A Parallel Strategy for a Level Set Simulation of Droplets Moving in a Liquid Medium

Oliver Fortmeier * and H. Martin Bücker

RWTH Aachen University, Institute for Scientific Computing,
D-52056 Aachen, Germany
{fortmeier,buecker}@sc.rwth-aachen.de

Abstract. The simulation of two-phase flow problems involving two time-dependent spatial regions with different physical properties is computationally hard. The numerical solution of such problems is complicated by the need to represent the movement of the interface. The level set approach is a front-capturing method representing the position of the interface implicitly by the root of a suitably defined function. We describe a parallel adaptive finite element simulation based on the level set approach. For freely sedimenting n-butanol droplets in water, we quantify the parallel performance on a Xeon-based cluster using up to 256 processes.

1 Introduction

Two-phase flow problems play a dominant role in various areas of computational science and engineering. Systems containing different liquids such as oil slicks in coastal waters or liquid-liquid extraction columns are illustrating examples. To develop predictive models for such extraction columns, the study of single droplets is important. At RWTH Aachen University, an interdisciplinary team of researchers from engineering, mathematics and computer science is interested in analyzing the behavior of single droplets in surrounding liquids [1, 8, 9, 15]. These flow problems involve two spatial regions that vary with time. In each of these regions, the physical properties of a material is uniformly distributed in space. The numerical simulation of two-phase flow problems is complicated by the fact that the interface between the two phases needs to be represented for the reconstruction of the interfacial movement. In front-tracking methods, the interface is explicitly represented by computational elements that follow its movement [21]. In contrast, front-capturing methods represent the interface implicitly by suitably defined functions.

The volume of fluid technique [11, 13] is a popular front-capturing method representing the interface by a function whose values are interpolated on the underlying mesh. The reconstruction of smooth interfaces, however, requires advanced interpolation techniques. The level set approach [17, 18] is another front-capturing method eliminating this drawback. In these methods, the position of

* Candidate to the Best Student Paper Award

the interface is given by the root of a scalar-valued function that splits the computational domain into two regions. An advantage of the level set approach is its elegance and simplicity to handle complicated problems involving breaking or joining regions. While serial level set approaches are widely used, parallel techniques are hardly available in the open literature. An early reference [14] carries out numerical experiments on an Intel/IPSC-860 hypercube but does not focus on parallel computing. A parallel implementation of a hybrid technique that brings together the level set approach and the volume of fluid method is reported in [19].

The approach taken in [22] to parallelize a level set technique for the simulation of two-dimensional dendritic growth is to exploit processor virtualization rather than domain decomposition. Essentially, the program decomposes the computation into a large number of objects called virtual processors which are then mapped to different physical processors. Performance results on a system with Intel Xeon-based dual-core processors are reported demonstrating a good scalability for up to 32 cores, but a degradation of the performance for 64 cores.

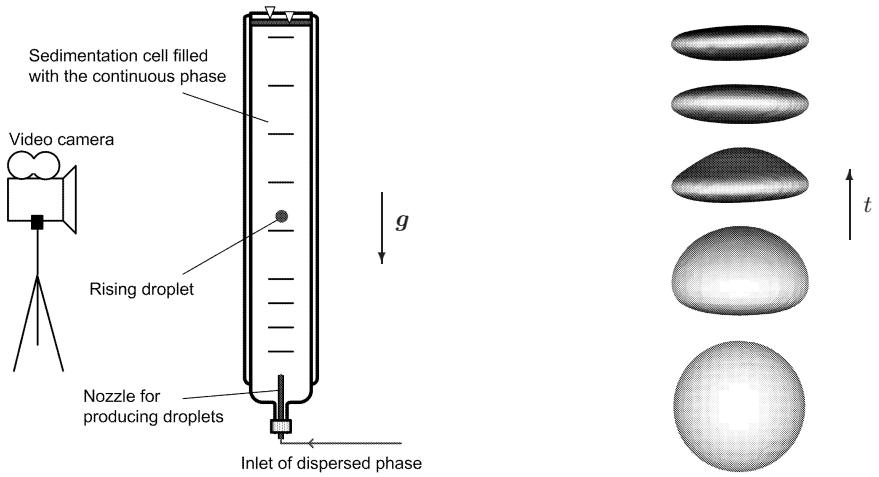
The idea behind the recent domain decomposition approach described in [10] is to generate two different grids that can be adapted independently of each other. In addition to the grid on which the flow solution is computed, this approach employs a separate Cartesian grid for tracking the movement of the interface using a level set approach. The approach is shown to scale on a cluster with Intel Xeon-based quad-core processors using up to 2048 cores.

