A High Performance Computing Framework for Multiphase, Turbulent Flows on Structured Grids

Petr Karnakov
Computational Science and Engineering Laboratory
ETH Zürich
Switzerland

Fabian Wermelinger
Computational Science and Engineering Laboratory
ETH Zürich
Switzerland

Michail Chatzimanolakis
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 a high performance computing framework for multiphase, turbulent flows on structured grids. The computational methods are validated on a number of benchmark problems such as the Taylor-Green vortex that are extended by the inclusion of bubbles in the flow field. We examine the effect of bubbles on the turbulent kinetic energy dissipation rate and provide extensive data for bubble trajectories and velocities that may assist the development of engineering models. The implementation of the present solver on massively parallel, GPU enhanced architectures allows for large scale and high throughput simulations of multiphase flows.

CCS CONCEPTS

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

KEYWORDS

multiphase flows, Taylor-Green vortex, plunging jet, volume of fluid, curvature, high performance computing

ACM Reference Format:
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 (PASC '19), June 12–14, 2019, Zurich, Switzerland*. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3324989.3325727

1 INTRODUCTION

Multiphase, turbulent flows are of great significance in numerous industrial applications including nuclear reactors, oil wells and cloud cavitation in pumps. The prediction of such flows has relied traditionally on reduced-order models, scaling arguments or statistical correlations. However, real-world problems usually involve a broad range of spatio-temporal scales which can not be captured accurately with reduced-order methods. Moreover, flow features are three-dimensional (3D) in nature and therefore require advanced techniques applied in direct numerical simulation (DNS) to capture the whole range of scales. In turn DNS may provide data that enhances insight and guides the development of improved engineering models for multiphase flows.

DNS of multiphase flows are particularly challenging as they are faced with the problem of accurate interface tracking. The DNS of three-dimensional multicomponent flows was pioneered by Bunner and Tryggvason [8]. Effects on the lift coefficient due to bubble deformation in vertical shear flow has been shown by the same authors in [9].

Lu et al. [28] examine how the injection of bubbles in channel flow can affect turbulence. They conclude that introducing bubbles close to the walls of low Reynolds number flows can induce a significant decrease of the drag coefficient, claiming that bubble deformability vastly affects turbulence. This was verified in [29], where it is shown that the lateral migration of a bubble strongly depends on its ability to deform. In [6] the authors compare the energy dissipation in turbulent channel flow cases with and without bubbles, showing that the presence of bubbles causes higher dissipation rates. Prakash et. al [37] present an experimental water tunnel approach where energy spectra and velocity fluctuations are examined for varying flow conditions that correspond to single- and multiphase flow. Similar comparisons for homogeneous isotropic turbulence can be found in [7], where the results are based on a spectral cascade model that allows for closure of the Reynolds-Averaged Navier-Stokes (RANS). Feng and Bolotnov [14] study bubble induced turbulence through the interaction of turbulent eddies with a single bubble using a level set approach; bubble deformability is again found to lead to increased turbulence intensity.

In this work we present results for an incompressible, multiphase flow solver working on structured grids with a finite volume discretization. Pressure coupling is achieved through the SIMPLE

*Corresponding author: petros@ethz.ch

PASC '19, June 12–14, 2019, Zurich, Switzerland

ACM ISBN 978-1-4503-6770-7/19/06…$15.00
https://doi.org/10.1145/3324989.3325727

method [16, 34] and the volume-of-fluid (VOF) method with piecewise linear reconstruction of the interface is used to solve the advection equation of the volume fraction [2]. The surface tension force is computed using a novel technique for the curvature estimation. We employ our method to simulate two flow cases: a Taylor-Green vortex modified by the presence of air bubbles and a plunging water jet with air entrainment. Both of these applications result in complex turbulent multiphase flow patterns for which few numerical results exist in the literature. Existing experimental results are used to assess our presented results.

The outline of this paper is as follows: Section 2 presents a short description of the governing equations, followed by an overview of the numerical algorithm used by our solver. Section 3 describes Cubism, the parallel framework used to implement our flow solver. The applications are described in Section 4; finally Section 5 concludes our work.

2 MODEL

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

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

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

and the advection equation for the volume fraction α

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

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.

2.1 Numerical algorithm

We use a finite volume discretization based on the SIMPLE method for pressure coupling [16, 34]. The advection equation is solved using the VOF method with piecewise linear reconstruction [2]. The key computational challenge of multiphase flows is the tracking or capturing of the interfaces between phases in the flow field and computing the surface tension force. A number of approaches suggest heterogeneous models [27], combining a level-set method for the resolved interfaces with a Lagrangian treatment of smaller bubbles and drops [30, 31]. A hybrid technique [40] resolves the interfaces with the VOF method and the smaller details with a homogeneous mixture approach. One drawback of such approaches is that they rely on criteria for switching between the models and suffer from inconsistencies in the intermediate regimes. Techniques such as Adaptive Mesh Refinement (AMR) have been shown to perform well in cases of isolated interfaces [35]. However, the refinement is not beneficial in regions with a uniform distribution of smaller bubbles.

We refer to [36] for a detailed review of the previous approaches for numerical models of surface tension. According to the review, the method of height functions [35] is the state-of-the-art method commonly used for computing the surface tension force in the context of VOF methods. Here we combine the standard VOF formulation with a novel technique for the interface curvature estimation [24]. We demonstrate better accuracy at low resolutions than the open-source solver Basilisk [3] implementing the method of height functions. Our method allows for solving the transport problems with bubbles and drops at low resolutions up to one cell per radius. Therefore, we apply the same algorithm across various scales on a uniform mesh.

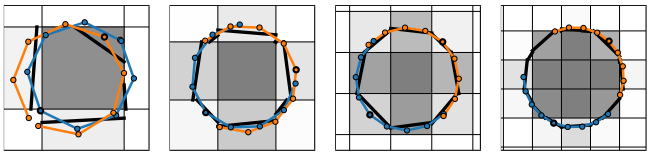

Figure 1: Line segments of the interface (black lines) and positions of particles from two selected cells for a circle at resolutions of 0.59, 0.84, 1.19 and 1.68 cells per radius. The particles are connected by lines (blue and orange) and the central particle is highlighted by a thicker edge.

