Numerical simulations of free surface flows on adaptive cartesian grids with the level set function method [*]

Kirill D. Nikitin[†] Maxim A. Olshanskii[‡] Kirill M. Terekhov[§] Yuri V. Vassilevski[¶]

Abstract

The paper studies a method for numerical simulation of free surface flows of viscous incompressible fluids. The approach is based on the level set method for capturing free surface evolution and features compact finite difference approximations of fluid and level set equations on locally refined and dynamically adapted octree cartesian grids. We consider in detail an extension of staggered grid stable approximation of the Navier-Stokes and level-set equations for the case of evolving free boundary and dynamically refined / coarsened octree meshes. The problem of surface reconstruction and preserving distance property of the discrete level set function is addressed. Several re-initialization techniques are introduced, compared, and their impact on the accuracy of flow computation is considered. Numerical examples include analytical tests, the breaking dam, and the oscillating droplet benchmark problems. It is demonstrated that altogether the considered techniques and algorithms result in an accurate and efficient approach to modeling incompressible viscous flows with free surfaces.

1 Introduction

Examples of flows with free surfaces include sea waves, rain drops, and falling thin films, to mention a few. Modeling such phenomena numerically is a challenging task due to the non-trivial coupling of flow dynamics and free surface evolution. Substantial progress has been made during the last two decades in developing efficient and accurate numerical methods for computing flows with free surfaces and interfaces, see e.g. [44, 45] and references therein. Possible approaches include algorithms based on explicit surface tracking [51, 50] and implicit surface capturing techniques [37, 47], both finite difference [21], finite volume [19] and finite element [3, 4] methods have been employed as discretization techniques. Since important physical phenomena often happen in a vicinity of the free boundary and the surface may undergo complex topological changes including formation of singularities, using grids adaptively refined near the free surface is a common practice, e.g. [5, 19]. Most of the adaptive methods are based on locally refined triangulations (tetrahedra) and finite element discretizations, e.g. [5, 14], which enable one to handle complex geometries emerging in the process of the free surface evolutions. While adaptive finite element methods now constitute a well developed approach, its application for flows with evolving free surfaces leads to algorithms where substantial effort is required for data handling and mesh reconstruction. Adaptive cartesian grids are much more convenient for dynamic refining / coarsening procedures. Thus, it is no surprise that octree cartesian grids became a standard choice in image processing [49] and other applications where non-trivial geometries occur [31]. In particular, the apparent success of octree grids in the visualization of smoke and water [29] inspired our own studies of fluid solvers based on such meshes. Discretizations on octree (quadtree) cartesian grids already enjoyed an employment in free surface flow computations [32, 34, 39, 46], yet many aspects of using octree grids for accurate and predictive computations of free surface flow require more thorough study. In particular, the accuracy validation of octree based free surface 3D flow solvers on

[*]This work has been supported in part by RFBR grants 08-01-00159-a, 09-01-00115-a and the federal program "Scientific and scientific-pedagogical personnel of innovative Russia".

[†]Institute of Numerical Mathematics, Russian Academy of Sciences, Moscow; nikitin.kira@gmail.com

[‡]Department of Mechanics and Mathematics, Moscow State University, Moscow; Maxim.Olshanskii@mtu-net.ru; www.mathcs.emory.edu/~molshan

[§]Institute of Numerical Mathematics, Russian Academy of Sciences, Moscow; kirill.terekhov@gmail.com

[¶]Institute of Numerical Mathematics, Russian Academy of Sciences, Moscow; yuri.vassilevski@gmail.com

standard benchmark test problems is still largely missing. Our objective is to develop the approach of [34] and build an efficient algorithm based on compact finite difference discretizations on octree cartesian grids which provides simple data handling and dynamic re-meshing, is suitable for multi-processor implementation, and still is accurate enough to recover physics of 3D free-surface flows.

The conventional mathematical model of incompressible viscous Newtonian flow with a free surface is based on the Navier-Stokes equations coupled with the level set function equation, e.g. [45]. The model is reviewed in section 2. Besides well-known difficulties of incompressible viscous flow calculations, treating free surface flows brings additional complications. The latter include (i) Finding the domain where the fluid equations are posed, is a part of the solution process; (ii) The free surface may undergo complex topological changes, such as merging or pinching of two fronts. Resolving such evolutions requires dynamically refined/coarsened grids; (iii) For accurate and stable computations, the level set (indicator) function should preserve the property of approximate signed distance; (iv) The accurate modeling of surface tension forces and rendering requires a suitable approximation of free surface normal vector and local curvature.

