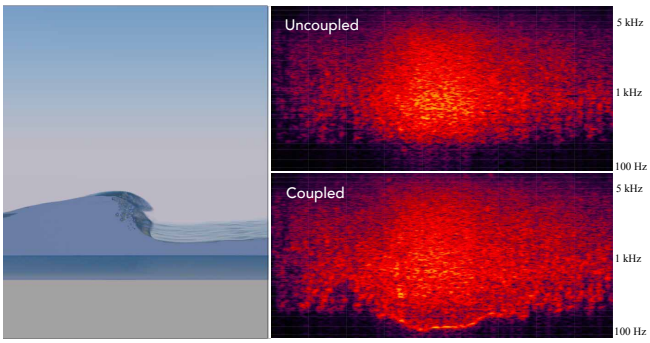


Fig. 12. **Wave Crash**

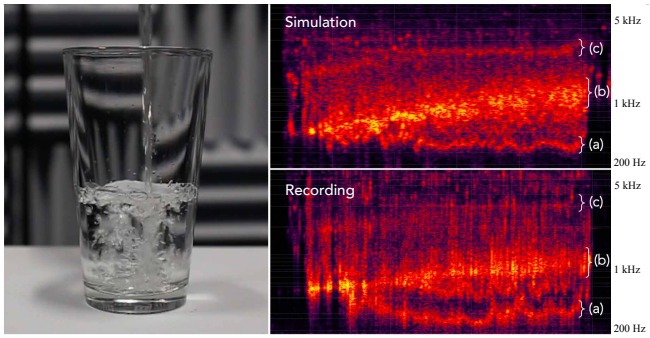


Fig. 13. **Glass Pour recording:** (Left) we use a Canon EOS 6D camera and Shure SM57 microphone to capture audiovisual footage of a pouring glass. (Right/Top) The simulated coupled audio for the Glass Pour example exhibits similar spectral features as the (Right/Bottom) recorded audio. Matching annotations for both spectrograms are provided which show (a) the falling pitch of bubbles coupling together, (b) the rising pitch accompanied by the rising water level, and (c) fixed container resonances.

7 CONCLUSION

We have shown several practical extensions of bubble-based sound synthesis models to capture richer low-frequency sound effects associated with coupled-bubble vibrations. Even the simple $1/d$ bubble coupling forces can produce perceptible sound effects which reduce the high-frequency character of uncoupled harmonic bubbles, which some have referred to as "harsh" or "digital sounding." Simulating many coupled bubbles is computationally expensive for both oscillator dynamics and time-domain radiation analysis, and we have proposed a number of numerical optimizations and details to make them possible for modern fluid animation techniques.

7.1 Limitations and Future Work

Despite these advances, there are many limitations and opportunities for future work. We achieve clearly audible transformations of frequency spectra even on simulation domains of modest size (roughly $\leq 1\ m^3$). However, for larger environments, our method may over-couple bubbles by not accounting for time delays, and because scene geometry is not accounted for by the monopole and dipole coupling models. Damping is also more complex for coupled systems, and may not be accurately predicted by damping models derived under the assumption of spherical mono-frequency bubbles and time delays. Few-bubble systems coupled together with time-delays have been studied, and analytical solutions for the two-bubble case are known (in which time delays are shown to increase damping), however, work remains to be done on investigating the effects of time delays on bubble sounds, and whether they can be accounted for in a stable and efficient manner.

Our coupling formulation assumes spherical bubbles. Non-spherical bubble coupling requires solving more complex equations for bubble frequencies and coupling pressures. Our procedural fluid examples do not include spatially varying single-bubble frequency effects, in part because they are expensive to compute, and they are perceptually masked by the emissions in the many-bubble examples, but they can be important in scenarios with low bubble counts.