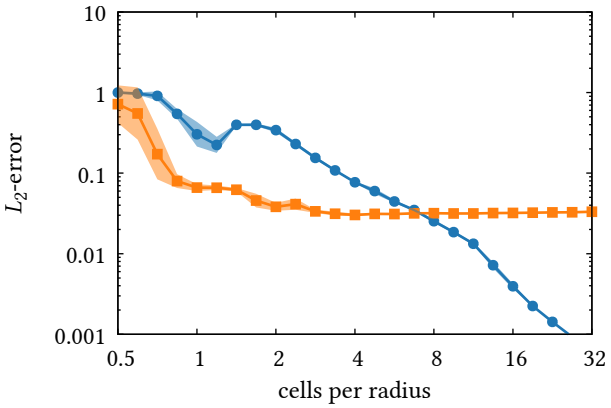

Figure 2: Curvature error for a sphere depending on the number of cells per radius by Basilisk [3] —●— in comparison to the present method —■— .

2.2 Curvature estimation

We use the piecewise linear reconstruction of the interface obtained from the discrete volume fraction field [2]. The interface curvature is estimated in each cell independently using the polygons from a $5\times5\times5$ stencil.

In two dimensions, the reconstructed interface is a set of line segments. The curvature in one cell is computed by fitting a circular arc to the interface. The circular arc is represented as a string of connected particles $\boldsymbol{x}_i$, $i = 1,\ldots,N$ which evolve until equilibration under constraints and forces attracting them to the interface. The constraints maintain a prescribed distance h_p between the particles and a uniform angle between them. This implies that the particles belong to a circular arc and, therefore, their configuration can be described by three parameters: position of the central particle $\boldsymbol{p}$,

the orientation angle ϕ and the bending angle θ:

$$\boldsymbol{x}_i(\boldsymbol{p},\phi,\theta) = \begin{cases} \boldsymbol{p} + \sum_{j=1}^{i-c} h_p \boldsymbol{e}\big(\phi + (j-\frac{1}{2})\,\theta\big) & i > c\,, \\ \boldsymbol{p} & i = c\,, \\ \boldsymbol{p} - \sum_{j=1}^{c-i} h_p \boldsymbol{e}\big(\phi - (j-\frac{1}{2})\,\theta\big) & i < c\,, \end{cases} \tag{4}$$

where $\boldsymbol{e}(\xi) = \cos\xi\,\boldsymbol{e}_x + \sin\xi\,\boldsymbol{e}_y$ and $c = (N-1)/2$ is the central particle. The attraction force on a particle at position $\boldsymbol{x}$ is computed as $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{x}_L(\boldsymbol{x}) - \boldsymbol{x}$, where $\boldsymbol{x}_L(\boldsymbol{x})$ is the nearest point on the interface. Furthermore, we apply an under-relaxation to the force and modify the force by replacing the nearest line segment with a circular arc passing through its endpoints and having the curvature known from the previous iteration of the optimization algorithm. Initially the central particle is placed at the line segment at which we estimate the curvature and particles are oriented along the line segment. The iterative optimization algorithm computes corrections of parameters $\boldsymbol{x}$, ϕ and θ from the forces. For instance, the angle ϕ is corrected by

$$\Delta\phi = \sum_{i=1}^{N} \boldsymbol{f}_i \cdot \frac{\partial \boldsymbol{x}_i}{\partial \phi} \;/\; \sum_{i=1}^{N} \frac{\partial \boldsymbol{x}_i}{\partial \phi} \cdot \frac{\partial \boldsymbol{x}_i}{\partial \phi}\,, \tag{5}$$

where $\boldsymbol{f}_i = \boldsymbol{f}(\boldsymbol{x}_i)$. The same correction is applied to θ, and $\boldsymbol{p}$ is corrected by the force acting on the central particle so that $\Delta\boldsymbol{p} = \boldsymbol{f}_c$.

In three dimensions, the reconstructed interface is a set of planar polygons. We follow the definition of the mean curvature of a surface and estimate the curvature of the interface at each polygon as the arithmetic mean over multiple cross sections by planes perpendicular to the polygon. A plane section of the interface is again a set of line segments to which we apply the above procedure. We use $N = 9$ particles per cross section and two planes per cell which results in 18 particles to estimate the curvature in one cell. The algorithm requires multiple iterations but needs to be applied only in cells containing the interface. The current implementation takes about 20% of the runtime for the considered problems with further potential for optimization.

The role of particles in our method is only for computing the curvature given a piecewise linear reconstruction of the interface. This is in contrast to other methods using particles to capture interfaces. We note that there are two techniques that are called Particle Level Sets in the literature. The Particle Level Set technique introduced by Enright and Fedkiw [12] that solves the advection equations on a grid and particles are added to improve the subgrid scale resolution. The Lagrangian Particle Level Set introduced by Hieber and Koumoutsakos [21], solves the level set equations consistently on remeshed particles and it has also been extended to particle-wavelet level sets by Bergdorf and Koumoutsakos [5]. These are also different from solving the advection equation on particles using Smoothed Particle Hydrodynamics [1]. Such methods compute the surface tension using derivatives of fields represented on particles which limits their accuracy at low resolutions due to the particle distortion. We note that the Lagrangian Particle Level Set avoids this distortion by remeshing techniques that allow for accurate derivative calculations.

Figure 1 illustrates the equilibrium configurations of particles at resolutions below two cells per radius. Our algorithm outperforms the commonly used method of height functions [3, 35] at low resolutions as shown in Figure 2. The algorithm estimates the curvature with the relative error below 0.1 even at low resolutions up to one cell per radius. This enables multiphase flow simulations over a large range of spatial scales. We also provide a visual web-based demonstration of the method[1] and a reference implementation in Python [2].

3 SOFTWARE DESIGN

Our multiphase flow solver is based on an open-source, structured grid library Cubism [10], which has been demonstrated to exhibit excellent scaling on large scale systems [17, 18, 39]. The present work extends Cubism by integrating coroutines [26] which allow for a strict separation of the user code from the structured grid framework. This separation minimizes the exposure of user code and significantly simplifies the implementation of complex applications that make use of Cubism.