The approach studied in this paper combines a time splitting algorithm with a mesh adaptation. The splitting scheme decouples each time step into separate substeps of computing velocity, pressure and the level set function. For the sake of adaptation, we use graded octree cartesian grids which are adapted towards the free boundary. When the free surface evolves, the grid is dynamically refined or coarsened according to the distance to the free boundary. Finite difference approximations with compact stencils are used to discretize divergence, gradient and Laplace operators. Particular attention is paid to the following important ingredients of the algorithm: preserving the distance property of the discrete level set functions, and approximation of normal vectors and curvatures of the free surface. The distance property is recovered by solving the discrete Eikonal equation (so called re-initialization). We introduce two re-initialization methods based on the marching cubes algorithm for free surface triangulation, compare several Eikonal solvers and study their impact on the accuracy of fluid problem solution. A method of particles [12] is considered as another approach to enhancing accuracy. The paper discusses benefits as well as limitations of these technologies and the entire approach.

The remainder of the paper is organized as follows. Section 2 reviews the mathematical model. In section 3 we discuss the splitting algorithm for numerical time integration of the coupled system of the Navier-Stokes and level set function equations. A finite difference method for space discretization is considered in section 4. In section 5 we focus on re-initialization methods for the level set function. Numerical results for several test problems including the Zalesak's disk, the breaking dam, and the freely oscillating droplet benchmark problems are presented in section 6. Section 7 contains some closing remarks.

2 Mathematical model

We consider a Newtonian incompressible fluid flow in a bounded time-dependent domain $\Omega(t) \in \mathbb{R}^3$. We assume that $\overline{\partial\Omega(t)} = \overline{\Gamma_D} \cup \overline{\Gamma(t)}$, where Γ_D is the static boundary[1] (walls) and $\Gamma(t)$ is a free surface. In the time interval $(0, T]$, the fluid flow is governed by the incompressible Navier-Stokes equations

$$\left\{ \begin{aligned} \rho\left(\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u}\right) - \nu\Delta\mathbf{u} + \nabla p = \mathbf{f} \\ \nabla\cdot\mathbf{u} = 0 \end{aligned} \right. \quad \text{in } \Omega(t), \tag{1}$$

where $\mathbf{u}$ is the vector velocity field, p is the kinematic pressure, $\mathbf{f}$ is the external force (e.g., gravity), ρ is the density, and ν is the kinematic viscosity. At the initial time $t = 0$ the domain and the velocity field are known:

$$\Omega(0) = \Omega_0, \quad \mathbf{u}|_{t=0} = \mathbf{u}_0. \tag{2}$$

On the static part of the flow boundary we assume the velocity field satisfies Dirichlet boundary condition

$$\mathbf{u} = \mathbf{g} \quad \text{on } \Gamma_D, \tag{3}$$

[1]The Γ_D part of the boundary may vary in time, although remaining static, see e.g. the breaking dam problem from Sec. 6.4.

$\mathbf{g}$ is given. On the free surface $\Gamma(t)$, we impose the kinematic condition

$$v_\Gamma = \mathbf{u}|_\Gamma \cdot \mathbf{n}_\Gamma \tag{4}$$

where $\mathbf{n}_\Gamma$ is the normal vector for $\Gamma(t)$ and v_Γ is the normal velocity of the free surface $\Gamma(t)$. Balancing the surface tension and stress forces yields the second condition on $\Gamma(t)$:

$$\boldsymbol{\sigma}\mathbf{n}_\Gamma|_\Gamma = \tau\kappa\mathbf{n}_\Gamma - p_{\text{ext}}\mathbf{n}_\Gamma \quad \text{on } \Gamma(t), \tag{5}$$

where $\boldsymbol{\sigma} = \nu[\nabla\mathbf{u} + (\nabla\mathbf{u})^T]/2 - p\,\mathbf{I}$ is the stress tensor of the fluid, κ is the sum of the principal curvatures, τ is the surface tension coefficient, p_{ext} is an exterior pressure which we assume to be zero, $p_{\text{ext}} = 0$.

Existing numerical approaches to the numerical solution of (1)-(5) can be roughly divided into two groups: methods based on surface tracking and those which involve surface capturing. Free surface tracking algorithms are based on the surface evolution equation (4). We employ surface capturing algorithms based on the implicit definition of $\Gamma(t)$ as the zero level of a globally defined function $\phi(t,\mathbf{x})$. A smooth (at least Lipschitz continuous) function ϕ such that

$$\phi(t,\mathbf{x}) = \begin{cases} < 0 & \text{if } \mathbf{x} \in \Omega(t) \\ > 0 & \text{if } \mathbf{x} \in \mathbb{R}^3 \setminus \overline{\Omega(t)} \\ = 0 & \text{if } \mathbf{x} \in \Gamma(t) \end{cases} \qquad \text{for all } t \in [0,T]$$

is called the *level set* function. The initial condition (2) allows us to define $\phi(0,\mathbf{x})$. For $t > 0$ the level set function satisfies the following transport equation [36]:

$$\frac{\partial\phi}{\partial t} + \widetilde{\mathbf{u}} \cdot \nabla\phi = 0 \quad \text{in } \mathbb{R}^3 \times (0,T] \tag{6}$$