Fast GPU FDTD methods for fluid sounds that use sequential time-stepping instead of sample-and-hold could provide faster sound synthesis and should be investigated (c.f. [Allen and Raghuvanshi 2015]). Achieving low-noise renderings in the presence of under-resolved interfaces with rapid shape and topology changes is a challenging problem. Our sample-and-hold GPU renderer is approximate, and it benefits from the noisy nature of bubbly flows which may not be suitable for other sound synthesis applications. Furthermore, while inspired by the parallel-in-time algorithm [Wang et al. 2018], the sample-and-hold solution no longer converges to the true solution due to the piecewise constant geometry. In addition, the sample-and-hold strategy sometimes creates small, static resonant cavities that trap the waves for too long, resulting in frequency artifacts. However, in practice, we find that the speed and robustness boost significantly outweigh the decay in quality.

Our initial renderings are monophonic, but stereo and spatialized sound should be explored and are important for appreciating noisy water sources [Verron et al. 2009]. Finally, we estimate airborne acoustic emissions, but renderings of underwater sounds are important to some applications. These will require further work on waterborne sound and compelling listening models.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for constructive feedback, Pu Zhang for early experiments (ca. 2015), Cole Sohn for Houdini assistance, and Jiayi Eris Zhang for proof reading. We thank Adobe and Meta for academic support, SideFX for donating Houdini licenses for academic research, Maxon Redshift, and Intel and XSEDE for compute resources. This work was completed no thanks to a pandemic, two child births, a skateboard mishap, and too many deadline power outages. This material is based upon work supported by the Department of Energy, National Nuclear Security Administration under Award Number DE-NA0003968; and by the National Science Foundation (HCC-0905506) and Graduate Research Fellowships (DGE-1656518 & DGE-1144153). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation or others.

REFERENCES

Andrew Allen and Nikunj Raghuvanshi. 2015. Aerophones in Flatland: Interactive wave simulation of wind instruments. *ACM Transactions on Graphics (TOG)* 34, 4