3.1 Implementation

The Cubism library divides the grid into small compute blocks to increase parallel granularity. The size of the blocks is chosen such as to optimize temporal and spatial locality on the target architecture. The current implementation of the solver utilizes the Message Passing Interface (MPI) library to distribute the compute blocks to parallel processors. Figure 3 shows the rank-level organization of the software. User code operates on compute blocks which may enqueue communication requests into a buffer that is maintained by Cubism. When a block issues a communication request, it will suspend its execution flow and return control to the caller. This functionality is implemented with coroutines [26]. User code is further divided into stages which usually correspond to different compute kernels, such as advection or diffusion operators. These kernels are stencil based and involve communication of halo cells for blocks adjacent to subdomain boundaries. In our current implementation, each rank processes compute stages sequentially for each block. A stage may partially complete and yield control to the caller if synchronization of messages is required at some point. In such a case, the state of the block is stored in the per-block instance of the suspended stage. Our current implementation does not account for compute-transfer (C/T) overlap. We enable C/T overlap by dividing blocks on a rank into boundary blocks (message-bound) and interior blocks that do not require communication. A queue of communication requests is then created by first processing the list of boundary blocks, which then suspend due to outstanding messages. This C/T scheme is currently under development. The communication queue is part of a synchronizer class which coordinates message exchange between ranks using point-to-point routines or collective operations depending on the message type. Asynchronous routines are used for point-to-point communication such that interior blocks can be processed to hide communication overhead. Once the outstanding messages have been exchanged, the synchronizer passes control back to the blocks with suspended stages which become active again to finalize their work. In general, a stage implemented in user code operates on

[1] Visual demonstration: https://cselab.github.io/hydro/grid.html
[2] Reference implementation: https://cselab.github.io/hydro/curv.py

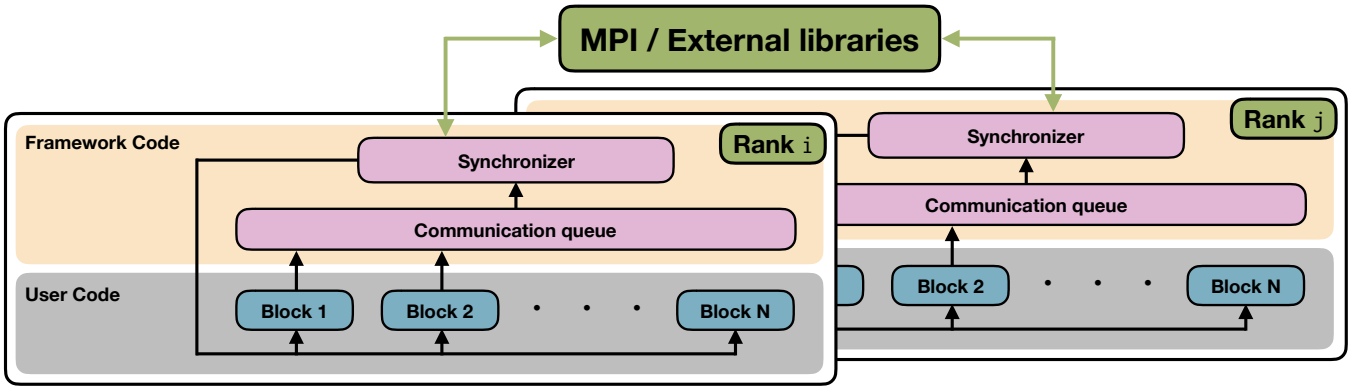

Figure 3: Separation of user code from Cubism. User code is mapped to compute blocks which may enqueue communication requests in a queue managed by Cubism. Communication tasks are dequeued by a synchronizer object which coordinates message exchange between ranks through MPI or issues calls to external libraries.

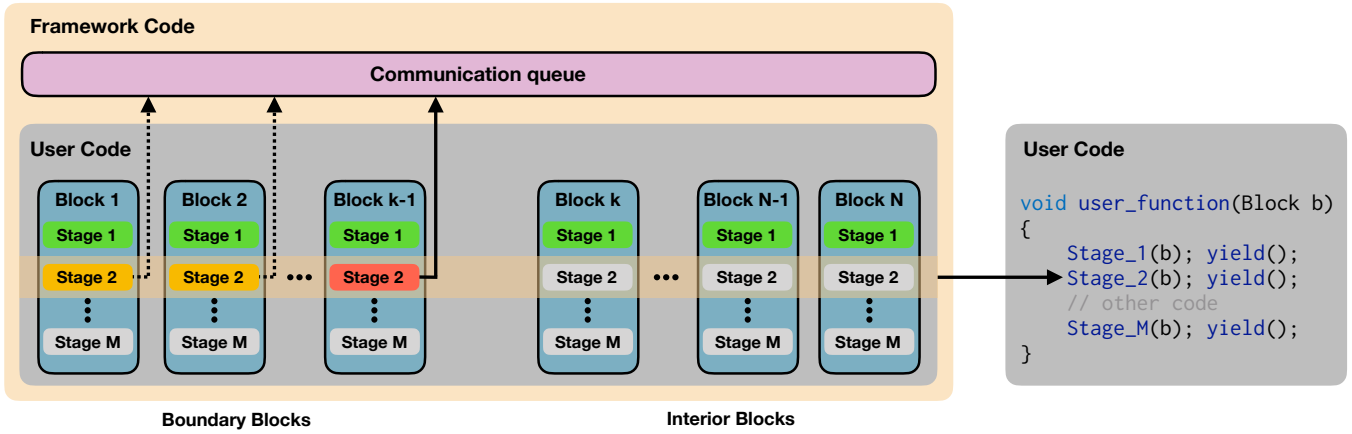

Figure 4: User code is mapped to compute blocks and executed sequentially. Utilization of coroutines allows for the suspension of a compute block enabling C/T overlap by asynchronous execution. Green stages indicate completion, orange stages are in suspended state and red indicates currently executed stages. Gray stages are pending.

one or multiple fields, which are a subset of the data required to describe the problem. Density or velocity are examples of scalar or vector fields, respectively. This abstraction of the problem allows us to write modular user code which is mapped to the compute blocks on a rank. Individual blocks can be seen as independent work packages. It is therefore straightforward to implement new operators without the need to modify any code in Cubism.

Figure 4 illustrates this algorithm based on the asynchronous point-to-point communication example outlined above. Blocks are processed sequentially, starting with blocks that are adjacent to subdomain boundaries. If a block is required to suspend execution due to communication, it will return control to the caller which then starts to process the next block. Successfully completed stages are highlighted in green, whereas stages in suspended states are highlighted in orange and the currently executing stage is shown in red. Stages not yet processed are indicated in light gray. The interface of the synchronizer class is not limited to the MPI library only. External libraries can also communicate with the synchronizer in a similar way. Our current implementation utilizes this extended interface to solve large linear systems through the Hypre library [13, 22] (namely, the conjugate gradient solver for the Poisson equation and GMRES for the convection-diffusion equation) as well as for I/O through the HDF5 library.

The granularity introduced by the compute blocks also enables thread-level parallelism (TLP) that can be combined with MPI for a hybrid execution model. In this case, the schematic in Figure 4 would allow for multiple blocks with active stages (marked with red color). Our implementation currently does not exploit TLP but is planned to be added in a future release.

Moreover, user functions are designed to operate on a particular compute block passed as an argument, see Figure 4. This design paradigm allows for flexible optimizations of individual stages that evaluate a computational pattern on a block. In particular, it enables specific programming models that can be applied to exploit data level parallelism using tools such as Intel's ISPC compiler or GPU architectures, for example.