where $\widetilde{\mathbf{u}}$ is any smooth velocity field such that $\widetilde{\mathbf{u}} = \mathbf{u}$ on $\Gamma(t)$. The employed mathematical model consists of equations (1), (2), (3), (5), and (6). We note that the implicit definition of $\Gamma(t)$ as zero level of a globally defined function ϕ leads to numerical algorithms which can easily handle complex topological changes of the free surface. The level set function provides an easy access to useful geometric characteristics of $\Gamma(t)$. For instance, the unit outward normal to $\Gamma(t)$ is $\mathbf{n}_\Gamma = \nabla\phi/|\nabla\phi|$, and the surface curvature is $\kappa = \nabla \cdot \mathbf{n}_\Gamma$. From the numerical point of view, it is often beneficial if the level set function possesses the signed distance property, i.e. it satisfies the Eikonal equation

$$|\nabla\phi| = 1. \tag{7}$$

3 Numerical time integration

Various numerical methods have been proposed for the time integration of (1) and (6), ranging from fully implicit schemes to fractional steps methods. Fully implicit schemes provide unconditional stability by the expense of several nested iterative processes [20]. Here we apply a semi-implicit spitting method that avoids nested iteration loops and thus possesses time step restrictions. The algorithm is built on the well-known splitting procedures, see [7, 36].

Each time step of the method (given $\mathbf{u}(t)$, $p(t)$, $\phi(t)$ find approximations to $\mathbf{u}(t+\Delta t)$, $p(t+\Delta t)$, $\phi(t+\Delta t)$) consists of the following substeps. For the sake of presentation simplicity, we suppress spacial discretization details in this section. The spacial discretization of all involved operators will be introduced in the next section.

Level set part: $\Omega(t) \to \Omega(t+\Delta t)$

1. Extend velocity to the exterior of fluid body: $\mathbf{u}(t)|_{\Omega(t)} \to \widetilde{\mathbf{u}}(t)|_{\mathbb{R}^3}$. The particular extension operator is defined for discrete velocities and is given in section 4.3. In practice, the extension is performed to a bulk computational domain, rather than $\mathbb{R}^3$.

2. Find $\phi(t+\Delta t)$ from (6) by a numerical integration with a semi-Lagrangian method [48] and using the extended velocity field. The semi-Lagrangian method consists of the following substeps. First, for every grid point $\mathbf{y}$, solve the characteristic equation backward in time

$$\frac{\partial\mathbf{x}(\tau)}{\partial\tau} = \widetilde{\mathbf{u}}(\mathbf{x}(\tau),\tau), \quad \mathbf{x}(t+\Delta t) = \mathbf{y}, \quad \text{for } \tau \in [t+\Delta t, t]. \tag{8}$$

The characteristic equation is integrated numerically with the second order accuracy:

$$\mathbf{x}(t+\frac{\Delta t}{2}) = \mathbf{y} - \frac{\Delta t}{2}\widetilde{\mathbf{u}}(\mathbf{y},t), \quad \mathbf{x}(t) = \mathbf{x}(t+\frac{\Delta t}{2}) - \frac{\Delta t}{2}\left[3\widetilde{\mathbf{u}}(\mathbf{x}(t+\frac{\Delta t}{2}),t) - \widetilde{\mathbf{u}}(\mathbf{x}(t+\frac{\Delta t}{2}),t-\Delta t)\right]. \quad (9)$$

The space point $\mathbf{x}(t+\frac{\Delta t}{2})$ is not necessarily a grid point, therefore to compute $\widetilde{\mathbf{u}}(\mathbf{x}(t+\frac{\Delta t}{2}),t)$ and $\widetilde{\mathbf{u}}(\mathbf{x}(t+\frac{\Delta t}{2}),t-\Delta t)$ an interpolation of velocity values should be applied. This interpolation procedure is described in section 4.3, where a spacial discretization is introduced.
Second, assign

$$\phi^*(\mathbf{y},t+\Delta t) = \phi(\mathbf{x}(t),t). \quad (10)$$

Again, to compute $\phi(\mathbf{x}(t),t)$ in (10) the interpolation is used. Note that at this step the signed distance property of ϕ and the volume balance may be lost.

3. Perform the correction $\phi^*(t+\Delta t) \to \widehat{\phi}^*(t+\Delta t)$ in order to enforce the global volume conservation as described in section 5.1;

4. Re-initialize the level set function $\widehat{\phi}^*(t+\Delta t) \to \phi(t+\Delta t)$ so that $\phi(t+\Delta t)$ (approximately) satisfies (7). We introduce and study several re-initialization procedures in section 5.2.

The "level set" part of our splitting algorithm is now complete. The computed $\phi(t+\Delta t)$ implicitly defines the new fluid domain $\Omega(t+\Delta t)$. Given the new fluid domain we update the grid.

Remeshing. Adapt the grid accounting for the new position of the free surface and re-interpolate all discrete variables to the new grid. The detail of the remeshing procedure is given in section 4.1.