(2015), 1–11.
Karl Bolin and Mats Åbom. 2010. Air-borne sound generated by sea waves. *The Journal of the Acoustical Society of America* 127, 5 (2010), 2771–2779.
Quentin Bolsée and Vivian Bolsée. 2018. A Fast Water Droplet Sound Simulation. In *2018 International Conference on 3D Immersion (IC3D)*. 1–5. https://doi.org/10.1109/IC3D.2018.8657882
Landon Boyd and Robert Bridson. 2012. MultiFLIP for Energetic Two-Phase Fluid Simulation. *ACM Trans. Graph.* 31, 2, Article 16 (apr 2012), 12 pages. https://doi.org/10.1145/2159516.2159522
Sir William Henry Bragg. 1920. *The World of Sound.* G. Bell and Sons Ltd.
Robert Bridson. 2008. *Fluid simulation for computer graphics.* CRC Press.
Jeffrey N Chadwick, Steven S An, and Doug L James. 2009. Harmonic Shells: A practical nonlinear sound model for near-rigid thin shells. *ACM Transactions on Graphics* 28, 5 (2009), 1–119.
Helen Czerski. 2011. A candidate mechanism for exciting sound during bubble coalescence. *The Journal of the Acoustical Society of America* 129, 3 (2011), EL83–EL88.
Helen Czerski and Grant B Deane. 2010. Contributions to the acoustic excitation of bubbles released from a nozzle. *The Journal of the Acoustical Society of America* 128, 5 (2010), 2625–2634.
Helen Czerski and Grant B Deane. 2011. The effect of coupling on bubble fragmentation acoustics. *The Journal of the Acoustical Society of America* 129, 1 (2011), 74–84.
Fernando de Goes, Corentin Wallez, Jin Huang, Dmitry Pavlov, and Mathieu Desbrun. 2015. Power Particles: An Incompressible Fluid Solver Based on Power Diagrams. *ACM Trans. Graph.* 34, 4, Article 50 (jul 2015), 11 pages. https://doi.org/10.1145/2766901
Grant B Deane and Helen Czerski. 2008. A mechanism stimulating sound production from air bubbles released from a nozzle. *The Journal of the Acoustical Society of America* 123, 6 (2008), EL126–EL132.
Grant B Deane and Dale M Stokes. 2002. Scale dependence of bubble creation mechanisms in breaking waves. *Nature* 418 (2002), 839–844. https://doi.org/10.1038/nature00967
Grant B Deane and M Dale Stokes. 2010. Model calculations of the underwater noise of breaking waves and comparison with experiment. *The Journal of the Acoustical Society of America* 127, 6 (2010), 3394–3410.
Douglas Enright, Stephen Marschner, and Ronald Fedkiw. 2002. Animation and rendering of complex water surfaces. *ACM Transactions on Graphics (TOG)* 21, 3 (2002), 736–744.
Paul C Etter. 2018. *Underwater acoustic modeling and simulation.* CRC press.
Christopher Feuillade. 2001. Acoustically coupled gas bubbles in fluids: Time-domain phenomena. *The Journal of the Acoustical Society of America* 109, 6 (2001), 2606–2615.
Ryan Goldade, Mridul Aanjaneya, and Christopher Batty. 2020. Constraint Bubbles and Affine Regions: Reduced Fluid Models for Efficient Immersed Bubbles and Flexible Spatial Coarsening. *ACM Trans. Graph.* 39, 4, Article 43 (aug 2020), 15 pages. https://doi.org/10.1145/3386569.3392455
Shannon T Greenwood and Donald H House. 2004. Better with bubbles: enhancing the visual realism of simulated fluid. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation.* 287–296.
Henrik Wann Jensen, Stephen R Marschner, Marc Levoy, and Pat Hanrahan. 2001. A practical model for subsurface light transport. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques.* 511–518.
Chenfanfu Jiang, Craig Schroeder, Andrew Selle, Joseph Teran, and Alexey Stomakhin. 2015. The Affine Particle-in-Cell Method. *ACM Trans. Graph.* 34, 4, Article 51 (jul 2015), 10 pages. https://doi.org/10.1145/2766996
Byungmoon Kim. 2010. Multi-Phase Fluid Simulations Using Regional Level Sets. *ACM Trans. Graph.* 29, 6, Article 175 (dec 2010), 8 pages. https://doi.org/10.1145/1882261.1866197
Vern O Knudsen, RS Alford, and JW Emling. 1948. Underwater ambient noise. *J. mar. Res* 7, 3 (1948), 410–429.
Timothy R Langlois, Changxi Zheng, and Doug L James. 2016. Toward animating water with complex acoustic bubbles. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 1–13.
Timothy Leighton. 2012. *The Acoustic Bubble.* Academic Press.
Valentin Leroy, Nicolas Chastrette, Margaux Thieury, Olivier Lombard, and Arnaud Tourin. 2018. Acoustics of Bubble Arrays: Role Played by the Dipole Response of Bubbles. *Fluids* 3, 4 (2018). https://doi.org/10.3390/fluids3040095
V Leroy, M Devaud, T Hocquet, and J-C Bacri. 2005. The bubble cloud as an N-degree of freedom harmonic oscillator. *The European Physical Journal E* 17, 2 (2005), 189–198.