The structure of this note is as follows. In Sect. 2, we sketch a particular two-phase flow problem which is detailed in [1]. This section also includes the underlying mathematical model. The new contributions of this note are given in Sect. 3 and 4. First, the overall strategy to parallelize the finite element method and the level set technique is described. Second, performance results of a new implementation are reported for a freely sedimenting n-butanol droplet in water with up to 256 processes.

2 Numerical simulation of droplets sedimenting in water

Throughout this article, we follow the overall setting reported in [1] in which the solvent extraction standard-test system of n-butanol droplets sedimenting in water is considered [16]. The experimental setup is schematically depicted in Fig. 1(a). The droplets are generated through a nozzle submerged in a cylindrical cell that contains the continuous phase. Upon generation, the droplet starts to accelerate upwards until it reaches its terminal sedimentation velocity which is determined by monitoring the droplet's position by a camera. A numerical simulation is carried out to predict the velocity and the deformation of different droplets over time. In Fig. 1(b), the deformation of a droplet with radius of 2 mm is illustrated. In Fig. 2, the simulated position and velocity are depicted for a droplet of the same radius showing the oscillating behavior [1].

The numerical simulation is based on the following computational model originally introduced in [20]. The incompressible Navier-Stokes equations are



(a) Sedimentation apparatus (b) Droplet deformation over time

Fig. 1. Experimental setup and simulated shape of a droplet both taken from [1].

employed to model the velocity $\boldsymbol{u}$ and the pressure p of a two-phase flow problem in a domain $\Omega \subset \mathbb{R}^3$. The phase interface, denoted by Γ, is described by the root of the scalar level set function $\phi = \phi(\boldsymbol{x}, t)$ where $\boldsymbol{x} \in \Omega$ and t denotes time. The level set function splits the domain Ω into two disjoint subdomains $\Omega_1(t) := \{\boldsymbol{x} \in \Omega \mid \phi(\boldsymbol{x}, t) < 0\}$ and $\Omega_2(t) := \{\boldsymbol{x} \in \Omega \mid \phi(\boldsymbol{x}, t) > 0\}$. Here, the droplet is represented by Ω_1 and the continuous phase by Ω_2. The effect of the surface tension τ is expressed in terms of a localized force at the interface, so-called continuum surface force [3, 7]. Combining these approaches leads to the

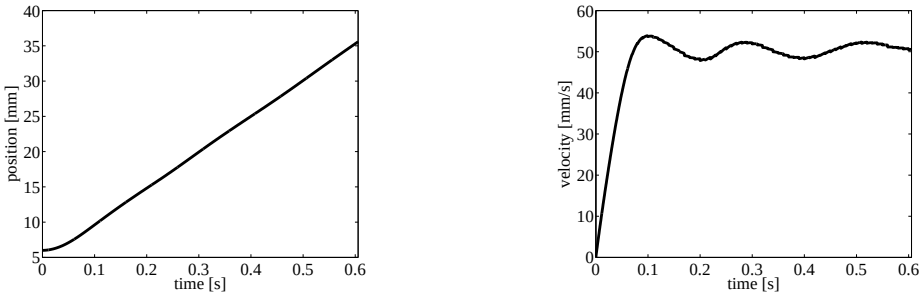


Fig. 2. Position and sedimentation velocity of an n-butanol droplet in water.