Fluid part: $\{\mathbf{u}(t),p(t)\} \to \{\mathbf{u}(t+\Delta t),p(t+\Delta t)\}$. We find we new velocity and pressure in several steps. Fist we perform a pure advection step by the semi-Lagrangian method, next we add viscous terms, and finally we project the velocity into (discretely) divergence-free functions subspace and recover new pressure:

1. For each velocity component u_k, $k=1,2,3$, we apply the semi-Lagrangian method similar to the case of the level set function as described above. The only differences are the following: $\mathbf{y}$ denotes now not a cell vertex, but a node where particular velocity component is defined, and (10) is replaced by

$$u_k^*(\mathbf{y},t+\Delta t) = u_k(\mathbf{x}(t),t). \quad (11)$$

Again, to compute $u_k(\mathbf{x}(t),t)$ in (11) the interpolation is used.

2. The viscous step:

$$\widehat{\mathbf{u}}^*(t+\Delta t) = \mathbf{u}^*(t+\Delta t) + \Delta t\left[\nu\Delta\mathbf{u}(t) + \mathbf{f}(t)\right].$$

3. The projection step: Solve for pressure $p(t+\Delta t)$:

$$\begin{cases} \nabla\cdot\nabla p(t+\Delta t) = \dfrac{1}{\Delta t}\nabla\cdot\widehat{\mathbf{u}}^*(t+\Delta t) \quad \text{in } \Omega(t+\Delta t), \\ p(t+\Delta t) = \tau\kappa(t+\Delta t) \quad \text{on } \Gamma(t+\Delta t) \quad \text{and} \quad \dfrac{\partial p(t+\Delta t)}{\partial\mathbf{n}} = 0 \quad \text{on } \Gamma_D. \end{cases} \quad (12)$$

Update velocity

$$\mathbf{u}(t+\Delta t) = \widehat{\mathbf{u}}^*(t+\Delta t) - \nabla p(t+\Delta t).$$

Time step adaptation. Choose the new time step Δt satisfying the CFL-condition:

$$[\Delta t]_{\text{new}} = C_{\text{cfl}} h_{\min}\left(\max_{\mathbf{x}\in\Omega(t+\Delta t)}|\mathbf{u}(t+\Delta t)|\right)^{-1},$$

where C_{cfl} is an application dependent parameter. In our numerical experiments we choose $C_{\text{cfl}} = \frac{10}{11}$. The time step restriction due to the explicit viscous step is not imposed since the coefficient ν is assumed to be small enough.

Goto the level set part.

Remark 3.1 It was found beneficial in some cases to integrate (8) more accurately, dividing the time interval $[t, t + \Delta t]$ on n subintervals and applying (9) on each subinterval. Using $n = 2, 3$ we observe a notable overall accuracy improvement comparing to $n = 1$.

4 Spacial discretization and grid adaptation

4.1 Octree grid and adaptation

A possibly complex geometry of free surface and the accurate approximation of surface tension forces require a sufficiently fine grid in a neighborhood of $\Gamma(t)$. In this case, the use of uniform grids becomes prohibitively expensive, especially in 3D. Locally refined meshes often need considerably less computational resources. However, such meshes have to be dynamically refined and coarsened if the free surface evolves. The remeshing is, in general, CPU time and memory demanding procedure for consistent regular tetrahedrizations. This step becomes considerably less expensive if one uses cartesian octree meshes with cubic cells. The two-dimensional analog of an octree mesh refined towards free surface is illustrated in Figure 1. More details on quadtree/octree data structures can be found in [41, 42]. The use of cubic cells is also appealing due to the economic staggered distribution of velocity degrees of freedom as well as straightforward data interpolation between two consecutive meshes, see below. Moreover, due to the inherited hierarchical structure, efficient multilevel techniques can be adopted for the solution of linear systems, e.g. [2, 6, 11].

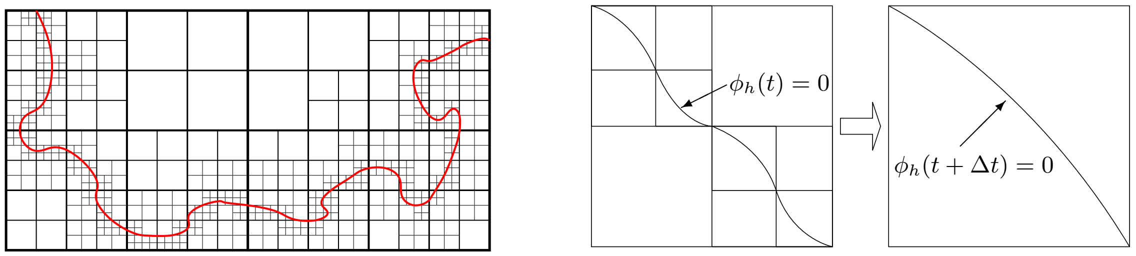


Figure 1: Left: 2D quadtree grid adapted to free boundary. Right: The loss of discrete free surface geometric information when ϕ_h is transported from a region with finer mesh to the one with a coarser mesh

