Aphros: High Performance Software for Multiphase Flows with Large Scale Bubble and Drop Clusters

Petr Karnakov
Computational Science and Engineering Laboratory
ETH Zürich
Switzerland

Fabian Wermelinger
Computational Science and Engineering Laboratory
ETH Zürich
Switzerland

Sergey Litvinov
Computational Science and Engineering Laboratory
ETH Zürich
Switzerland

Petros Koumoutsakos*
Computational Science and Engineering Laboratory
ETH Zürich
Switzerland

ABSTRACT

We present the high performance implementation of a new algorithm for simulating multiphase flows with bubbles and drops that do not coalesce. The algorithm is more efficient than the standard multi-marker volume-of-fluid method since the number of required fields does not depend on the number of bubbles. The capabilities of our methods are demonstrated on simulations of a foaming waterfall where we analyze the effects of coalescence prevention on the bubble size distribution and show how rising bubbles cluster up as foam on the water surface. Our open-source implementation enables high throughput simulations of multiphase flow, supports distributed as well as hybrid execution modes and scales efficiently on large compute systems.

CCS CONCEPTS

• **Computing methodologies** → **Modeling methodologies**; *Massively parallel and high-performance simulations.*

KEYWORDS

multiphase flows, foam, coalescence prevention, volume of fluid, high performance computing

ACM Reference Format:
Petr Karnakov, Fabian Wermelinger, Sergey Litvinov, and Petros Koumoutsakos. 2020. *Aphros*: High Performance Software for Multiphase Flows with Large Scale Bubble and Drop Clusters. In *Proceedings of the Platform for Advanced Scientific Computing Conference (PASC '20).* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3394277.3401856

*Corresponding author: petros@ethz.ch

PASC '20, June 2020, Geneva, Switzerland

ACM ISBN 9781450379939…$15.00
https://doi.org/10.1145/3394277.3401856

1 INTRODUCTION

Many industrial processes and natural phenomena involve multiphase flows with large clusters of bubbles and drops. The presence and size of these clusters depends on the process of bubble coalescence. When two bubbles or drops collide, the liquid film between them gets thinner and finally breaks, leading to coalescence. However, the presence of surfactants [11] or other impurities, such as electrolytes [7], in the liquid, can delay or prevent coalescence. Collisions of bubbles without coalescence are also observed in clean liquids free of surfactants if the film drainage time is sufficiently large [4].

Three factors contribute to the inhibition of coalescence: surfactants, viscosity and the Marangoni flow [26]. Surfactants are chemical compounds that tend to accumulate at the interface and alter its properties. In particular, they make the interface elastic and therefore reduce its sensitivity to perturbations. Viscosity of liquids creates shear stresses and increases the time of the film drainage. Finally, the Marangoni flow occurs in the direction of the surface tension gradients and brings additional liquid to the film. Such gradients are caused by a falling concentration of gas (e.g. due to diffusion into bubbles) or differences in temperature. These mechanisms explain how a very thin film can be stable.

The modeling and simulation of inhibition of coalescence presents a number of numerical challenges. The liquid film between bubbles can be arbitrarily thin, while the spatial scales resolved by numerical simulations are finite. Front-capturing and front-tracking methods are two classes of numerical methods commonly used for flows with interfaces. These methods have different capabilities in relation to coalescence. Front-capturing methods, such as the volume-of-fluid [22] or level-set methods [24], represent the interface on a fixed grid, and one computational cell can not contain more than one interface. Therefore, bubbles and drops always coalesce at distances shorter than one cell. Opposite to them are front tracking methods [25] that represent the interface with connected particles. They allow multiple interfaces in one cell but require explicit re-connection of particles to describe breakup and coalescence.

Front capturing methods remain the main tool for simulations of flows with bubbles and drops [12]. Consequently, there have been developments to extend these methods to cases of non-coalescing bubbles. One approach is to represent bubbles with separate marker functions, volume fractions [6] or level-sets [3, 18], and solve the

advection equation independently for each field. However, this becomes computationally expensive for systems with many bubbles. Another approach uses a single level-set function and applies a Voronoi reconstruction near the interface to advance the level-set function [20].

We propose an algorithm that implements the multi-marker volume-of-fluid method [6] but the number of required marker functions does not depend on the number of bubbles. This results in a particularly computationally efficient implementation for multiphase flow simulations.

The paper is organized as follows. Section 2 describes the algorithm and the corresponding model for multiphase flows with surface tension. We discuss aspects of the high-performance computing (HPC) implementation of our flow solver Aphros [1] in Section 3 and present verification tests in Section 4. Finally, in Section 5 we demonstrate the capabilities of our algorithm on simulations of a foaming waterfall which involves the phenomena of air entrainment, bubble breakup and foam formation. The source code of our solver, documentation and examples of simulation setups are available online[1].

2 NUMERICAL MODEL

2.1 Flow equations

A two-component incompressible flow is described by the Navier-Stokes equations for the mixture velocity $\boldsymbol{u}$ and pressure p

$$\nabla \cdot \boldsymbol{u} = 0\,, \tag{1}$$

$$\rho\Big(\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u}\cdot\nabla)\,\boldsymbol{u}\Big) = -\nabla p + \nabla\cdot\mu(\nabla\boldsymbol{u} + \nabla\boldsymbol{u}^T) + \boldsymbol{f}_\sigma + \rho\boldsymbol{g} \tag{2}$$

and the advection equation for the volume fraction α

$$\frac{\partial \alpha}{\partial t} + (\boldsymbol{u}\cdot\nabla)\,\alpha = 0\,, \tag{3}$$

with the mixture density $\rho = (1-\alpha)\rho_1 + \alpha\rho_2$, dynamic viscosity $\mu = (1-\alpha)\mu_1 + \alpha\mu_2$ and gravitational acceleration $\boldsymbol{g}$. The surface tension force is defined as $\boldsymbol{f}_\sigma = \sigma\kappa\nabla\alpha$ with the surface tension coefficient σ and interface curvature κ.

We use a finite volume discretization based on Chorin's projection method [5] with the second order backward discretization in time and iterative corrections. The advection equation is solved using the VOF method with piecewise linear PLIC reconstruction [23] and the fluxes computed using the conservative split technique [28]. As required for the surface tension term, we compute the interface curvature using the method of particles [15].