following system of partial differential equations in $\Omega \times [0,T]$:

$$\rho(\phi)\left(\frac{\partial}{\partial t}\boldsymbol{u} + (\boldsymbol{u}\cdot\nabla)\boldsymbol{u}\right) = -\nabla p + \rho(\phi)\boldsymbol{g} + \operatorname{div}\left(\mu(\phi)\mathbf{D}(\boldsymbol{u})\right) + \tau\mathcal{K}\delta_\Gamma \boldsymbol{n}_\Gamma, \quad (1)$$

$$\operatorname{div}\boldsymbol{u} = 0, \quad (2)$$

$$\frac{\partial}{\partial t}\phi + \boldsymbol{u}\cdot\nabla\phi = 0 \quad (3)$$

with appropriate boundary and initial conditions. Here, the density is given by ρ, the viscosity is denoted by μ, the strain tensor is defined by $\mathbf{D}(\boldsymbol{u}) := \nabla\boldsymbol{u} + (\nabla\boldsymbol{u})^T$, and the symbol $\mathcal{K}$ is used for the curvature. Furthermore, the Dirac function with support on Γ is denoted by δ_Γ and $\boldsymbol{n}_\Gamma$ represents the normal vector on Γ. The external gravity force is denoted by $\boldsymbol{g}$.

For the solution of the coupled problem (1)–(3), the finite element solver DROPS [5] is employed. Its main features are as follows: The three-dimensional geometry is discretized by a hierarchical tetrahedral grid accounting for adaptive refinements that change over time. Piecewise quadratic functions are used as finite element functions for the level set and the velocity. The pressure is represented by piecewise linear functions. Various Navier-Stokes, Stokes, and Krylov solvers are implemented. Time integration is based on a linear theta scheme. The level set function and the Navier-Stokes equations are decoupled by a fix-point interaction. A fast-marching variant is used to re-initialize the level set function.

3 Parallel hierarchy of triangulations

In DROPS, the hierarchical tetrahedral grid representing the computational domain Ω is decomposed and distributed to the processes. Here, we sketch the data structures used for the parallel refinement algorithm introduced in [6]. Let $\mathcal{T}_0$ denote a coarse triangulation representing the computational domain. A finer triangulation $\mathcal{T}_1$ is obtained by refining some tetrahedra of $\mathcal{T}_0$ by a red/green refinement algorithm [2]. Multiple recursive refinements of tetrahedra lead to a multi-level triangulation $\mathcal{M} = (\mathcal{T}_0, \ldots, \mathcal{T}_{k-1})$, where the triangulation $\mathcal{T}_{k-1}$ represents the finest triangulation. The multi-level triangulation is admissible if the following conditions hold for all levels $l \in \{1, \ldots, k-1\}$:

1. A tetrahedron $T \in \mathcal{T}_l$ is either in $\mathcal{T}_{l-1}$ or is obtained by a refinement of a tetrahedron $T' \in \mathcal{T}_{l-1}$.
2. If $T \in \mathcal{T}_{l-1} \cap \mathcal{T}_l$ then $T \in \mathcal{T}_{l+1}, \ldots, \mathcal{T}_{k-1}$. That is, if the tetrahedron T is not refined then it stays unrefined.

Due to these two conditions, we can assign each tetrahedron T a unique level

$$l(T) := \min\{m \mid T \in \mathcal{T}_m\}.$$

The set of all tetrahedra on level l is denoted by $\mathcal{G}_l$ and is called the hierarchical surplus. These hierarchical surpluses define the hierarchical decomposition