Our adaptation strategy is based on the graded refinement (the sizes of two neighboring cells may differ at most by the factor of two) of the mesh towards the *current and predicted* location of the free surface. By the predicted location at time t we mean the one occupied by $\Gamma(t + \Delta t)$ if the characteristic equation (8) is solved with *current* velocity and Δt. The reason for grid refinement towards the predicted interface location is to reduce the loss of the local surface geometric information which occurs if $\Gamma(t + \Delta)$ is approximated by a trilinear function on a coarser grid; such possible loss is illustrated in Figure 1. Note, that the predicted location may slightly differ from the actually computed $\Gamma(t + \Delta t)$ in the level set part of the algorithm, since the mesh adaptation step is performed *before* the velocity and Δt are updated in the fluid part of the algorithm. However, this allows us to preserve most of the local surface geometry and avoids double remeshing. In our numerical experiments all cells intersected by $\Gamma(t)$ or $\Gamma(t + \Delta t)$ have the same width $h_{\min}$. Away from these two surfaces the mesh is aggressively coarsened up to the maximum cell width $h_{\max}$ in the fluid domain $\Omega(t)$ and h_{ext} in the rest of computational domain.

4.2 Spacial discretization

To produce a stable approximation we use a staggered location of velocity and pressure unknowns to discretize the fluid equations [21, 26] (see Figure 2): The pressure is approximated in cell centers, velocity components are approximated in face centers. The level set function is approximated in cell vertices.
Discrete gradient. For every internal cell face from the octree mesh Ω_h we define a corresponding component of the pressure gradient. For example, we approximate p_x in velocity u-nodes. Since we use graded octree meshes,

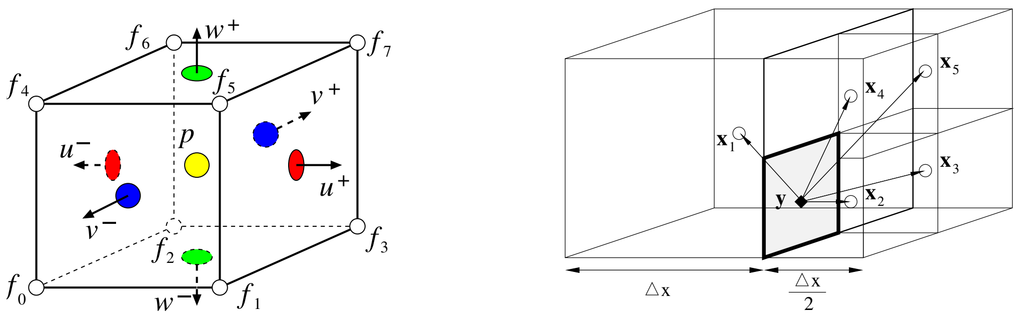


Figure 2: Left: Location of variables in staggered grid; p is pressure, $\{u^{\pm}, v^{\pm}, w^{\pm}\}$ are velocity components, f is nodal scalar function, e.g. level set function. Right: Discretization stencil for $\partial p/\partial x$.

for any interior cell face there can be only two geometric cases. In the first case, the face is shared by two equal cells, and we use the standard central finite difference to approximate the corresponding gradient component. In the second case, the sizes of the cells sharing the face are different. For the sake of simplicity, we show the discretization of the x-component of the gradient operator at the face center $\mathbf{y}$, see Fig. 2. To this end, consider the centers of five surrounding cells $\mathbf{x}_1, \dots, \mathbf{x}_5$ and expand the pressure value $p(\mathbf{x}_i)$ with respect to $p(\mathbf{y})$:

$$p(\mathbf{x}_i) = p(\mathbf{y}) + \nabla p(\mathbf{y}) \cdot (\mathbf{x}_i - \mathbf{y}) + O(|\mathbf{x}_i - \mathbf{y}|^2).$$

Neglecting the second-order term we obtain a system of 5 linear equations with 4 unknowns

$$\begin{pmatrix} 1 & -\frac{\Delta}{2} & \frac{\Delta}{4} & \frac{\Delta}{4} \\ 1 & \frac{\Delta}{4} & 0 & 0 \\ 1 & \frac{\Delta}{4} & \frac{\Delta}{2} & 0 \\ 1 & \frac{\Delta}{4} & 0 & \frac{\Delta}{2} \\ 1 & \frac{\Delta}{4} & \frac{\Delta}{2} & \frac{\Delta}{2} \end{pmatrix} \begin{pmatrix} p(\mathbf{y}) \\ p_x(\mathbf{y}) \\ p_y(\mathbf{y}) \\ p_z(\mathbf{y}) \end{pmatrix} = \begin{pmatrix} p(\mathbf{x}_1) \\ p(\mathbf{x}_2) \\ p(\mathbf{x}_3) \\ p(\mathbf{x}_4) \\ p(\mathbf{x}_5) \end{pmatrix} \tag{13}$$

where $\Delta \equiv \Delta x = \Delta y = \Delta z$. The least squares solution of (13) gives the stencil for the x-component of the gradient:

$$p_x(\mathbf{y}) \approx \frac{1}{3\Delta}(p_2 + p_3 + p_4 + p_5 - 4p_1). \tag{14}$$