2.2 Coalescence prevention

Standard formulations of the volume-of-fluid method are unable to describe situations where one cell contains more than one interface. Therefore, if two bubbles approach each other at a distance below one computational cell, they always coalesce. One standard technique to treat multiple interfaces in the same cell is the multi-marker volume-of-fluid method [6]. The method places each object in a separate volume fraction field and solves the advection equation

[1] Aphros: https://github.com/cselab/aphros

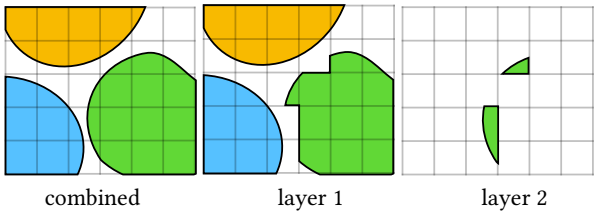

Figure 1: Multiple layers with color functions. Combined field can describe overlapping interfaces, while each layer has at most one bubble per cell.

for each field independently

$$\frac{\partial \alpha^l}{\partial t} + (\boldsymbol{u}\cdot\nabla)\,\alpha^l = 0\,, \tag{4}$$

where $l = 1\ldots L$ and L is the number of volume fraction fields. Each object (e.g. bubble or droplet) then belongs to a separate volume fraction field such that $L = N_{\text{bubbles}}$. While this method prevents coalescence of bubbles, each bubble requires its own volume fraction field. Therefore, describing systems with many bubbles using this method is computationally expensive as the cost of this method scales as $O(N_{\text{bubbles}}\ N_{\text{cells}})$.

On a structured grid, the discretization of these advection equations is based on the standard PLIC technique [2, 23]. First, the normal vectors $\boldsymbol{n}^l$ are estimated from the volume fraction field α^l. Then, the fluid volume in each cell is reconstructed by a polyhedron, and the fluxes are calculated by deforming the polyhedron according to the given velocity field $\boldsymbol{u}$. One time step of this discretization can be schematically written in terms of discrete operators $\mathcal{A}$ and $\mathcal{N}$

$$\alpha_i^{l,\text{new}} = \mathcal{A}\big(S_i[\alpha^l], S_i[\boldsymbol{n}^l]\big)\,, \tag{5}$$

$$\boldsymbol{n}_i^{l,\text{new}} = \mathcal{N}\big(S_i[\alpha^l]\big)\,, \tag{6}$$

where $i \in \Omega$ is the index of cells in the computational domain Ω, S_i is indices of cells in the $3\times3\times3$ stencil centered at i and $S_i[\phi]$ is a list of values of a field ϕ in stencil S_i

$$S_i[\phi] = \big[\phi_j,\quad j \in S_i\big]. \tag{7}$$

We propose an algorithm that implements the multi-marker volume-of-fluid method but has the computational cost that does not depend on the number of bubbles in the simulation. In a system consisting of many bubbles, each volume fraction field (4) would contain only one bubble and would be empty away from it. This indicates that keeping track of all volume fraction fields is redundant. Instead, we assign each bubble with a unique *color* $q \in \mathbb{R}$ and store the colors of all bubbles in discrete fields $q^l : \Omega \to \mathbb{R}$. Fields α^l store the corresponding volume fractions. The pairs (α^l, q^l) are referred to as *layers*, and the combined field (α, q) can store multiple values in the same cell distinguished by colors. We modify the discretization of the advection equations (5) to take into account the color functions. The discrete operators are now applied to stencils assembled locally for a given pair of cell index and color

$$\alpha_i^{l,\text{new}} = \mathcal{A}\big(\hat{S}_i[\alpha^l, q_i^l], \hat{S}_i[\boldsymbol{n}^l, q_i^l]\big)\,, \tag{8}$$

$$\boldsymbol{n}_i^{l,\text{new}} = \mathcal{N}\big(\hat{S}_i[\alpha^l, q_i^l]\big)\,, \tag{9}$$

where $\hat{S}_i[\phi, q]$ selects values of a field ϕ from the neighboring cells S_i and all layers having the same color q

$$\hat{S}_i[\phi, q] = \left[\phi_j^l, \quad j \in S_i \text{ and } l \,:\, q_j^l = q\right]. \tag{10}$$

The corresponding surface tension force is defined from all layers as

$$f_\sigma = \sum_{l=1}^{L} \sigma \kappa^l \nabla \alpha^l \,, \tag{11}$$

where κ_i^l is the interface curvature computed from cells of color q_i^l. Figure 1 illustrates how the volume fraction field is represented on fields with multiple layers. For the number of layers we choose $L = 4$ as justified in Section 4.1.

Layers	Cubism Standard	Cubism Halo	Speedup
1	9.04 s	6.02 s	1.50
2	11.93 s	7.23 s	1.65
3	14.29 s	8.16 s	1.75
4	16.64 s	8.92 s	1.87

Table 1: Time to solution for different number of color function layers in the advection step. The column denoted as "Cubism Standard" corresponds to the case when the library owns the memory of the computational grid. The column denoted by "Cubism Halo" corresponds to halo coordination only. The work presented in [16] belongs to the "Cubism Standard" case.

3 HIGH PERFORMANCE IMPLEMENTATION

The software design of the present solver is described in [16]. In this section we address shortcomings that have been outlined in the previous work and present improvements on the software regarding halo cell synchronization and compute-transfer overlap.

The test case used for the benchmark results presented here consists of initially cubic cavities, homogeneously distributed in the computational domain. The amount of cavities increases proportionally with the resolution and corresponds to the case presented in Section 5 at comparable resolution. The simulation is advected for ten time steps using a sinusoidal velocity field and periodic boundary conditions. The time to solution is defined by the accumulated time to perform the time steps. In the case of multiple MPI ranks the reported time is the average over all ranks. The benchmark does not take into account I/O or in-situ post-processing operations. The data has been collected on the Cray XC50 compute nodes of the Piz Daint supercomputer at the Swiss National Supercomputing Center (CSCS) using the GNU 8.3.0 compiler tools.

3.1 Zero-copy exchange of halo cells

The memory for the storage of structured computational data is maintained by the structured grid library Cubism [8, 27]. Memory is allocated in contiguous chunks for the full computational domain. Chunks are further divided into compute blocks which represent the smallest entity that can be queried by the user to perform operations on [16]. Temporary copies of compute blocks are created when the user requests a block for stencil operation, where the number of halo cells is defined in a stencil object that allows for querying halos on faces, edges and corners of the block. Halo cells at irregular locations are either obtained from an asynchronously managed MPI communication buffer or boundary conditions.