Wei Li, Yihui Ma, Xiaopei Liu, and Mathieu Desbrun. 2022. Efficient Kinetic Simulation of Two-Phase Flows. *ACM Trans. Graph.* 41, 4, Article 114 (jul 2022), 17 pages. https://doi.org/10.1145/3528223.3530132
N.Q. Lu, A. Prosperetti, and S.W. Yoon. 1990. Underwater noise emissions from bubble clouds. *IEEE Journal of Oceanic Engineering* 15, 4 (1990), 275–281. https://doi.org/10.1109/48.103521
Richard Manasseh and Andrew Ooi. 2009. Frequencies of acoustically interacting bubbles. *Bubble Science, Engineering & Technology* 1, 1-2 (2009), 58–74.
Steven L Means and Richard M Heitmeyer. 2001. Low-frequency sound generation by an individual open-ocean breaking wave. *The Journal of the Acoustical Society of America* 110, 2 (2001), 761–768.
Herman Medwin and Matthew M Beaky. 1989. Bubble sources of the Knudsen sea noise spectra. *The Journal of the Acoustical Society of America* 86, 3 (1989), 1124–1130.
Marcel Minnaert. 1933. XVI. On musical air-bubbles and the sounds of running water. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 16, 104 (1933), 235–248.
William Moss, Hengchin Yeh, Jeong-Mo Hong, Ming C. Lin, and Dinesh Manocha. 2010. Sounding Liquids: Automatic Sound Synthesis from Fluid Simulation. *ACM Trans. Graph.* 29, 3 (July 2010), 21:1–21:13. https://doi.org/10.1145/1805964.1805965
HN Oguz. 1994. A theoretical study of low-frequency oceanic ambient noise. *The Journal of the Acoustical Society of America* 95, 4 (1994), 1895–1912.
Stanley Osher and Ronald Fedkiw. 2006. *Level set methods and dynamic implicit surfaces.* Vol. 153. Springer Science & Business Media.
S. Popinet. 2003. Gerris: A Tree-Based Adaptive Solver for the Incompressible Euler Equations in Complex Geometries. *J. Comput. Phys.* 190 (2003), 572–600.
Lord Rayleigh. 1917. VIII. On the pressure developed in a liquid during the collapse of a spherical cavity. *Philosophical Magazine Series 6* 34, 200 (1917), 94–98.
Frédéric Risso and Jean Fabre. 1998. Oscillations and breakup of a bubble immersed in a turbulent field. *Journal of Fluid Mechanics* 372 (1998), 323–355. https://doi.org/10.1017/S0022112098002705
Hyuga Saito, Syuhei Sato, and Yoshinori Dobashi. 2021. A Liquid Sound Retrieval Using History of Velocities in Physically-Based Simulation. In *SIGGRAPH Asia 2021 Posters* (Tokyo, Japan) *(SA '21 Posters)*. Association for Computing Machinery, New York, NY, USA, Article 31, 2 pages. https://doi.org/10.1145/3476124.3488643
B. Solenthaler and R. Pajarola. 2008. Density Contrast SPH Interfaces. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Dublin, Ireland) *(SCA '08)*. Eurographics Association, Goslar, DEU, 211–218.
Jos Stam. 1999. Stable fluids. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques.* ACM Press/Addison-Wesley Publishing Co., 121–128.
M Strasberg. 1953. The pulsation frequency of nonspherical gas bubbles in liquids. *The Journal of the Acoustical Society of America* 25, 3 (1953), 536–537.
N. Thürey, F. Sadlo, S. Schirm, M. Müller-Fischer, and M. Gross. 2007. Real-Time Simulations of Bubbles and Foam within a Shallow Water Framework. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (San Diego, California) *(SCA '07)*. Eurographics Association, Goslar, DEU, 191–198.
Kees van den Doel. 2005. Physically based models for liquid sounds. *ACM Transactions on Applied Perception* 2, 4 (Oct. 2005), 534–546.
Charles Verron, Mitsuko Aramaki, Richard Kronland-Martinet, and Grégory Pallone. 2009. A 3-D immersive synthesizer for environmental sounds. *IEEE Transactions on Audio, Speech, and Language Processing* 18, 6 (2009), 1550–1561.
Jui-Hsien Wang, Ante Qu, Timothy R Langlois, and Doug L James. 2018. Toward wave-based sound synthesis for computer animation. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–16.
Kai Wang and Shiguang Liu. 2018. Example-based synthesis for sound of ocean waves caused by bubble dynamics. *Computer Animation and Virtual Worlds* 29, 3-4 (2018), e1835. https://doi.org/10.1002/cav.1835 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/cav.1835 e1835 cav.1835.
Changxi Zheng and Doug L. James. 2009. Harmonic Fluids. *ACM Trans. Graph.* 28, 3 (Aug. 2009), 37:1–37:12. http://www.cs.cornell.edu/projects/HarmonicFluids/
Changxi Zheng and Doug L James. 2011. Toward high-quality modal contact sound. In *ACM SIGGRAPH 2011 papers.* 1–12.
Yongning Zhu and Robert Bridson. 2005. Animating Sand as a Fluid. *ACM Trans. Graph.* 24, 3 (jul 2005), 965–972. https://doi.org/10.1145/1073204.1073298