Discrete divergence. The approximation of the velocity divergence in the center $\mathbf{x}_V$ of a fluid grid cell V makes use of the Gauss formula

$$\int_V \nabla \cdot \mathbf{u} \, d\mathbf{x} = \int_{\partial V} \mathbf{u} \cdot \mathbf{n} \, ds \tag{15}$$

where $\mathbf{n}$ is the outward unit normal to the cell boundary. Let $\mathcal{F}(V)$ be the set of faces for V, i.e. $\partial V = \cup_{F \in \mathcal{F}(V)} F$, and $\mathbf{y}_F$ denotes the center of $F \in \mathcal{F}(V)$. Based on (15) we define

$$(\mathrm{div}_h \, \mathbf{u}_h)(\mathbf{x}_V) = |V|^{-1} \sum_{F \in \mathcal{F}(V)} |F| (\mathbf{u}_h \cdot \mathbf{n})(\mathbf{y}_F). \tag{16}$$

Thanks to the staggered location of velocity nodes $(\mathbf{u}_h \cdot \mathbf{n})(\mathbf{y}_F)$ is well-defined.

Discrete viscous terms. Below we define $\Delta_h \mathbf{u}_h$ for a given discrete velocity vector function $\mathbf{u}_h$ in internal velocity nodes of the fluid domain. While the face centered staggered location of velocity nodes is advantageous for stability reasons, it makes the construction of a compact stencil approximation on octree meshes somewhat complicated for Δ_h. Thus, in the previous studies of the FD approximations of the fluid equations on the cartesian octree meshes, see [29, 32, 38, 39], the viscous terms were either ignored (Euler limit) or velocity d.o.f. were collocated in cells' vertices (this simplifies discretization, but requires an additional stabilization to be introduced, see e.g. [35]). To build a compact stencil disctretization for face centered location of velocity d.o.f.,

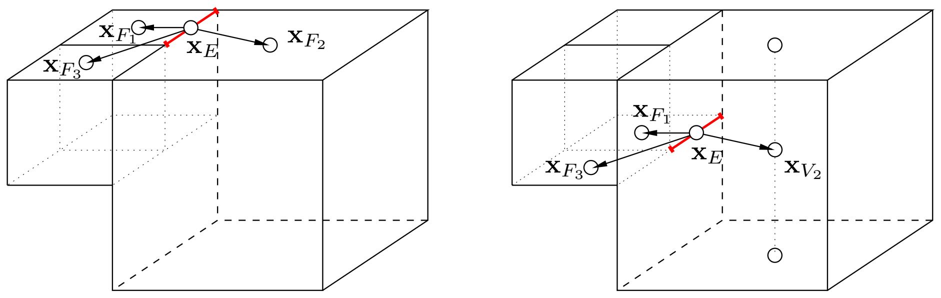


Figure 3: The sets of nodes for edge-fluxes approximation through the system (19).

we handle separately 'normal' and 'tangential' derivatives of Δu for each velocity component. As an example, consider the evaluation of the discrete Laplacian for the x-component u of $\mathbf{u}_h$. We split

$$\Delta_h u_h = \left(\frac{\partial^2 u_h}{\partial x^2}\right)_h + \Delta_{yz,h} u_h, \qquad \text{with} \;\; \Delta_{yz} u = \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}. \tag{17}$$

First, in the center point $\mathbf{x}_V$ of every interior cell V of the fluid domain we assign, similar to (15)–(16):

$$\left(\frac{\partial u_h}{\partial x}\right)_h (\mathbf{x}_V) = |V|^{-1} \sum_{F \in \mathcal{F}(V)} |F| (u_h n_1)(\mathbf{y}_F). \tag{18}$$

where $\mathbf{n} = (n_1, n_2, n_3)$ is the outward normal vector to ∂V in $\mathbf{y}_F$. Now the values of $\left(\frac{\partial^2 u_h}{\partial x^2}\right)_h$ in the u-nodes of interior cells are found in the same way as the x-component of the pressure gradient, i.e. with the central difference formula if the face is shared by two cells of the same size, and (14) (with p replaced by $\left(\frac{\partial u_h}{\partial x}\right)_h$) if F is shared by two cells of different sizes.

The contribution of $\partial^2/\partial y^2 + \partial^2/\partial z^2$ in (17) is evaluated differently. Consider any yz-face F of any interior fluid cell V. Denote by $\mathcal{E}(F)$ the set of all (four) edges of F and let $\mathbf{n}_E$ denote the outward unit normal (in plane yz) to edge $E \in \mathcal{E}(F)$. If the fluxes $U_E \approx \dfrac{\partial u}{\partial \mathbf{n}_E}$ are known, then similarly to (16) we set

$$(\Delta_{yz,h} u_h)(\mathbf{y}_F) = |F|^{-1} \sum_{E \in \mathcal{E}(F)} U_E |E|.$$