The present solver manages its own memory and relies on the Cubism library for updating local and global halo cells. The exclusive ownership of memory to the flow solver implies a performance penalty due to excessive copy overhead for halo cell synchronization. Communication intensive operations like the advection algorithm presented in this work are especially affected by this overhead. To address this bottleneck, we have modified the Cubism library such that it can be used as a standalone tool for halo cell synchronization only. The modified library operates on view objects that are wrapped around externally managed data that models a computational block. The only requirement on computational blocks is contiguous memory layout and that iteration over the data by a pointer to its underlying type is defined. Individual blocks may not necessarily be next to each other in memory. Table 1 shows the performance gain of the advection step using the modified library for a single compute node with 12 MPI ranks mapped to individual cores. Arbitrary numbers of fields subject to communication are organized in field containers and processed in batches. Communication requests with many fields, as it is the case for the advection algorithm, improve the efficiency of communication buffer packing internal to Cubism library as well as network utilization due to larger message sizes.

3.2 MPI+OpenMP execution model

The flow solver presented in [16] is limited to synchronous exchange of halo cells. In order to perform meaningful work during the communication of data, the halo exchange during advection has been split into an asynchronous send initialization and a message synchronization step in user code. The first step posts asynchronous send and receive requests for blocks on subdomain boundaries. After message posting, a list of internal blocks not subject to communication is returned to the caller for immediate processing. The total time for advection and the total time spent in message synchronization is shown in Figure 2. The measurements correspond to the accumulated time over all time steps where time spent in message synchronization is one order of magnitude lower than the total time spent in computation.

To reduce the surface-to-volume ratio of compute blocks on a given node, a multi-threaded MPI+OpenMP execution model has been implemented in addition to pure MPI. The multi-threaded model maps MPI ranks to compute nodes instead of cores as in the case of pure MPI. Compute blocks are mapped at a granularity of one block per thread using a dynamic scheduling policy. Our hybrid model requires two additional MPI communicators to embed the MPI only Hypre [14] library. Hypre is needed to obtain the solution of the pressure p after the flow field has been advected. Average time measurements for the hybrid execution model are shown in Figure 2 in addition to the pure MPI model. The hybrid advection scheme is currently slower compared to pure MPI because management of contending resources in multi-threaded execution has not been optimized yet. The reduction of the surface-to-volume ratio in hybrid execution reduces the overall time spent in MPI synchronization which can be beneficial for compute kernels with very large messages.

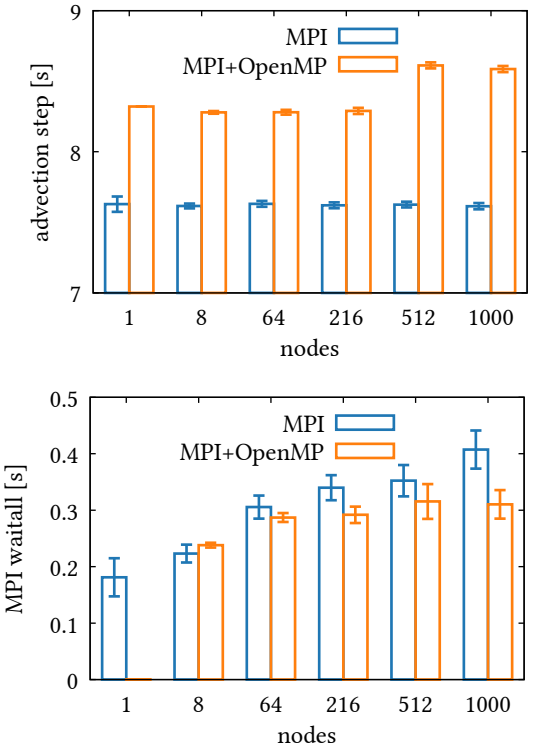

Figure 2: Average time for the pure MPI and hybrid MPI+OpenMP execution models. Time for advection with four marker layers (top) and time spent in MPI synchronization (bottom).

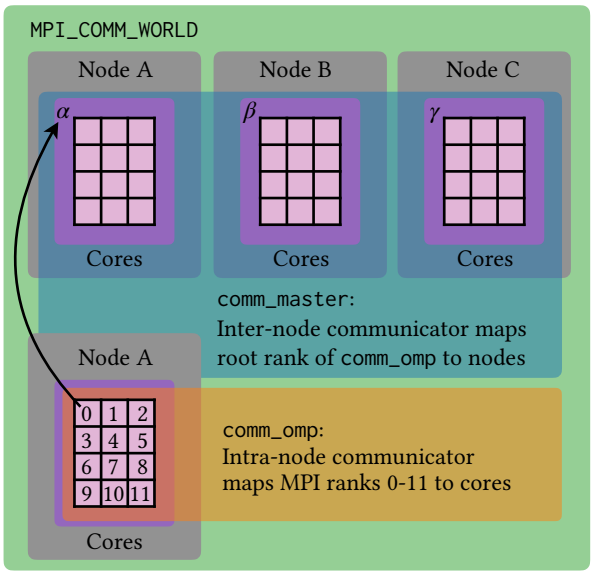

Figure 3: MPI communicator splitting used in our hybrid execution model.

The additional communicators in the hybrid model are derived based on the host (and optionally NUMA) affinity of each rank.

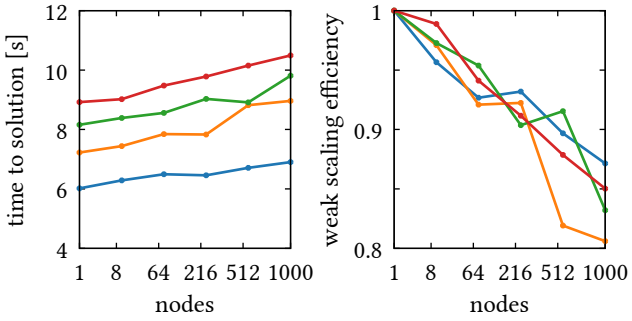

Figure 4: Time to solution (left) and weak scaling efficiency (right) for the pure MPI execution model of our advection algorithm depending on the number of layers $L = 1$ ——, 2 ——, 3 —— and 4 ——. The data is obtained from the Cray XC50 compute nodes on Piz Daint.