A WORST-CASE ANALYSIS OF EIGEN-FREQUENCIES

To ensure stable time-stepping for close-proximity bubbles, it is useful to understand the highest frequencies the coupled system can produce in the worst case of overlap. Consider N bubbles, and let them have equal radii for simplicity so that $\mathbf{K} = \omega_0^2\mathbf{I}$. In that case, the generalized eigenvalue problem (assuming $\mathbf{y} = e^{i\omega t}\bar{\mathbf{y}}$) for their undamped frequencies is:

$$\mathbf{K}\bar{\mathbf{y}} = \omega^2\tilde{\mathbf{M}}\bar{\mathbf{y}} \quad \rightarrow \quad \bar{\mathbf{y}} = \frac{\omega^2}{\omega_0^2}\tilde{\mathbf{M}}\bar{\mathbf{y}} \quad \rightarrow \quad \bar{\mathbf{y}} = \lambda\tilde{\mathbf{M}}\bar{\mathbf{y}}, \tag{22}$$

where $\lambda = \omega^2/\omega_0^2$ is the normalized eigenvalue of $\mathbf{K}$ relative to $\tilde{\mathbf{M}}$. To better understand regularization, consider the worst-case scenario of N overlapping bubbles ($d_{ij} = 0$) so that $\tilde{\mathbf{M}}$ is (26). In this case, the

eigenvalues can be computed analytically (in Mathematica):

$$\lambda_{1\ldots(N-1)} = \frac{\varepsilon}{\varepsilon - 1} \qquad \text{and} \qquad \lambda_N = \frac{\varepsilon}{\varepsilon - 1 + N}. \tag{23}$$

Since the normalized frequency multiplier is $\frac{\omega}{\omega_0} = \sqrt{\lambda}$, the largest frequency that can be observed for a positive-mass regularized system ($\varepsilon > 1$) is

$$\omega_{max} = \omega_0 \sqrt{\frac{\varepsilon}{\varepsilon - 1}}, \tag{24}$$

which is independent of N, whereas the minimum is

$$\omega_{min} = \omega_0 \sqrt{\frac{\varepsilon}{\varepsilon - 1 + N}} \tag{25}$$

and shows lower frequencies for higher N in this fully overlapping case ($\omega_{min} \sim 1/\sqrt{N}$). Therefore, to limit the frequency increase factor ω_{max}/ω_0 to f, we can use $\varepsilon = \frac{f^2}{f^2-1}$, e.g., for $f = \sqrt{2}$ we get $\varepsilon = 2$.

B POSITIVITY OF REGULARIZED MASS MATRICES

We show that the regularized mass matrix $\tilde{\mathbf{M}}$ (13) can be made positive definite and sufficiently positive. Specifically, consider values of the regularization parameter ε such that $\hat{\mathbf{v}}^T \tilde{\mathbf{M}} \hat{\mathbf{v}} > m_0 > 0,\ \forall \hat{\mathbf{v}} \in \mathbb{R}^n$ with $\|\hat{\mathbf{v}}\|_2 = 1$, where $m_0 > 0$ is a small parameter that bounds the smallest eigenvalues of $\tilde{\mathbf{M}}$ away from zero. Unfortunately, the position dependence of $\tilde{\mathbf{M}}$ makes analysis difficult, and therefore we again consider the worst case where all bubbles overlap perfectly, so that $d_{ij} = 0$. In that case, the regularized mass matrix is

$$\tilde{\mathbf{M}}_{ij} = \delta_{ij} + (1 - \delta_{ij})/\varepsilon \quad \Leftrightarrow \quad \tilde{\mathbf{M}} = \varepsilon^{-1}\mathbf{1}\mathbf{1}^T + (1 - \varepsilon^{-1})\mathbf{I}, \tag{26}$$

where $\mathbf{1}$ is the vector of all ones, and

$$\hat{\mathbf{v}}^T \tilde{\mathbf{M}} \hat{\mathbf{v}} = \varepsilon^{-1}(\mathbf{1} \cdot \hat{\mathbf{v}})^2 + (1 - \varepsilon^{-1}). \tag{27}$$

Substituting (27) into $\hat{\mathbf{v}}^T \tilde{\mathbf{M}} \hat{\mathbf{v}} > m_0$, and solving for ε we obtain

$$\varepsilon > \frac{1 - (\mathbf{1} \cdot \hat{\mathbf{v}})^2}{1 - m_0}, \quad m_0 \in (0, 1), \tag{28}$$

then, since $\min_{\hat{\mathbf{v}}} (\mathbf{1} \cdot \hat{\mathbf{v}})^2 = 0$, we conclude that

$$\boxed{\varepsilon > \frac{1}{1 - m_0} > 0, \quad m_0 \in (0, 1).} \tag{29}$$

C FLIP STOCHASTIC BUBBLE MODEL

To augment FLIP simulations with bubble generation processes suitable for sound synthesis, we implement a modified version of the stochastic bubble seeding model originally presented in [Zheng and James 2009]. Our model also accounts for splitting and merging events.