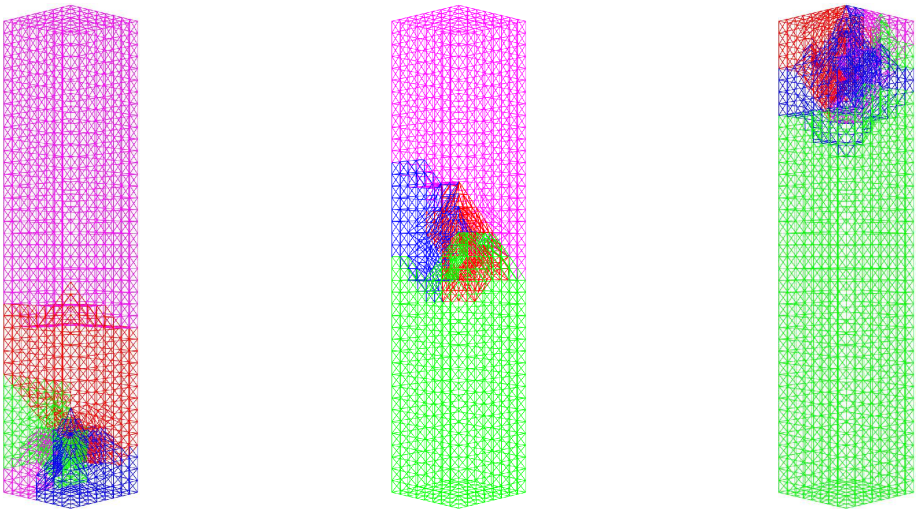
$\mathcal{H} = (\mathcal{G}_0, \ldots, \mathcal{G}_{k-1})$. Since each tetrahedron is only located in one hierarchical surplus this decomposition is used to efficiently store all tetrahedra. The equations (1)–(3) are solved on the finest triangulation $\mathcal{T}_{k-1}$. Therefore, the domain decomposition approach is based on distributing the tetrahedra of $\mathcal{T}_{k-1}$ among the processes and leads to a decomposition of the hierarchical surplus $\mathcal{H}$.

In general, repeated local grid modifications lead to large differences in the number of tetrahedra stored on each process. Therefore, a load balancing algorithm is implemented consisting of three steps:

1. The triangulation $\mathcal{T}_{k-1}$ is described by a weighted graph $G = (V, E, \varrho_V, \varrho_E)$. Each node $v \in V$ represents a set of tetrahedra. Each edge $(v, w) \in E$ corresponds to a pair of tetrahedron sets v and w that have at least a face in common. The number of common faces between the tetrahedron sets determines the edge weight $\varrho_E(v, w)$. The vertex weight $\varrho_V(v)$ is given by the number of tetrahedra represented by v. The underlying graph model is discussed in more detail in [4].
2. The weighted graph is partitioned into P parts, where P denotes the number of processes. This leads to a partition of the vertices $V = V_1 \cup \cdots \cup V_P$ where $V_i \cap V_j = \emptyset$ for all $i \neq j$. The library ParMetis [12] partitions the graph in an attempt to minimize the number of edges between V_i and V_j with $i \neq j$ while balancing the sum of the vertex weights in each V_i. In particular, a routine is used that computes a new partition based on a previous one. This is of special interest for solving two-phase problems, since the triangulation $\mathcal{T}_{k-1}$ changes only slightly by performing a time step of the simulation.
3. The final step consists of migrating tetrahedra and its vertices, edges and faces among the processes. After the migration, each process $p = 1, \ldots, P$ stores the tetrahedra represented by the subset V_p. Note that the migration has to take into account not only the tetrahedra in $\mathcal{T}_{k-1}$ but also on other levels so that the complete hierarchy is still admissible.

Since most physical effects occur in the vicinity of the phase interface Γ we refine the grid in this subdomain leading to a high resolution close to Γ. While evolving in simulation time, the location of Γ changes and demands for modifying the finest triangulation. However, this can again imply a large difference in the number of tetrahedra stored on each process. To avoid this imbalance the three load balancing steps are performed each time the grid changes.

DROPS uses piecewise linear and quadratic finite element functions. The corresponding degrees of freedom (DOF) are located at vertices and edges of $\mathcal{T}_{k-1}$. So, the migration algorithm handles not only geometric but also DOF data. Therefore, the non-overlapping decomposition of the tetrahedra in $\mathcal{T}_{k-1}$ leads to a distribution of DOF among the processes. Vertices and edges may be stored by multiple processes if they are located at a process boundary. Hence, the corresponding DOF are stored on multiple processes as well. Linear algebra operations on these DOF require updating mechanisms which include communication between neighboring processes. This communication is overlapped by computation. The systems of linear equations resulting from linearizing and discretizing (1)–(3) are iteratively solved by Krylov subspace methods which involve linear



(a) Initial (b) Middle (c) End

Fig. 3. Decomposition of a triangulation among 4 processes.

algebra operations such as matrix vector products, inner products, and vector updates. These linear algebra operations are rearranged to minimize the number of communications by re-using updated DOF.