We split the world communicator into a node local `comm_omp` communicator, where the root ranks in `comm_omp` are further joined in a group used to create `comm_master` for communication among nodes. Figure 3 illustrates the communicator splitting used in our hybrid model. The example shows three nodes A, B and C. At program start, the ranks in `MPI_COMM_WORLD` are mapped to cores. The world communicator is then split into `comm_omp` and `comm_master` based on the rank affinity described above. The `comm_master` communicator is obtained from the root ranks in `comm_omp`, that is, α, β and γ for nodes A, B and C in Figure 3, respectively. Communicator `comm_master` is the main communicator that is used for halo communication among nodes. Ranks associated to this communicator further control the spawning and joining of OpenMP threads. The spawned threads are assigned the same affinity that corresponds to ranks in the `comm_omp` communicator. For multi-threaded execution, the idle ranks in `comm_omp` are set to sleep which triggers a context switch to activate the contending OpenMP threads. Calls to the Hypre library are performed on the world communicator where matrix assembly prior to the call is carried out on the `comm_omp` communicator using node local data. Compute blocks on a node are allocated by the ranks in `comm_master` and arranged on a virtual Cartesian topology. Ranks in `comm_omp` work on a contiguous subset of the Cartesian block grid during matrix assembly to ensure efficient Hypre collaboration. The same subset of blocks is used for other compute kernels executed in the multi-threaded environment. After matrix assembly, the spawned threads are joined and sleeping ranks are assigned back their resources to execute the Hypre routines on the world communicator.

3.3 Scaling and memory requirements

The time to solution and weak scaling efficiency for the pure MPI execution model of our multi-layer advection algorithm is shown in Figure 4. The algorithm exhibits efficient scaling for large node counts and is consistent among increasing numbers of marker layers. Our hybrid MPI+OpenMP execution model exhibits identical scaling. The memory footprint scales linearly with the number of marker layers. Each layer requires 96 byte of double precision data per cell. A typical problem with a mesh size of 384^3 cells requires about 5 GiB of memory for each layer.

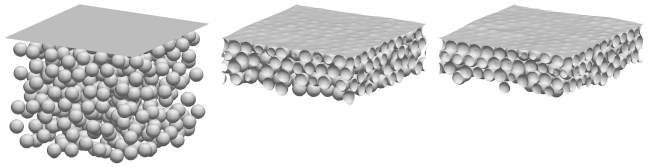

Figure 5: Rising bubbles. Snapshots of the interfaces at t = 0, 5 and 10. Bubbles cluster up on the surface.

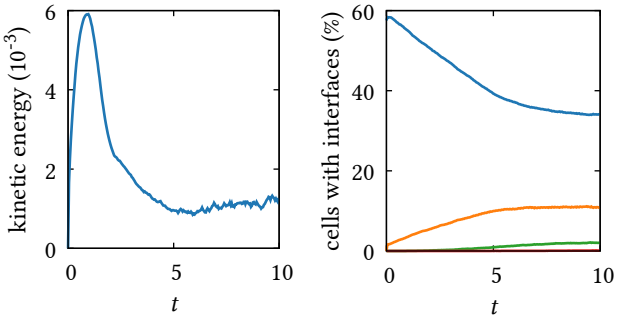

Figure 6: Rising bubbles. Kinetic energy of the mixture (left) and the percentage of cells containing 1 ——, 2 ——, 3 —— and 4 —— interfaces (right).

4 VALIDATION

4.1 Rising bubbles

The first test demonstrates how overlapping interfaces are distributed over multiple layers. The problem is solved in the cubic domain $[0, 1]^3$ on a mesh of 64^3 cells with periodic boundary conditions in the $x-$ and z-directions and free-slip walls in the y-direction. The initial velocity is zero, and the volume fraction field represents a layer of gas $0.8 < y < 1$ together with 397 bubbles of radius 0.05 placed uniformly over the remaining volume. Parameters of the problem are density $\rho_1 = 1$ and $\rho_2 = 0.01$, viscosity $\mu_1 = 0.01$ and $\mu_2 = 0.0001$, gravitational acceleration $g = 5$ and surface tension coefficient $\sigma = 0.1$. The simulation is performed with $L = 8$ layers.

Snapshots of the interfaces are shown in Figure 5. The bubbles rise and cluster up on the surface, which deforms under the pressure from bubbles. The initial separation between bubbles is sufficiently large such that the interfaces do not overlap and all fit in one layer. Cells with multiple interfaces appear as the bubbles move and the distances between them reduce. Figure 6 shows the fraction of cells containing multiple interfaces. The volume fraction therefore distributes over multiple layers. However, the total gas volume $V_{\text{gas}}(t) = \sum_{l=1}^{L} \int \alpha^l(\boldsymbol{x}, t)\, dV$ is conserved within 10^{-4} relative to its initial value. The kinetic energy reduces to about 20% of its maximum reached at $t \approx 1$. This residual flow is attributed to the spurious flow inherent to the surface tension model. The evolution of the kinetic energy suggests that the bubbles equilibrate near the surface. Since four layers are sufficient to represent all overlapping interfaces in this case, we use $L = 4$ layers in other simulations.

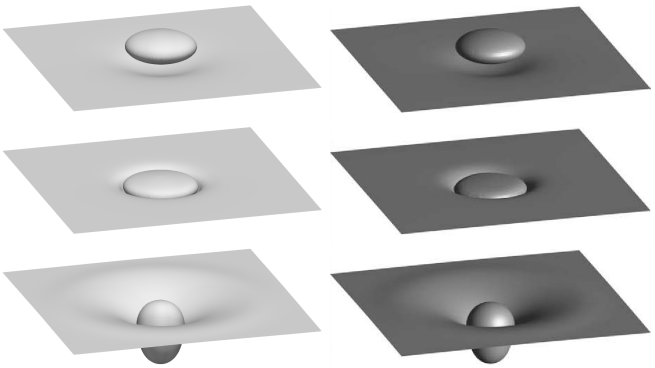

Figure 7: Drop impact on liquid-liquid interface. Snapshots of the interfaces at t = 0.63, 0.67, and 0.77 s produced by the present method (left) compared to images from numerical study [6] (right). Reproduced from [6], with permission from Elsevier.

4.2 Drop impact on liquid-liquid interface

The second test validates the model on the case of the gravity-driven impact of a liquid drop onto a liquid-liquid interface. The problem is solved in a rectangular domain of size $5 \times 10 \times 5$ cm on a mesh containing $160 \times 320 \times 160$ cells with no-slip walls in the $y-$direction and periodic conditions in the other directions. Parameters of the problem are taken from numerical study [6] based on experiment [19] (Combination 1): density $\rho_1 = 949$ and $\rho_2 = 1128$ kg/m^3, viscosity $\mu_1 = 0.019$ and $\mu_2 = 0.0063$ Pa · s, gravitational acceleration $g = 9.8$ m/s^2 and surface tension coefficient $\sigma = 0.029$ N/m. They correspond to a water+glycerin drop falling in silicon oil. A spherical drop of a radius 5.1 mm is initially placed at a distance of 67 mm between its center and the interface.