It remains to evaluate U_E at the centers $\mathbf{x}_E$ of edges E. There may be three different geometric cases for the edge E: (i) The edge E is shared by two yz-faces of the equal size; (ii) E is shared by two yz-faces F_1 and F_2, the face F_2 is twice as big as F_1 (Fig. 3, left); or (iii) E is shared only by one yz-face F_1 and belongs to a xy-face (or xz-face) face of a larger cell V_2 (Fig. 3, right). In the first case, U_E is evaluated by the central finite differences. In two other cases, the following approach is adopted: we form a set Σ_E of collocation points $\mathbf{x}_i$, $i = 1, \dots, N_{\Sigma_E}$ (for simplicity, we choose $N_{\Sigma_E} = 3$) and write the Taylor expansion

$$u(\mathbf{x}_i) = u(\mathbf{x}_E) + \left(\frac{\partial u(\mathbf{x}_E)}{\partial \mathbf{n}_E}, \frac{\partial u(\mathbf{x}_E)}{\partial \mathbf{t}_E}\right)^T \cdot (\mathbf{x}_i - \mathbf{x}_E) + O(|\mathbf{x}_i - \mathbf{x}_E|^2), \quad i = 1, \dots, N_{\Sigma_E}. \tag{19}$$

Here $\mathbf{t}_E$ denotes the tangential unit vector for edge E. Neglecting the second order term we obtain a system of N_{Σ_E} linear equations with 3 unknowns $u(\mathbf{x}_E)$, $\dfrac{\partial u(\mathbf{x}_E)}{\partial \mathbf{n}_E}$, $\dfrac{\partial u(\mathbf{x}_E)}{\partial \mathbf{t}_E}$. The solution (least squares solution if

$N_{\Sigma_E} > 3$) of this system gives us U_E. In the case (ii), Σ_E is formed from the centers of faces F_1 and F_2 and the center of the yz-face adjunct to both F_1 and F_2 (Fig. 3, left). In the case (iii), Σ_E is formed from the center of F_1, the center of the yz-face adjunct to both F_1 and V_2, and the center of V_2 (Fig. 3, right). The value $u(\mathbf{x}_i)$ in the center of V_2 is computed by linear interpolation.

We note that another approach for discretizing ∇, div, and Δ operators on rather general meshes could use mimetic finite differences, see e.g. [27].

4.3 Velocity interpolation and extension

For the semi-Lagrangian steps of the time-splitting algorithm we need a rule for computing velocity values in any point of the computational domain. We call this procedure the interpolation. A natural choice of interpolation procedure would be, first, to define velocity values in cells vertices by averaging the corresponding velocity nodal values in the neighboring cells faces and, second, compute the velocity value in any point of a cell as the trilinear interpolation of the vertices values. We use this simple (and fast) procedure when the level set function advection step is performed. However, such interpolation was found to produce large numerical diffusion when applied with the semi-Lagrangian method in the fluid part of the splitting algorithm. Therefore, at this step we use another interpolation method described below. The procedure is somewhat more computationally expensive, but involves a smaller interpolation stencil and was found to reduce the numerical diffusion.

Interpolation of $\mathbf{u}_h$ *in arbitrary* $\mathbf{y} \in \Omega$. For a given point $\mathbf{y}$ in computational domain we compute $\mathbf{u}_h(\mathbf{y})$ as follows. Assume $\mathbf{y}$ belongs to a cell V and we are interested in computing the x-component of velocity in $\mathbf{y}$, i.e. $u_h(\mathbf{y})$. Consider a plane $\mathcal{P}$ such that $\mathbf{y} \in \mathcal{P}$ and $\mathcal{P}$ is orthogonal to the Ox axis. Let $\mathbf{x}_V \in \mathcal{P}$ be the orthogonal projection of the center of V on $\mathcal{P}$ and $\mathbf{x}_k$, $k = 1, \ldots, m$, $m \leq 12$, are the projections of centers of all cells sharing a face with V. The values $u_h(\mathbf{x}_V)$ and $u_h(\mathbf{x}_k)$ can be defined by a linear interpolation of the velocity values at u-nodes. Once $u_h(\mathbf{x}_V)$ and $u_h(\mathbf{x}_k)$, $k = 1, \ldots, m$, are computed, we define $u_h(\mathbf{y})$ distinguishing between two cases: First, if $\mathbf{y}$ belongs to a triangle from the triangle fan based on $\mathbf{x}_V$ and $\mathbf{x}_k$, $k = 1, \ldots, m$, then $u_h(\mathbf{y})$ is defined by a linear interpolation between the values of u_h in the vertices of this triangle, cf. Fig. 4; Second, if there is no such triangle (this can be the case for $\mathbf{y} \in \Gamma(t)$, since $\mathbf{u}_h$ needs to be computed on $\Gamma(t)$ *before* the velocity field is extended to the bulk computational domain from $\Omega(t)$), then $u_h(\mathbf{y})$ is defined by the inverse distances method based on the values of u_h in the closest three points from the set $\{\mathbf{x}_V, \mathbf{x}_k\}$, $k = 1, \ldots, m$.