4 Parallel performance

In this section, we present results of sedimenting n-butanol droplets in water. In [1], droplets of initial radius from 0.5 mm to 2.0 mm are investigated in a brick-shaped domain. However, in this note, we are interested in studying larger problems by simulating droplets of radius 2 mm and 3mm. In Fig. 3(a), the initial distribution of the tetrahedra of the finest triangulation is illustrated. In Fig. 3(b) and 3(c), the distribution of tetrahedra is depicted in the middle and at the end of the simulation time, respectively. These simulations are performed on a triangulation which is 4 or 5 times recursively refined in the vicinity of the phase boundary. That is, each tetrahedron whose barycenter is located at a radius of 0.4 mm around the phase boundary is recursively refined into eight sub-tetrahedra. In Table 1, the problem sizes in terms of number of tetrahedra and DOF are presented.

All experiments are performed on a cluster of Xeon-based quad-core processors (E5450) at the center for computing and communication at RWTH Aachen University. Each node of the cluster consists of two quad-core processors which share one InfiniBand network card for inter-node communication. An Intel implementation of MPI handles the communication between the processes.

To present the performance results, we distinguish between two different placements of MPI processes on the quad-core processors: a compact strategy (*Comp*), and a scatter strategy (*Scat*). In *Comp*, one MPI process is placed on each core of the processors leading to eight MPI processes per node. In *Scat*, only a single MPI process is located on each processor implying idle cores and two processes per node. The choice of the strategy has a big impact on the time to perform one time step. For example, while simulating a 2 mm drop with 4 levels of refinement, switching from *Scat* to *Comp* for 8 processes increases the runtime by a factor of 3.22. The major reason for this effect seems to be given by the limited memory bandwidth. Since the underlying data structures represent unstructured grids the access pattern to the memory is not consecutive resulting in unstructured memory access. Hence, the cache hierarchy of the processors cannot be exploited. Additionally, the network card is likely to serve two MPI processes more efficiently than eight processes.

In Fig. 4 the speedup of four test cases is shown. In this figure, the speedup for P processes is defined by

$$S_p(P) = \frac{p \cdot T(p)}{T(P)},$$

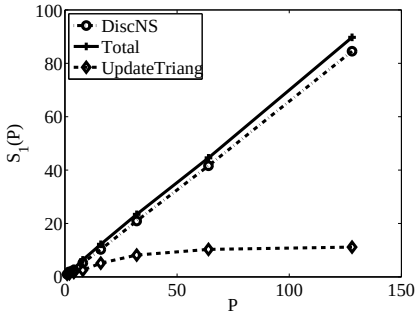
where $T(P)$ denotes the runtime on P processes and p denotes the smallest number of processes used to perform the corresponding simulation. That is, we assume perfect speedup while using p processes. These plots show the parallel performance of a single time step for the solution of the two-phase flow problem. The time required by a time step is denoted by *Total*. The update of the triangulation including grid modifications and load balancing are referred to as *UpdateTriang*. The discretization of the system (1)–(2) is denoted by *DiscNS* The remaining time required by a time step dominates *Total* and is not shown separately in these plots. The total time scales well up to 128 and 256 processes using the *Scat* or *Comp* strategy, respectively.

The updating does not scale as good as the other parts. In Fig. 4(b), the relative time spent in updating the triangulation w.r.t. the total time is 0.24 % on one process and 1.35 % on 128 processes. In Fig. 4(c), this relative time increases from 0.36 % on 8 processes to 1.47 % on 256 processes. In this scenario of a sedimenting droplet and in typical simulation, the updating is performed about every tenth discrete time step. Therefore, this updating part of the simulation is currently not a major bottleneck.