Snapshots of the interfaces and the corresponding slices are shown in Figures 7 and 8 compared to experimental [19] and numerical data [6]. The falling drop creates a bulge as it approaches the interface. Then the liquid film between the liquids drains but no coalescence occurs, and the liquid drop rests on the interface. Our algorithm produces the same results as method [6], but it is less computationally expensive for systems with many bubbles or drops.

5 APPLICATION TO FOAMING WATERFALL

We apply our software to simulations of a foaming waterfall and consider two cases: with coalescence of bubbles using the standard VOF method and without coalescence of bubbles using the algorithm for coalescence prevention described in Section 2.2. This application combines various physical processes and serves to demonstrate the capabilities of our method.

5.1 Simulation setup

The problem is solved in a rectangular domain of size $2H \times H \times H$ with $H = 10$ cm. The boundary conditions are illustrated to scale in Figure 9 and include inlet, outlet, free-slip walls and periodic conditions. The initial velocity is zero, and the volume fraction

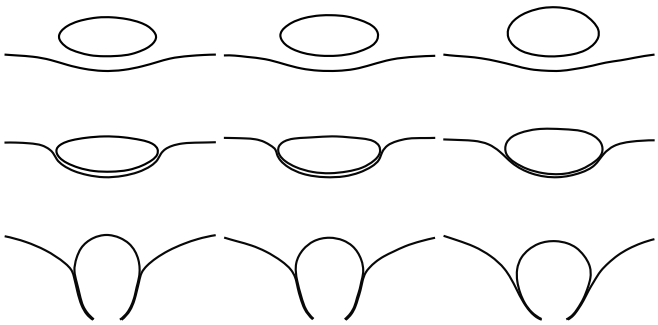

Figure 8: Drop impact on liquid-liquid interface. Slices of the interfaces at $t = 0.63,\ 0.67,$ and 0.77 s produced by the present method (left) compared to images from numerical study [6] (center) and experimental data [19] (right).

represents the water surface in the middle of the domain. Parameters of the problem are density $\rho_1 = 1000$ and $\rho_2 = 10$ kg/m^3, viscosity $\mu_1 = 10^{-3}$ and $\mu_2 = 10^{-5}$ Pa $\cdot$ s, gravitational acceleration $g = 9.8$ m/s^2 and surface tension coefficient $\sigma = 0.072$ N/m. The thickness of the waterfall is 5 mm and the inlet velocity is 1.5 m/s. The dimensionless time is defined as $t^* = t/\sqrt{H/g}$ relative to time unit $\sqrt{H/g} \approx 0.1$ s. Unless stated otherwise, all statistics and visualizations are obtained from simulations on a mesh of $768 \times 384 \times 384$ cells. Selected statistics are compared to results on coarser meshes of $384 \times 192 \times 192$ or $192 \times 96 \times 96$ cells.

Simulation with coalescence prevention on the finest mesh of $768 \times 384 \times 384$ cells took 20 hours to reach time $t^* = 12$ on 1152 compute nodes of Piz Daint supercomputer equipped with 12-core CPU Intel® Xeon® E5-2690 v3 processors.

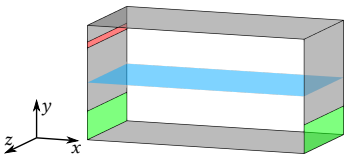

Figure 9: Initial position of water surface (blue) and boundary conditions: inlet (red), outlet (green), free-slip walls (gray) and periodic on other boundaries.

5.2 Air entrainment

The development of a waterfall involves various phenomena: entrainment of air by a sheet of water, breakup and deformation of bubbles and, depending on the model, either coalescence or clustering of bubbles. The mechanisms of air entrainment in breaking waves are discussed in [17] using plunging jets as a model example. Four mechanisms are distinguished: entrapment of a tube of air when the plunging jet hits the front face of the wave, entrainment around the jet impact site as the jet drags air into the water, entrainment by impacts of subsequent splashes created by the primary jet impact and the leading-edge entrainment of the turbulent breaking region at later stages of the breaking process.

In our simulations we find two major mechanisms of air entrainment. Three-dimensional visualizations of the waterfall evolution are shown in Figures 11 and 12, while Figure 10 shows the volume fraction field averaged in the z-direction. The behavior at early stages is the same for both cases with and without coalescence of bubbles: a sheet of water impacts the water surface and creates a tube of air leading to air entrainment. This corresponds to one major mechanism of air entrainment in breaking waves discussed in [17]. Once the air tube disappears by $t^* = 6$, we observe another mechanism of air entrainment: the plunging jet (or sheet) of water drags air into the water.

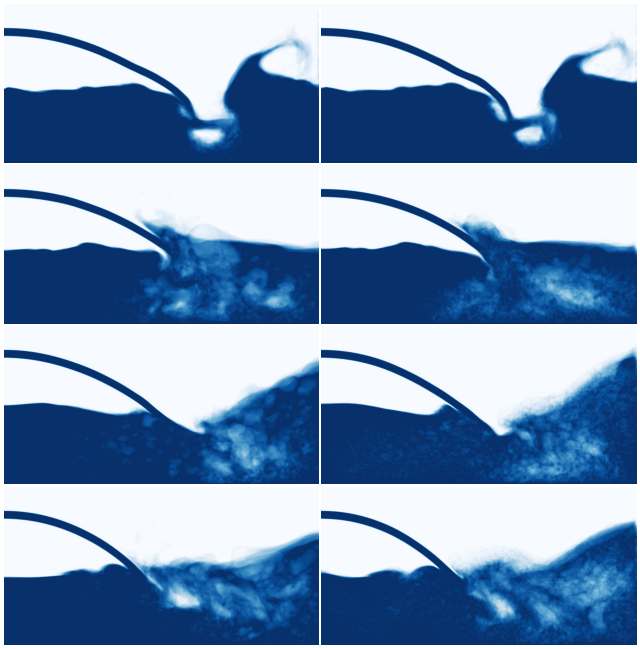

Figure 10: Volume fraction averaged in the z-direction: with (left) and without coalescence (right) at $t^* = 2.5,\ 6,\ 10$ and 12.

5.3 Bubble size distribution

One important characteristic of the air entrainment and bubble generation is the bubble size distribution. Theoretical model [13] suggests the scaling law for the distribution of the bubble radius

$$N(r) \propto r^{-10/3}, \tag{12}$$

where $N(r)dr$ is the number of bubbles of radii between r and $r+dr$. The model stems from the assumption that the inflow of air per unit volume is constant, and the number of bubbles depends only on the turbulent dissipation rate and the bubble radius. This scaling law is commonly observed for bubbles generated by breaking waves and has been reported in experimental [9] and numerical [10] studies.