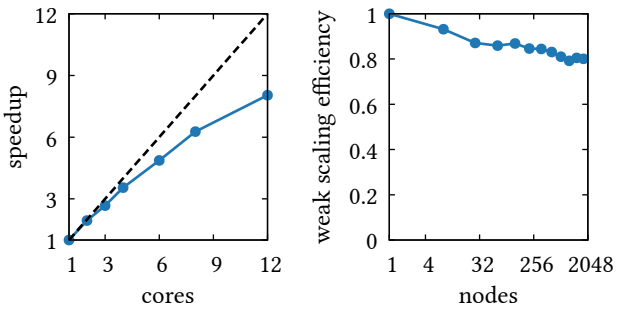

Figure 5: Strong and weak scaling on Piz Daint for the benchmark described in Section 3.2.

3.2 Scaling and memory requirements

We demonstrate strong and weak scaling of our solver based on a Taylor-Green vortex in liquid water extended with air bubbles. The scaling analysis is performed on Piz Daint at the Swiss National Supercomputing Center (CSCS) and is shown in Figure 5. The measurement of the strong scaling is performed on a single node using a mesh with 96^3 cells and a block size of 8^3 cells, resulting in 1728 blocks. The strong scaling efficiency decreases with core count due to an increased usage of the memory subsystem. The analysis of weak scaling is performed with a constant problem size of 96^3 cells per node with 12 ranks. The solver is run on up to 2197 nodes with a weak scaling efficiency of 80 %. We expect the weak scaling efficiency to further improve with the deployment of the C/T overlap scheme outlined in the previous section. The memory footprint scales linearly with the number of cells and amounts to 8 KiB per cell. A typical problem size for production runs consists of a mesh size with 384^3 cells solved on 288 nodes on Piz Daint, which results in a overall memory footprint of 1.5 GiB per node.

4 APPLICATIONS

4.1 Taylor-Green vortex with bubbles

The Taylor-Green vortex is a classical benchmark for testing the capabilities of flow solvers to perform DNS of homogeneous turbulent flows. Here we present simulations that augment the classical flow field with a gaseous phase. The velocity is initialized in a periodic domain $[0, 2\pi]^3$ as $v_x = \sin x \cos y \cos z$, $v_y = -\cos x \sin y \cos z$ and $v_z = 0$. The gaseous phase is represented by 890 spherical bubbles of diameter $d = 0.2$ placed at random positions resulting in the average volume fraction of 1.4 %. The parameters of the problem are the Reynolds number $\mathrm{Re} = \rho_1/\mu_1 = 1600$, the Weber number $\mathrm{We} = \rho d/\sigma = 2$, density ratio $\rho_2/\rho_1 = 0.01$ and viscosity ratio $\mu_2/\mu_1 = 0.01$. The spatial resolution required for the numerical simulations is determined from the single-phase flow by comparing the energy dissipation rate with reference data [42] as shown in Figure 9a. We use two mesh sizes of 256^3 and 384^3.

Figure 8 presents three snapshots of our simulation, visualizing the bubble shapes and the vorticity magnitude. Initially all bubbles have the same size and larger bubbles form due to coalescence of smaller bubbles.

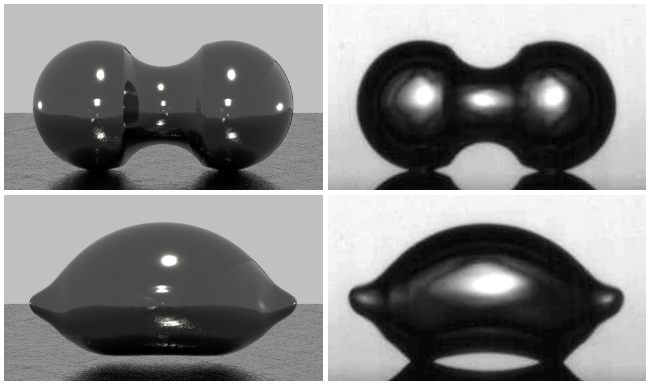

Figure 6: Near-wall coalescence of two bubbles. The bubble shapes at two time instants by the present method (left) compared to the experiment [41] (right).

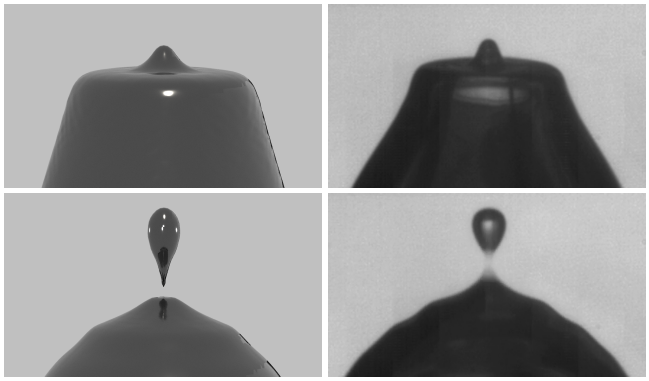

Figure 7: Coalescence of bubbles with satellite formation. The bubble shapes at two time instants by the present method (left) compared to the experiment [43] (right). Reprinted from [43] with the permission of AIP Publishing.

Bubble coalescence is driven by the surface tension force which tends to restore a spherical shape of the newly formed bubble, leading to an abrupt increase in the kinetic energy [41]. For the purpose of validation, we have performed additional simulations of individual coalescence events using our multiphase flow solver and present the shapes of bubbles in comparison to experimental data: coalescence of two bubbles near a solid wall [41] in Figure 6 and the formation of a satellite bubble [43] in Figure 7. Our solver also agrees well with experimental data for the bubble dynamics in electrochemical cells [20] where coalescence of bubbles plays an important role.

During the evolution of the Taylor-Green vortex, the number of bubbles reported in Figure 10b reduces by a factor of three in the considered time frame. Multiple coalescence events happening simultaneously appear as fluctuations of the energy dissipation rate reported in Figure 9b where the blue shade corresponds to the standard deviation computed over a window of width 0.5. The amplitude of the fluctuations increases for a higher rate of the coalescence events represented by the slope in Figure 10b.