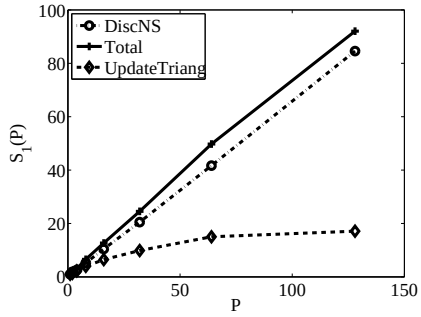
For the total time, using 128 processes and *Scat* leads to a speedup of $S_1(128) \approx 90$ for both radii and four refinements as presented in Fig. 4(a) and

Radius [mm]	Refinements	tetrahedra	velocity DOF	pressure DOF	level set DOF
2	4	155 132	525 756	22 485	178 101
2	5	876 776	3 006 030	126 190	1 004 859
3	4	328 472	1 102 782	46 799	370 443

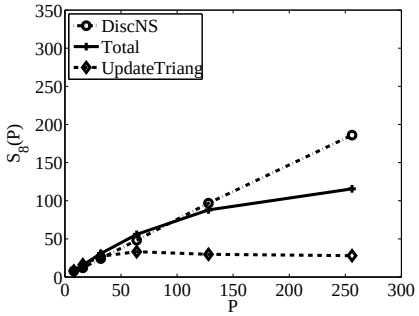
Table 1. Problem size in terms of number of tetrahedra and DOF.



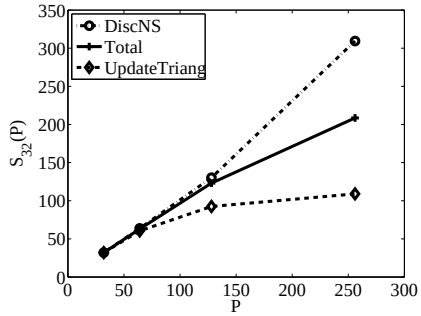
(a) *Scat*, radius 2 mm, refinements 4



(b) *Scat*, radius 3 mm, refinements 4



(c) *Comp*, radius 2 mm, refinements 4



(d) *Comp*, radius 2 mm, refinements 5

Fig. 4. Parallel performance of a single time step.

(b). Here, the serial execution time for simulating a 3 mm droplet is decreased from 8801 s to 96 s while using 128 processes. The serial time for updating the triangulation takes 21 s. On 128 processes, modifying the triangulation including the load-balancing steps takes 1.3 s. Figure 4(c) and (d) illustrate that for *Comp* the speedup is given by $S_8(256) \approx 116$ and $S_{32}(256) \approx 209$ when simulating a 2 mm droplet with four and five refinements, respectively. The execution time in Fig. 4(c) decreases from 1713 s on eight processes to 118 s on 256 processes. If simulating a droplet of radius 3 mm with four refinements, the total time of 4508 s on eight processes is reduced to 119 s on 256 processes, whereas the time for updating decreases from 9.6 s to 1.7 s.

5 Concluding remarks

The strategy to parallelize the three-dimensional computational fluid dynamics software DROPS is presented. This software employs a unique combination of discretization on tetrahedral grids using finite elements, local grid refinement techniques, and level set methods for interface capturing [5]. The parallelization

consists of decomposing the computational domain on the finest level of the hierarchy of triangulations. Load balancing is addressed via graph partitioning. In [1], the parallel simulation of a sedimenting n-butanol droplet in water is validated by comparing the numerical results with empirical models as well as with actual experimental measurements. In contrast to [1] where parallel computing is only briefly mentioned, the focus of the present note is on the parallelization strategy and the resulting performance on a cluster of Xeon-based quad-core processors. The reported parallel performance is good, but not excellent. The bottleneck in the scalability is shown to be the update of the triangulation. Moreover, there is room for introducing a more refined graph model for load balancing in [4] and more scalable approaches to re-initialize the level set function.

Acknowledgments

We thank our collaborators within SFB 540 "Model-based experimental analysis of kinetic phenomena in fluid multi-phase reactive systems" which is supported by the Deutsche Forschungsgemeinschaft (DFG). The Aachen Institute for Advanced Study in Computational Engineering Science (AICES) provides a stimulating research environment for our work.