Figure 13 shows the time-averaged distribution of the bubble size at various resolutions. In both cases with or without coalescence of bubbles, we recover the scaling law (12). Prevention of coalescence only increases the number of bubbles and does not have a major effect on distribution of the bubble size. One explanation for this behavior is that the size of bubbles is determined by the bubble breakup near the waterfall impact zone and not affected by coalescence.

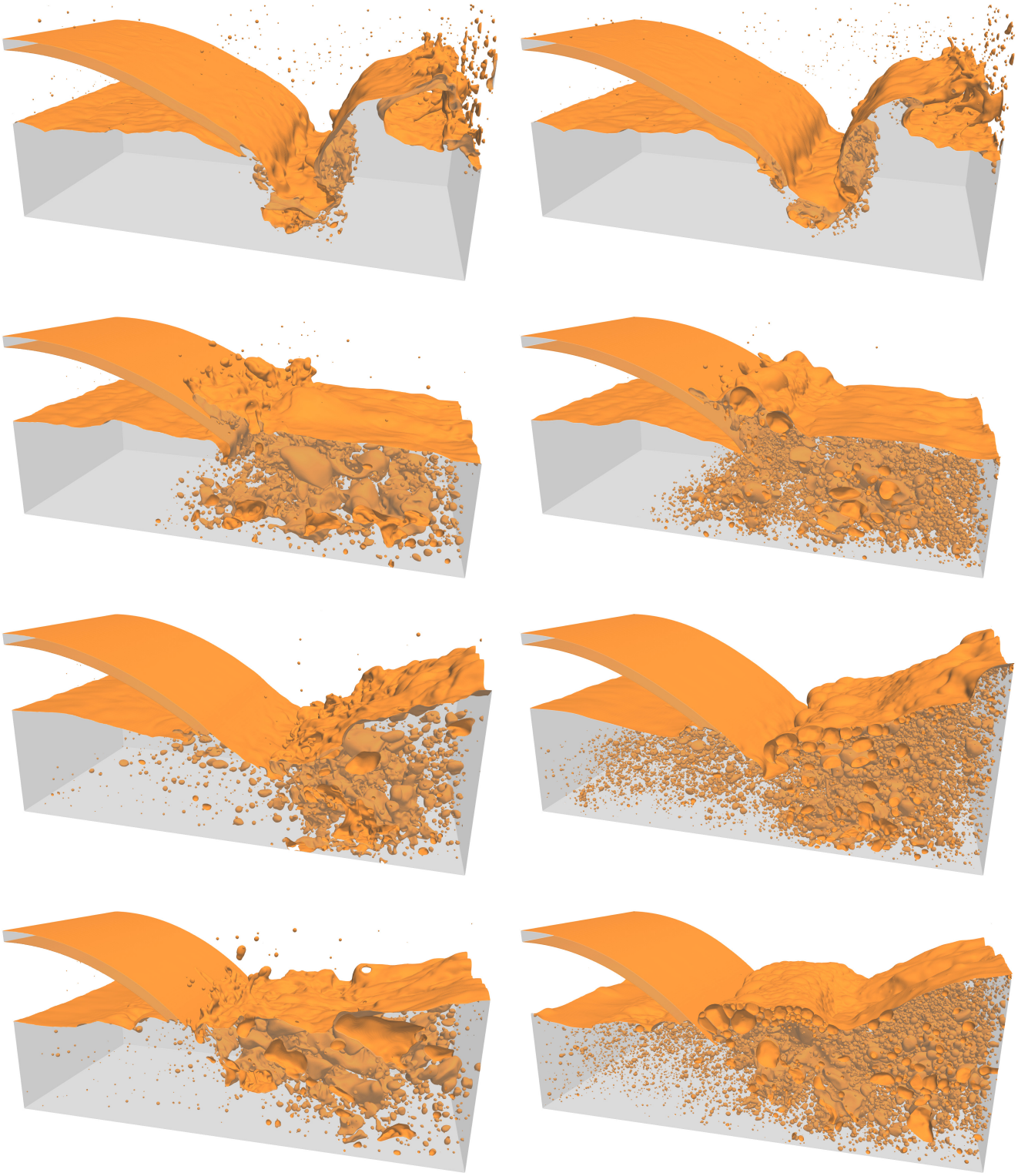

Figure 11: Snapshots of the interface at $t^* = 2.5,\ 6,\ 10$ and 12 with (left) and without coalescence (right).

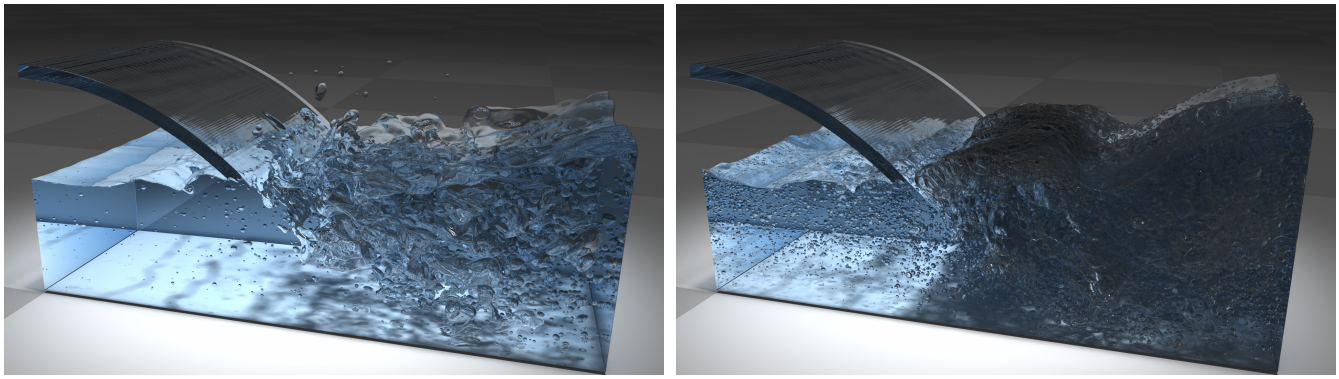

Figure 12: Snapshot of the interface at $t^* = 12$ with (left) and without coalescence (right).