Entrainment. We track mixing at the liquid-air interface by monitoring surface FLIP particles for rapid changes in depth. Given isosurface values ϕ for FLIP particles above a certain depth ϕ_{max}, we compute the difference at each timestep i as $\Delta\phi_i = \phi_i - \phi_{i-1}$. If $\Delta\phi_i$ exceeds some critical threshold ϵ, the particle becomes a seed and thus has some chance of entraining a bubble. Each seed has a time-to-live and entrainment strength parameter κ_e that determine the number of bubble creation attempts per timestep; we defer the reader to [Zheng and James 2009] for further details.

In order to entrain bubbles closer to the surface despite a coarse timestep resolution, we interpolate the isosurface values ϕ, such that the bubbles are created exactly where the difference in ϕ *equals* ϵ. We opt for linear interpolation and assume constant velocity over each timestep, so the entrainment positions x_e are given by

$$x_e = (1 - \alpha)x_{i-1} + \alpha x_i, \quad \alpha = \frac{\epsilon}{\phi_i - \phi_{i-1}} \tag{30}$$

At creation time, bubble radii are sampled from a statistical distribution. In our examples, we use statistics from the two-phase volume of fluid simulations from [Langlois et al. 2016] (Faucet Pour and Water Step), as well as the empirical power law distribution ([Deane and Stokes 2002]). Finally, if a bubble is to be entrained on top of existing surface bubbles, we collapse them to avoid intersections.

Splitting. A bubble is likely to fragment if turbulent forces across the bubble exceed the restoring force of surface tension. The Weber number provides a measure of the ratio between these two forces and can be expressed as

$$We = \rho \, \overline{\delta u^2(d)} \, d/\sigma \tag{31}$$

where $\overline{\delta u^2(d)}$ denotes the mean-square velocity difference over the bubble's diameter [Risso and Fabre 1998], and σ (≈ 0.0726) is the surface tension. Splitting occurs when We exceeds some critical threshold. Unfortunately, in practice, the FLIP velocity field can be noisy and under-resolved, which can lead to excessive splitting in only a select few locations. Instead, we adopt a probabilistic model, where the probability of splitting is directly proportional to the bubble diameter d and is given by

$$P(split) = \kappa_m (d - 2r_{min}) \tag{32}$$

where κ_m is the split strength parameter, and r_{min} is the minimum splitting radius. We use the $\overline{\delta u^2(d)}$ term to set a threshold, so that bubbles only split if they are in sufficiently turbulent flows (but we do not let this term dominate the splitting mechanism). After splitting, the new bubble volumes are divided randomly such that overall volume is conserved, and each individual bubble's volume is greater than $(2/3)\pi r_{min}^3$.

Merging. We merge bubbles if they are sufficiently close. Given two bubbles with positions x_i and x_j, radii r_i and r_j, as well as merge strength parameter κ_m, we merge if $|x_i - x_j| < \kappa_m(r_i + r_j)$. To prevent the creation of nonphysically large bubbles, we clamp the merged bubble volume at $(4/3)\pi r_{max}^3$. For bubble pairs where $\kappa_m(r_i + r_j) < |x_i - x_j| < (r_i + r_j)$, we resolve collisions separately.

We simulate bubbles at timestep rates comparable to our FLIP simulations. To avoid discretization artifacts in the audio, we randomly offset the bubble start and end times, ensuring that split and merge events remain coherent. Without robust collision detection, bubbles may intersect in between frames, so regularizing the mass matrix is important when synthesizing sounds for FLIP examples as well (despite all bubbles being spherical).