References

1. Bertakis, E., Groß, S., Grande, J., Fortmeier, O., Reusken, A., Pfennig, A.: Validated simulation of droplet sedimentation with finite-element and level-set methods. Chemical Engineering Science 65(6), 2037–2051 (2010)
2. Bey, J.: Simplicial grid refinement: On Freudenthal's algorithm and the optimal number of congruence classes. J. Numer. Math. 85(1), 1–29 (2000)
3. Brackbill, J.U., Kothe, D.B., Zemach, C.: A continuum method for modeling surface tension. J. Comput. Phys. 100(2), 335–354 (1992)
4. Fortmeier, O., Henrich, T., Bücker, H.M.: Modeling data distribution for two-phase flow problems by weighted graphs. In: Beigl, M., Cazorla-Almeida, F.J. (eds.) 23rd Workshop on Parallel Sytems and Algorithms, Hannover, Germany, Februar 12, 2010. pp. 31–38. VDE (2010)
5. Groß, S., Reichelt, V., Reusken, A.: A finite element based level set method for two-phase incompressible flows. Comput. Vis. Sci. 9(4), 239–257 (2006)
6. Groß, S., Reusken, A.: Parallel multilevel tetrahedral grid refinement. SIAM J. Sci. Comput. 26(4), 1261–1288 (2005)
7. Groß, S., Reusken, A.: Finite element discretization error analysis of a surface tension force in two-phase incompressible flows. SIAM J. Numer. Anal. 45(4), 1679–1700 (2007)
8. Gross-Hardt, E., Amar, A., Stapf, S., Pfennig, A., Blümich, B.: Flow dynamics inside a single levitated droplet. Ind. Eng. Chem. Res. 1, 416–423 (2006)
9. Gross-Hardt, E., Slusanschi, E., Bücker, H.M., Pfennig, A., Bischof, C.H.: Practical Shape Optimization of a Levitation Device for Single Droplets. Opt. Eng. 9(2), 179–199 (2008)

10. Herrmann, M.: A parallel Eulerian interface tracking/Lagrangian point particle multi-scale coupling procedure. J. Comput. Phys. 229(3), 745–759 (2010)
11. Hirt, C., Nichols, B.: Volume of fluid (VOF) method for the dynamics of free boundaries. J. Comput. Phys. 39(1), 201–225 (1981)
12. Karypis, G., Kumar, V.: A parallel algorithm for multilevel graph partitioning and sparse matrix ordering. J. Parallel Distrib. Comput. 48(1), 71–95 (1998)
13. Li, J., Renardy, Y.: Numerical study of flows of two immiscible liquids at low reynolds number. SIAM Rev. 42(3), 417–439 (2000)
14. Li, X.L.: Study of three-dimensional Rayleigh–Taylor instability in compressible fluids through level set method and parallel computation. Phys Fluids A-Fluid 5(8), 1904–1913 (1993)
15. Marquardt, W.: Model-based experimental analysis of kinetic phenomena in multi-phase reactive systems. Trans. Inst. Chem. Eng. 83(A6), 561–573 (2005)
16. Misek, T., Berger, R., Schröter, J.: Standard test systems for liquid extraction. No. 46 in Europ. Fed. Chem. Eng. Pub. Ser., Inst. Chem. Eng., Warwickshire, 2nd edn. (1985)
17. Osher, S., Sethian, J.A.: Fronts propagating with curvature dependent speed: Algorithms based on Hamilton-Jacobi formulations. J. Comput. Phys. 79(1), 12–49 (1988)
18. Sethian, J.A.: Level Set Methods and Fast Marching Methods—Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science. Cambridge University Press, 2 edn. (June 1999)
19. Sussman, M.: A parallelized, adaptive algorithm for multiphase flows in general geometries. Comput. Struct. 83(6-7), 435–444 (2005)
20. Sussman, M., Smereka, P., Osher, S.: A level set approach for computing solutions to incompressible two-phase flow. J. Comput. Phys. 114(1), 146–159 (1994)
21. Tryggvason, G., Bunner, B., Esmaeeli, A., Juric, D., Al-Rawahi, N., Tauber, W., Han, J., Nas, S., Jan, Y.J.: A front-tracking method for the computations of multiphase flow. J. Comput. Phys. 169(2), 708–759 (2001)
22. Wang, K., Chang, A., Kale, L.V., Dantzig, J.A.: Parallelization of a level set method for simulating dendritic growth. J. Parallel Distrib. Comput. 66(11), 1379–1386 (2006)