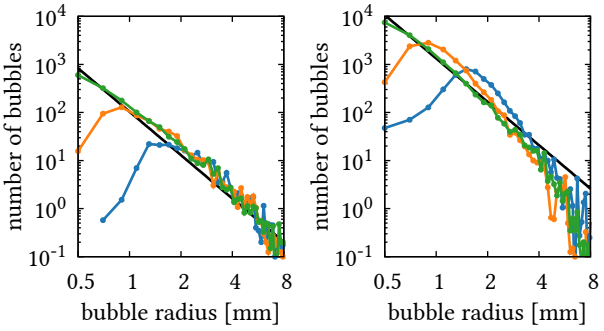

Figure 13: Bubble size distribution $N(r)$ with (left) and without coalescence (right) on a mesh of $N_y = 96$ ——, 192 —— and 384 —— cells compared to theoretical relation [13] $N(r) \propto r^{-10/3}$ ——. The results are averaged in time over $10 < t^* < 12$, and $N(r)$ is the number of bubbles with the equivalent radius between $r - 0.1$ and $r + 0.1$ mm.

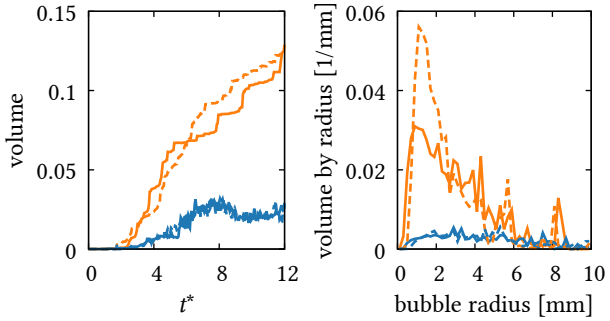

Figure 14: Volume of entrained air (left) and distribution dV/dr of the volume by radius (right), where $V(r)$ is the averaged in time $10 < t^* < 12$ total volume of bubbles with equivalent radius below r. Values of volume are relative to the initial volume of water. Mesh $N_y = 192$ with - - - - - and without coalescence - - - - - and mesh $N_y = 384$ with —— and without coalescence ——.

However, inhibition of coalescence increases the total volume entrained in the water by a factor of five. This is shown in Figure 14 where the volume of entrained air is obtained by summing up the volume of all bubbles. Another quantity reported therein is the histogram of the bubble radius weighted by the bubble volume. It indicates that without coalescence, more air is contained in smaller bubbles, while the effects of bubble coalescence result in a more flat profile of the histogram.

5.4 Formation of foam

The most apparent effect of coalescence inhibition is the clustering of bubbles on the surface. Bubbles generated in the entrainment zone rise to the surface, while they may experience breakups and reduce in size. Figure 15 gives three examples of such bubble trajectories. Bubbles reaching the surface cluster up as foam.

The formation and evolution of foam is an active topic of research [21]. Inhibition of coalescence, achieved in practice by adding surfactants, promotes the existence of stable films between bubbles. Once the liquid drains from the gaps separating the bubbles, sharp junctures appear at the intersections of different layers of film (lamellae) and form the so called Plateau borders. In our simulations we observe these characteristic features of foam as seen from Figure 16. There are multiple places where three bubbles are closely packed and the angle between the interfaces at the junctures approaches 120°.

6 CONCLUSION

We have presented a new algorithm for simulating multiphase flows with non-coalescing bubbles. Our algorithm produces the same results as the standard multi-marker volume-of-fluid but it is more computationally efficient since the number of fields required to represent overlapping interfaces does not depend on the number of bubbles in the simulation.

We have applied our method to investigate the evolution of a waterfall and analyze the effects of coalescence prevention on the bubble size distribution. We observe that prevention of coalescence increases the amount of entrained air in the water and reduces the size of bubbles but does not change the scaling law, which in both cases follows the standard relation of $N(r) \propto r^{-10/3}$. Moreover, our model allows us to describe the formation of clusters of bubbles on the surface which show the characteristic features of foam.

The implementation of our software is based on the Cubism [8, 27] structured grid library which has been extended with support

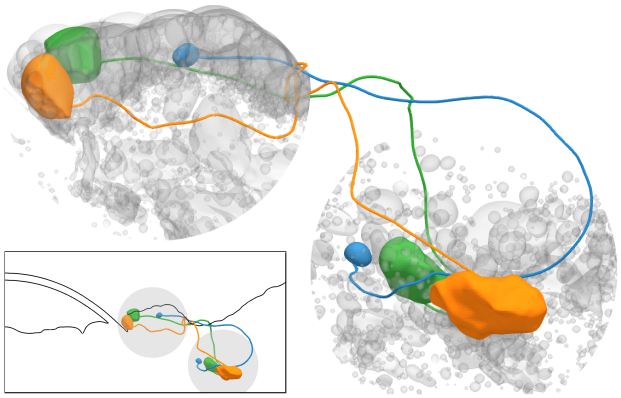

Figure 15: Trajectories of three bubbles and their shapes between time $t^* = 5.5$ (bottom right) and $t^* = 12$ (top left). Other bubbles in circular regions near them are shown in gray, and the inset image illustrates their positions in the domain. Bubbles created by the plunging sheet of water rise towards the surface and become part of a cluster of foam.

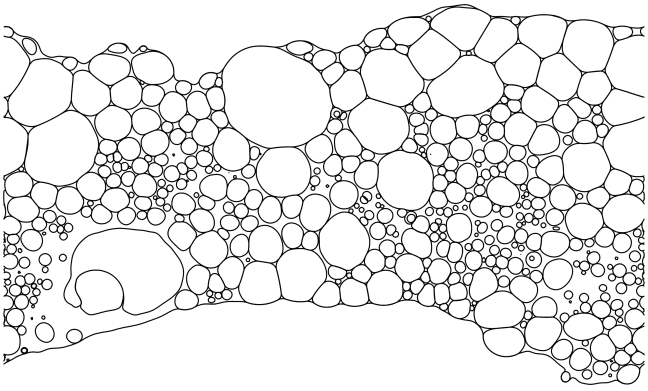

Figure 16: Slice of the interface through $y = 0.55$ at $t^* = 12$. Clusters of bubbles on the surface show characteristic features of foam: thin membranes separating neighboring bubbles (lamellae) and junctions of multiple membranes (Plateau borders).

for externally managed memory. This optimization reduces unnecessary pressure on the memory bus and enables an almost 2x improvement on run time of the advection step. The block processing of our flow solver has further been enhanced by implementing asynchronous message posting followed by a synchronization stage such that network latency is efficiently hidden. We have demonstrated efficient scaling results of our new advection algorithm with $O(10^4)$ cavities being transported. Furthermore, we have extended our software with a hybrid MPI+OpenMP execution model. Our hybrid model exhibits identical weak scaling as in the pure MPI case at slightly worse wall time during the execution of the advection kernel. We have successfully integrated the Hypre library in our hybrid model by utilizing two additional MPI communicators to coordinate between threaded execution and node local matrix assembly prior to Hypre calls. Our hybrid model is currently experimental and will address MPI3 specific one-sided communication features for future development.