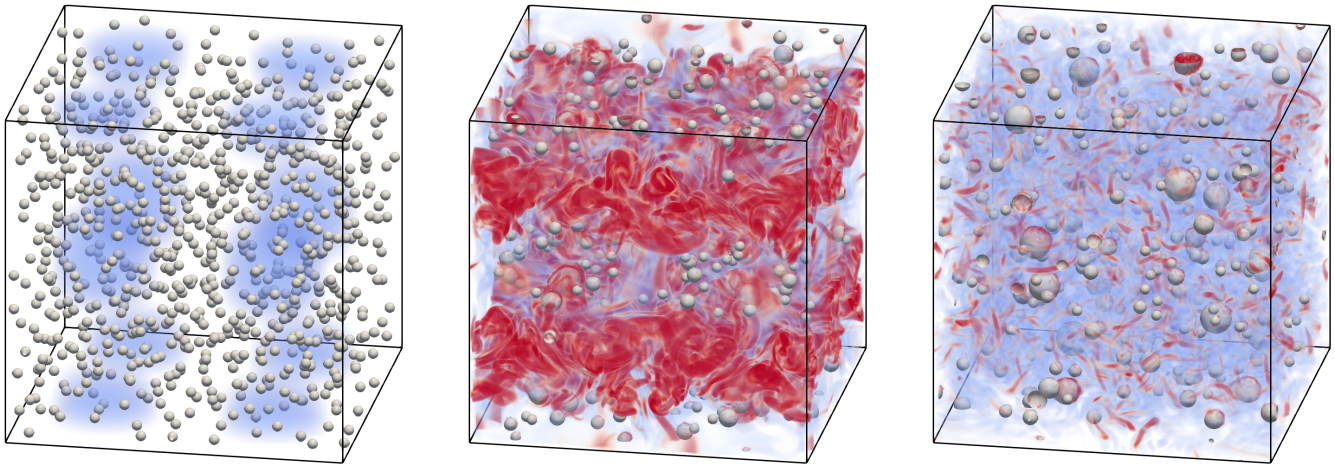

Figure 8: Taylor-Green vortex. Snapshots of the flow field with bubble surfaces (gray) and vorticity magnitude (increasing values from blue to red) at various time instants $t = 0,\ 10$ and 20.

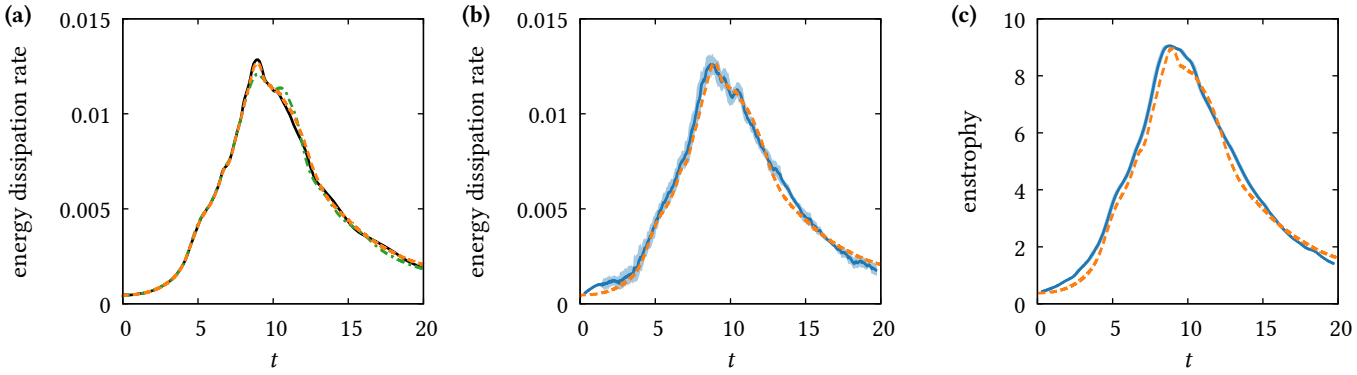

Figure 9: Taylor-Green vortex. (a) The energy dissipation rate without bubbles on mesh 256^3 and 384^3 compared to reference [42] . (b-c) The energy dissipation rate and enstrophy on mesh 384^3 with bubbles and without bubbles . The blue shade shows the standard deviation.

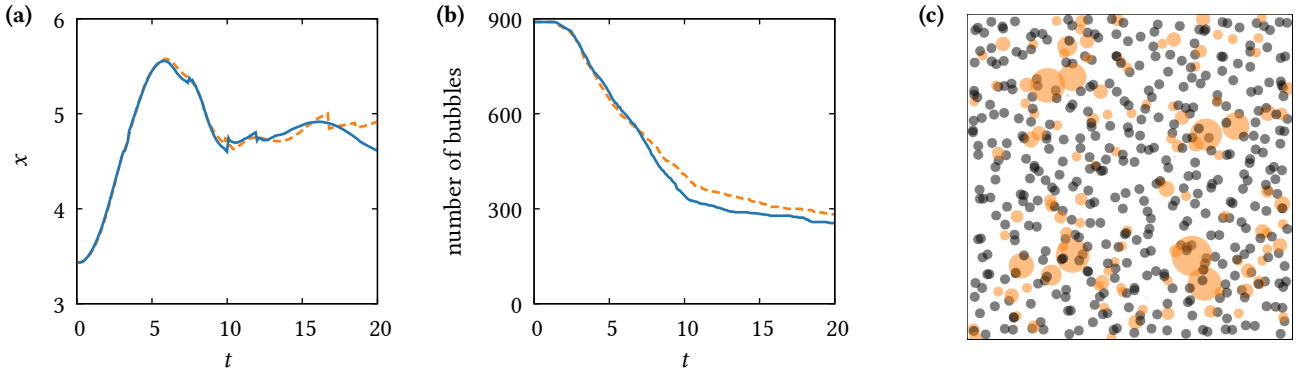

Figure 10: Taylor-Green vortex. (a-b) The x-trajectory of one bubble and the number of bubbles over time on mesh 256^3 and 384^3 . (c) Configurations of bubbles at $t = 0$ (black shades) and $t = 20$ (orange shades). Circles represent positions and the equivalent radius of bubbles from region $\frac{1}{2}\pi < z < \frac{3}{2}\pi$ projected on the z-plane.

Independence on the mesh size is also verified for the trajectories of individual bubbles as shown in Figure 10a for one bubble starting at position (3.44, 4.45, 3.77). Sudden jumps in the trajectory correspond to coalescence with other bubbles.

At the considered gas volume fraction of 1.4%, the presence of bubbles does not affect the energy dissipation rate, while it increases the enstrophy by about 10% for the first half of the simulation, as shown in Figure 9c. This is expected as the vorticity is generated on the bubbles.

The initial and final locations of bubbles presented in Figure 10c suggest that the bubbles migrate towards the centers of the vortices. This effect has also been investigated by [15] in the two-dimensional case.