ACKNOWLEDGMENTS

This research is funded by grant no. `CRSII5_173860` of the Swiss National Science Foundation. The authors acknowledge the use of computing resources from CSCS (projects s754 and s931).

REFERENCES

[1] Aphros: Parallel solver for incompressible multiphase flows. 2020. https://github.com/cselab/aphros
[2] Eugenio Aulisa, Sandro Manservisi, Ruben Scardovelli, and Stephane Zaleski. 2007. Interface reconstruction with least-squares fit and split advection in three-dimensional Cartesian geometry. *J. Comput. Phys.* 225, 2 (2007), 2301–2319.
[3] Néstor Balcázar, Oriol Lehmkuhl, Joaquim Rigola, and Assensi Oliva. 2015. A multiple marker level-set method for simulation of deformable fluid particles. *International Journal of Multiphase Flow* 74 (2015), 125–142.
[4] Derek YC Chan, Evert Klaseboer, and Rogerio Manica. 2011. Film drainage and coalescence between deformable drops and bubbles. *Soft Matter* 7, 6 (2011), 2235–2264.
[5] Alexandre Joel Chorin. 1968. Numerical solution of the Navier-Stokes equations. *Mathematics of computation* 22, 104 (1968), 745–762.
[6] Emil Coyajee and Bendiks Jan Boersma. 2009. Numerical simulation of drop impact on a liquid–liquid interface with a multiple marker front-capturing method. *J. Comput. Phys.* 228, 12 (2009), 4444–4467.
[7] VSJ Craig, BW Ninham, and RM Pashley. 1993. Effect of electrolytes on bubble coalescence. *Nature* 364, 6435 (1993), 317.
[8] Cubism: Parallel block processing library. 2019. https://gitlab.ethz.ch/mavt-cse/Cubism
[9] Grant B Deane and M Dale Stokes. 2002. Scale dependence of bubble creation mechanisms in breaking waves. *Nature* 418, 6900 (2002), 839.
[10] Luc Deike, W Kendall Melville, and Stéphane Popinet. 2016. Air entrainment and bubble statistics in breaking waves. *Journal of Fluid Mechanics* 801 (2016), 91–129.
[11] Lorena A Del Castillo, Satomi Ohnishi, and Roger G Horn. 2011. Inhibition of bubble coalescence: Effects of salt concentration and speed of approach. *Journal of colloid and interface science* 356, 1 (2011), 316–324.
[12] Said Elghobashi. 2019. Direct numerical simulation of turbulent flows laden with droplets or bubbles. *Annual Review of Fluid Mechanics* 51 (2019), 217–244.
[13] Chris Garrett, Ming Li, and David Farmer. 2000. The connection between bubble size spectra and energy dissipation rates in the upper ocean. *Journal of physical oceanography* 30, 9 (2000), 2163–2171.
[14] HYPRE: Scalable linear solvers. 2017. https://computation.llnl.gov/projects/hypre-scalable-linear-solvers-multigrid-methods
[15] Petr Karnakov, Sergey Litvinov, and Petros Koumoutsakos. 2020. A hybrid particle volume-of-fluid method for curvature estimation in multiphase flows. *International Journal of Multiphase Flow* (2020), 103209.
[16] Petr Karnakov, Fabian Wermelinger, Michail Chatzimanolakis, Sergey Litvinov, and Petros Koumoutsakos. 2019. A High Performance Computing Framework for Multiphase, Turbulent Flows on Structured Grids. In *Proceedings of the Platform for Advanced Scientific Computing Conference on - PASC '19*. ACM Press. https://doi.org/10.1145/3324989.3325727
[17] Kenneth T Kiger and James H Duncan. 2012. Air-entrainment mechanisms in plunging jets and breaking waves. *Annual Review of Fluid Mechanics* 44 (2012), 563–596.
[18] Barry Merriman, James K Bence, and Stanley J Osher. 1994. Motion of multiple junctions: A level set approach. *J. Comput. Phys.* 112, 2 (1994), 334–363.
[19] Zulfaa Mohamed-Kassim and Ellen K Longmire. 2003. Drop impact on a liquid–liquid interface. *Physics of Fluids* 15, 11 (2003), 3263–3273.
[20] Robert I Saye and James A Sethian. 2011. The Voronoi implicit interface method for computing multiphase physics. *Proceedings of the National Academy of Sciences* 108, 49 (2011), 19498–19503.
[21] Robert I Saye and James A Sethian. 2013. Multiscale modeling of membrane rearrangement, drainage, and rupture in evolving foams. *Science* 340, 6133 (2013), 720–724.
[22] Ruben Scardovelli and Stéphane Zaleski. 1999. Direct numerical simulation of free-surface and interfacial flow. *Annual review of fluid mechanics* 31, 1 (1999), 567–603.
[23] Ruben Scardovelli and Stephane Zaleski. 2000. Analytical relations connecting linear interfaces and volume fractions in rectangular grids. *J. Comput. Phys.* 164, 1 (2000), 228–237.

[24] Mark Sussman, Emad Fatemi, Peter Smereka, and Stanley Osher. 1998. An improved level set method for incompressible two-phase flows. *Computers & Fluids* 27, 5-6 (1998), 663–680.

[25] Grétar Tryggvason, Bernard Bunner, Asghar Esmaeeli, Damir Juric, N Al-Rawahi, W Tauber, J Han, S Nas, and Y-J Jan. 2001. A front-tracking method for the computations of multiphase flow. *Journal of computational physics* 169, 2 (2001), 708–759.

[26] Shiyan Wang, Tianqi Guo, Sadegh Dabiri, Pavlos P Vlachos, and Arezoo M Ardekani. 2017. Effect of surfactant on bubble collisions on a free surface. *Physical Review Fluids* 2, 4 (2017), 043601.

[27] Fabian Wermelinger, Ursula Rasthofer, Panagiotis E Hadjidoukas, and Petros Koumoutsakos. 2018. Petascale simulations of compressible flows with interfaces. *Journal of computational science* 26 (2018), 217–225.

[28] Gabriel D Weymouth and Dick K-P Yue. 2010. Conservative volume-of-fluid method for free-surface simulations on cartesian-grids. *J. Comput. Phys.* 229, 8 (2010), 2853–2865.