4.2 Plunging jet with air entrainment

A liquid jet is injected at a distance from the interface separating a liquid column and a layer of air. The jet passes through the air dragging it deeper into the water column and causes the air entrainment generating bubbles on the interface [25]. This problem has been investigated experimentally in [4, 11, 19, 33, 38] and numerically in [38, 40]. The numerical study in [38] was presented for a two-dimensional axisymmetric domain using a mixture model and level-set approaches. A more recent three-dimensional numerical study [40] reported a good agreement with the experiments from [38] at the initial stages of the jet penetration but the accuracy of the method at lower resolutions was not sufficient to describe the smaller bubbles after equilibration. Moreover, their hybrid algorithm involved semi-empirical criteria for switching between the mixture model and resolved interfaces.

For our numerical simulations, we use parameters from the experimental work [4]. The jet nozzle is represented as a square of side 6.0 mm on the top boundary with a prescribed liquid velocity of 3.78 m/s. The distance between the nozzle tip and the interface is 30 mm. The liquid density and dynamic viscosity are that of water and the surface tension coefficient is that for water/air interfaces. The gas/liquid density and dynamic viscosity ratios are both set to 0.01. In contrast to the experiment which is done in a large tank, the simulation domain is surrounded by free-slip walls and has a height of 400 mm and a square cross section of width 100 mm. To ensure that the total volumetric flux is zero, outlet boundaries are positioned in the region between 10 and 20 mm below the interface. We perform the simulations on a mesh of size $N \times 4N \times N$ with $N = 128$ and $N = 256$.

Figure 11 shows a snapshot of vorticity magnitude and the bubble shapes from the simulation compared to an image from the experiment done with the jet height of 43 mm (versus 30 mm in the simulation). As seen from the evolution of the kinetic energy of the mixture in Figure 12a, on the coarse mesh the simulation has reached a steady state. On the fine mesh a longer evolution would be needed which, however, was not done due to the computational cost. We observe a qualitative match for the shape of the jet and the distribution of the bubble size: bubbles near the axis move down along with the jet and on the outer side the bubbles rise towards the interface. A quantitative comparison, however, reveals differences from the experiment. The jet penetration depth (Figure 12b) is the same on both meshes but is overpredicted by a factor of 2.5. One

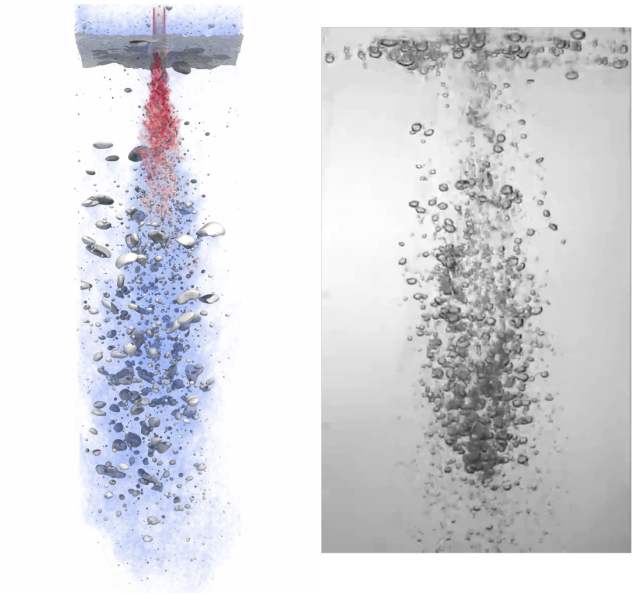

Figure 11: The isosurface of volume fraction (gray) and the vorticity magnitude (increasing values from blue to red) from the simulation on the fine mesh (left) compared to the experiment [4] (right).

explanation for this is that the experiment considers the jet in a large tank while in the simulation, the liquid volume is surrounded by free-slip walls. Another explanation for the discrepancy is the sensitivity of the penetration depth to the boundary conditions. For instance, as reported in the experiment, changing the impact velocity by 5% (for instance, by increasing the jet height) would change the penetration depth by 50%. The situation may improve with a more accurate treatment of the nozzle currently represented as a square area on the top boundary with a prescribed constant velocity.

From the simulation data, we extract the shapes and locations of bubbles and measure the distribution of the equivalent radius from their volume. We first triangulate the isosurfaces of the gas volume fraction with the marching cubes algorithm and then separate the connected components. All components which have a volume smaller than one computational cell are excluded from the evaluation. Figure 12c shows a good agreement with the experimental data [4].

We compare the velocity of selected bubbles rising on the outer side with experimental data [32] for the terminal velocity of a single bubble rising in clean viscous liquids. The selected bubbles are shown in Figure 13a. The velocity measured for the selected bubbles is about 30% higher than the terminal velocity as presented in Figure 13b. Note, however, that one can not expect an exact match with the terminal velocity due to the influence of surrounding bubbles and the bulk flow. This comparison serves the purpose of showing that even at low resolutions a reasonable accuracy is observed. Meanwhile, the numerical algorithm was verified with a finite volume solver Basilisk [3, 35] for a single rising bubble in the two-dimensional case shown in Figure 13c.

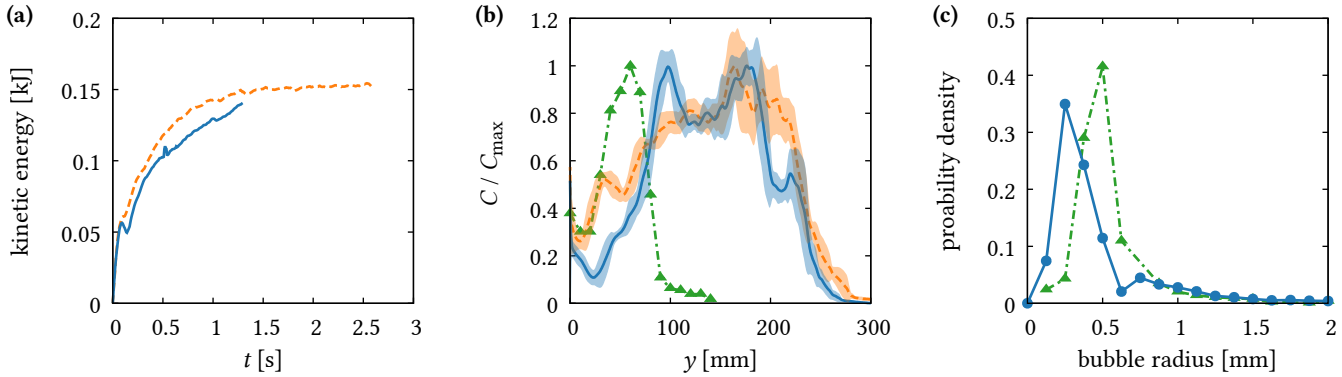

Figure 12: Pluging jet. (a) The kinetic energy of the mixture over the domain on the coarse mesh and fine mesh . (b) Axial concentration of gas versus the distance from the interface averaged over the last 0.13 s on the coarse mesh and fine mesh compared to the experiment [4] . (c) The distribution of the average bubble radius on the fine mesh compared to the experiment [4] .

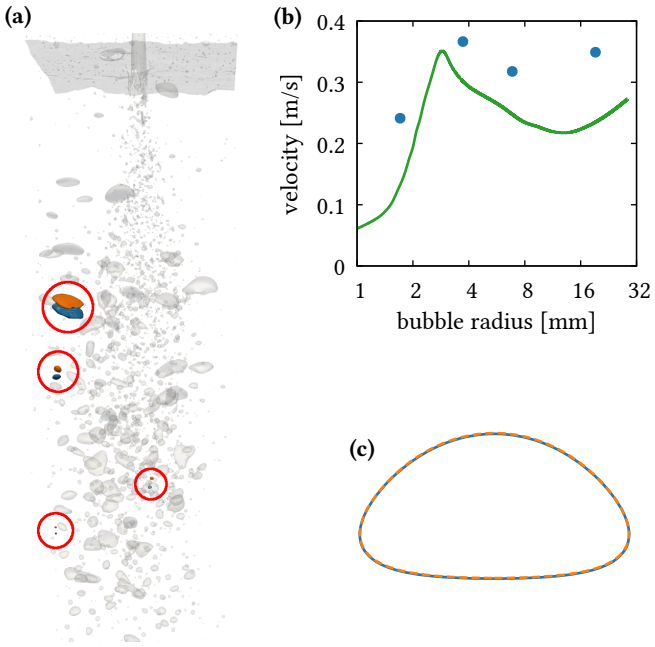

Figure 13: Plunging jet. (a) The isosurface of volume fraction with four selected bubbles highlighted by red circles. Their shapes at two time instants 0.013 s apart are shown in blue and orange. (b) The vertical velocity of the selected bubbles on the fine mesh and the terminal velocity of bubbles rising in an infinite tank [32] . (c) The shape of a rising bubble in two dimensions from Case 1 of [23] produced by Basilisk [3] and the present method .

5 CONCLUSION

We present an HPC framework for DNS of multiphase turbulent flows with a new method for tracking interfaces on structured grids. The proposed numerical method for multiphase flows uses a novel technique for curvature estimation which improves the accuracy at low resolutions. Therefore, the same algorithm can be applied across various spatial scales. We have demonstrated its potential to resolve finer scales than commonly found in other studies. Future work will be devoted to further validation and more detailed analysis of the influence of bubbles on turbulent flows. The framework is based on the Cubism library extended using the concept of coroutines to improve the flexibility in the communication sequence as required by the numerical algorithm. Future improvements will support an execution model exploiting thread-level parallelism and enable optimizations using accelerators and vectorization. Moreover, the compute-transfer overlap will be implemented and further improved by simultaneous execution of independent stages.

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 (project s754).

REFERENCES

[1] S Adami, XY Hu, and Nikolaus A Adams. 2010. A new surface-tension formulation for multi-phase SPH using a reproducing divergence approximation. *J. Comput. Phys.* 229, 13 (2010), 5011–5021.
[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] Basilisk. 2019. http://basilisk.fr
[4] Jesse Belden, Sai Ravela, Tadd T. Truscott, and Alexandra H. Techet. 2012. Three-dimensional bubble field resolution using synthetic aperture imaging: application to a plunging jet. *Experiments in Fluids* 53, 3 (jun 2012), 839–861. https://doi.org/10.1007/s00348-012-1322-4
[5] Michael Bergdorf and Petros Koumoutsakos. 2006. A Lagrangian particle-wavelet method. *Multiscale Modeling & Simulation* 5, 3 (2006), 980–995.
[6] Igor A. Bolotnov, Kenneth E. Jansen, Donald A. Drew, Assad A. Oberai, Richard T. Lahey, and Michael Z. Podowski. 2011. Detached direct numerical simulations of turbulent two-phase bubbly channel flow. *International Journal of Multiphase Flow* 37, 6 (jul 2011), 647–659. https://doi.org/10.1016/j.ijmultiphaseflow.2011.03.002
[7] Igor A. Bolotnov, Richard T. Lahey, Donald A. Drew, and Kenneth E. Jansen. 2008. Turbulent cascade modeling of single and bubbly two-phase turbulent flows. *International Journal of Multiphase Flow* 34, 12 (dec 2008), 1142–1151. https://doi.org/10.1016/j.ijmultiphaseflow.2008.06.006
[8] Bernard Bunner and Grétar Tryggvason. 1999. Direct numerical simulations of three-dimensional bubbly flows. *Physics of Fluids (1994-present)* 11, 8 (1999), 1967–1969.

[9] Bernard Bunner and Grétar Tryggvason. 2003. Effect of bubble deformation on the properties of bubbly flows. *Journal of Fluid Mechanics* 495 (2003), 77–118.

[10] Cubism: Parallel block processing library. 2019. https://gitlab.ethz.ch/mavt-cse/Cubism

[11] M. El Hammoumi, J. L. Achard, and L. Davoust. 2002. Measurements of air entrainment by vertical plunging liquid jets. *Experiments in Fluids* 32, 6 (01 Jun 2002), 624–638. https://doi.org/10.1007/s00348-001-0388-1

[12] Douglas Enright, Ronald Fedkiw, Joel Ferziger, and Ian Mitchell. 2002. A hybrid particle level set method for improved interface capturing. *Journal of Computational physics* 183, 1 (2002), 83–116.

[13] Robert D Falgout and Ulrike Meier Yang. 2002. hypre: A library of high performance preconditioners. In *International Conference on Computational Science*. Springer, 632–641.

[14] Jinyong Feng and Igor A. Bolotnov. 2017. Evaluation of bubble-induced turbulence using direct numerical simulation. *International Journal of Multiphase Flow* 93 (jul 2017), 92–107. https://doi.org/10.1016/j.ijmultiphaseflow.2017.04.003

[15] Antonino Ferrante and Said E Elghobashi. 2007. On the effects of microbubbles on Taylor–Green vortex flow. *Journal of Fluid Mechanics* 572 (2007), 145–177.

[16] Joel H Ferziger and Milovan Peric. 2012. *Computational methods for fluid dynamics*. Springer Science & Business Media.

[17] Panagiotis E. Hadjidoukas, Diego Rossinelli, Babak Hejazialhosseini, and Petros Koumoutsakos. 2015. From 11 to 14.4 PFLOPs: Performance Optimization for Finite Volume Flow Solver. In *Proceedings of the 3rd International Conference on Exascale Applications and Software*.

[18] Panagiotis E. Hadjidoukas, Diego Rossinelli, Fabian Wermelinger, Jonas Sukys, Ursula Rasthofer, Christian Conti, Babak Hejazialhosseini, and Petros Koumoutsakos. 2015. High throughput simulations of two-phase flows on Blue Gene/Q. In *Parallel Computing: On the Road to Exascale, Proceedings of the International Conference on Parallel Computing, ParCo 2015, 1-4 September 2015, Edinburgh, Scotland, UK*. 767–776. https://doi.org/10.3233/978-1-61499-621-7-767

[19] K. Harby, S. Chiva, and J.L. Muñoz-Cobo. 2014. An experimental study on bubble entrainment and flow characteristics of vertical plunging water jets. *Experimental Thermal and Fluid Science* 57 (2014), 207 – 220. https://doi.org/10.1016/j.expthermflusci.2014.04.004

[20] S Mohammad H Hashemi, Petr Karnakov, Pooria Hadikhani, Enrico Chinello, Sergey Litvinov, Christophe Moser, Petros Koumoutsakos, and Demetri Psaltis. 2019. A versatile and membrane-less electrochemical reactor for the electrolysis of water and brine. *Energy & Environmental Science* (2019).

[21] Simone E Hieber and Petros Koumoutsakos. 2005. A Lagrangian particle level set method. *J. Comput. Phys.* 210, 1 (2005), 342–367.

[22] HYPRE: Scalable linear solvers. 2017. https://computation.llnl.gov/projects/hypre-scalable-linear-solvers-multigrid-methods

[23] Shu-Ren Hysing, Stefan Turek, Dmitri Kuzmin, Nicola Parolini, Erik Burman, Sashikumaar Ganesan, and Lutz Tobiska. 2009. Quantitative benchmark computations of two-dimensional bubble dynamics. *International Journal for Numerical Methods in Fluids* 60, 11 (2009), 1259–1288.

[24] Petr Karnakov, Sergey Litvinov, and Petros Koumoutsakos. [n. d.]. Connected particles for curvature estimation applied to flows with surface tension. ([n. d.]). (In preparation).

[25] 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.

[26] Donald E. Knuth. 1997. *The Art of Computer Programming, Volume 1 (3rd Ed.): Fundamental Algorithms*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA.

[27] Yue Ling, Stéphane Zaleski, and R Scardovelli. 2015. Multiscale simulation of atomization with small droplets represented by a Lagrangian point-particle model. *International Journal of Multiphase Flow* 76 (2015), 122–143.

[28] Jiacai Lu, Arturo Fernández, and Gretar Tryggvason. 2005. The effect of bubbles on the wall drag in a turbulent channel flow. *Physics of Fluids (1994-present)* 17, 9 (2005), 095102.

[29] Jiacai Lu and Gretar Tryggvason. 2008. Effect of bubble deformability in turbulent bubbly upflow in a vertical channel. *Physics of Fluids (1994-present)* 20, 4 (2008), 040701.

[30] Xiuxiu Lyu, Xiangyu Hu, and Nikolaus A Adams. 2018. Investigation of Cavitation Bubble Cloud with Discrete Lagrangian Tracking. In *Proceedings of the 10th International Symposium on Cavitation (CAV2018)*. ASME Press.

[31] Kazuki Maeda and Tim Colonius. 2018. Eulerian–Lagrangian method for simulation of cloud cavitation. *J. Comput. Phys.* (2018).

[32] T. Maxworthy, C. Gnann, M. Kürten, and F. Durst. 1996. Experiments on the rise of air bubbles in clean viscous liquids. *Journal of Fluid Mechanics* 321, -1 (aug 1996), 421. https://doi.org/10.1017/s0022112096007781

[33] Shuichiro Miwa, Takahiro Moribe, Kohei Tsutstumi, and Takashi Hibiki. 2018. Experimental investigation of air entrainment by vertical plunging liquid jet. *Chemical Engineering Science* 181 (2018), 251 – 263. https://doi.org/10.1016/j.ces.2018.01.037

[34] Suhas V Patankar and D Brian Spalding. 1983. A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows. In *Numerical Prediction of Flow, Heat Transfer, Turbulence and Combustion*. Elsevier, 54–73.

[35] Stéphane Popinet. 2009. An accurate adaptive solver for surface-tension-driven interfacial flows. *J. Comput. Phys.* 228, 16 (2009), 5838–5866.

[36] Stéphane Popinet. 2018. Numerical models of surface tension. *Annual Review of Fluid Mechanics* 50 (2018), 49–75.

[37] Vivek N. Prakash, J. Martínez Mercado, Leen van Wijngaarden, E. Mancilla, Y. Tagawa, Detlef Lohse, and Chao Sun. 2016. Energy spectra in turbulent bubbly flows. *J. Fluid Mech.* 791 (feb 2016), 174–190. https://doi.org/10.1017/jfm.2016.49

[38] X.L. Qu, L. Khezzar, D. Danciu, M. Labois, and D. Lakehal. 2011. Characterization of plunging liquid jets: A combined experimental and numerical investigation. *International Journal of Multiphase Flow* 37, 7 (sep 2011), 722–731. https://doi.org/10.1016/j.ijmultiphaseflow.2011.02.006

[39] Diego Rossinelli, Babak Hejazialhosseini, Panagiotis Hadjidoukas, Costas Bekas, Alessandro Curioni, Adam Bertsch, Scott Futral, Steffen J. Schmidt, Nikolaus A. Adams, and Petros Koumoutsakos. 2013. 11 PFLOP/s Simulations of Cloud Cavitation Collapse. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC '13)*. ACM, New York, NY, USA, Article 3, 13 pages. https://doi.org/10.1145/2503210.2504565

[40] Olabanji Y. Shonibare and Kent E. Wardle. 2015. Numerical Investigation of Vertical Plunging Jet Using a Hybrid Multifluid–VOF Multiphase CFD Solver. *International Journal of Chemical Engineering* 2015 (2015), 1–14. https://doi.org/10.1155/2015/925639

[41] Álvaro Moreno Soto, Tom Maddalena, Arjan Fraters, Devaraj Van Der Meer, and Detlef Lohse. 2018. Coalescence of diffusively growing gas bubbles. *Journal of Fluid Mechanics* 846 (2018), 143–165.

[42] Wim M Van Rees, Anthony Leonard, DI Pullin, and Petros Koumoutsakos. 2011. A comparison of vortex and pseudo-spectral methods for the simulation of periodic vortical flows at high Reynolds numbers. *J. Comput. Phys.* 230, 8 (2011), 2794–2805.

[43] FH Zhang and ST Thoroddsen. 2008. Satellite generation during bubble coalescence. *Physics of Fluids* 20, 2 (2008), 022104.