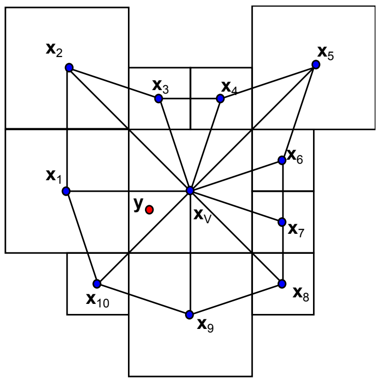


Figure 4: $u_h(\mathbf{y})$ is defined by a linear interpolation based on the fan triangulation with the center in $\mathbf{x}_V$, i.e. interpolation of $u_h(\mathbf{x}_V)$, $u_h(\mathbf{x}_1)$,$u_h(\mathbf{x}_{10})$ in this example.

For updating the level set function we need an extension of velocity field from $\Gamma_h(t)$ (t is fixed) to the entire computational domain. The extension procedure is described below.

Extension of $\mathbf{u}_h$ *from* $\Gamma_h(t)$. We build the normal extension of velocity field from free surface to the nodes of computational domain. To this end, for a given node $\mathbf{x} \in \Omega_h$ we find the "nearest" point $\mathbf{y}_\mathbf{x} \in \Gamma_h(t)$ by the following iterative algorithm. Set $\mathbf{y}^0 = \mathbf{x}$, define $\mathbf{y}^{n+1} = \mathbf{y}^n - \alpha \nabla \phi_h(\mathbf{y}^n)$, $n = 0, 1, \ldots$, with a relaxation parameter $\alpha > 0$. The iteration is terminated once $|\mathbf{y}^{n+1} - \mathbf{y}^n| \leq \varepsilon$ and we set $\mathbf{y}_\mathbf{x} = \mathbf{y}^{n+1}$, $u_h(\mathbf{x}) = u_h(\mathbf{y}_\mathbf{x})$, where $u_h(\mathbf{y}_\mathbf{x})$ is computed via the interpolation. In our calculations we chose $\varepsilon = 10^{-8}$ and $\alpha = \frac{\sqrt{5}-1}{2}$.

5 Free surface handling

In this section, we discuss one of the most important aspects of the algorithm: handling the free surface and computing its geometric quantities. In section 3 we considered the semi-Lagrangian method for (6) to find the evolution of the free surface: new values of the level set function are computed by the numerical integration of (6) backward along the characteristic curves that pass through the cells vertices.

5.1 Volume correction

The numerical advection of the free boundary may cause a noticeable divergence (loss or gain) of the fluid volume. This divergence can be reduced in several ways: (i) The refinement of the computational grid near $\Gamma(t)$. Naturally, the level of mesh refinement is limited by the available computational resources; (ii) More accurate time integration of the level set equation (6). We use the second order accurate method with the integration step $\frac{\Delta t}{n}$ with $n = 2, 3$. Further increase of n was not found to result in more accurate results; (iii) Correction of the level set function by the method of particles [12, 13]. As was shown in [34] the particle level set method alone improves, but does not guarantee the complete volume conservation. Thus we also use (iv) The adjustment of the level set function by adding a suitable constant to preserve the fluid volume. The adjustment of the level set function is performed by solving for a constant δ the following equation

$$\text{meas}\{\mathbf{x} : \phi(\mathbf{x}) < \delta\} = Vol^{\text{reference}}$$

and correcting $\phi^{new} = \phi - \delta$. The bisection algorithm was used to find δ and a Monte-Carlo method was applied to evaluate $\text{meas}\{\mathbf{x} : \phi(\mathbf{x}) < \delta\}$. We note that the volume correction method can be extended to the case of multi-connected fluid domains and more general adjustment functions [33].

5.2 Redistancing

Both the advection and the volume correction of the level set function may cause the loss of its signed distance property. For the continuous level set function this property can be written in the form of the Eikonal equation:

$$|\nabla\phi(\mathbf{x})| = 1, \quad \mathbf{x} \in \mathbb{R}^3, \tag{20}$$

with the boundary condition on the free surface $\Gamma(t)$:

$$\phi(\mathbf{x}) = 0, \quad \mathbf{x} \in \Gamma(t).$$

The property (20) is important for the computation of the geometric quantities of the free boundary and numerical stability. To recover the signed distance property we perform a redistancing procedure, also known as re-initialization.

First, the location of the interface $\Gamma(t)$ is explicitly found. To accomplish this, for each cell intersected by the discrete free surface (such cells are figured out by checking the signs of discrete level set function in cells vertices) a local internal surface triangulation is built using the marching cubes technique [25, 28]. Remarkably, the triangulated global approximation of the interface turns out to be a conformal triangulation in space.

The redistancing procedure is split into two steps: the assignment of distance values in the vertices of interface cells (i.e. cells that are intersected with the interface), and finding solution to a discrete counterpart of (20) in all remaining nodes. The second step is performed by the fast marching method [1] adapted to octree grids. The first step can be performed by several methods described below.

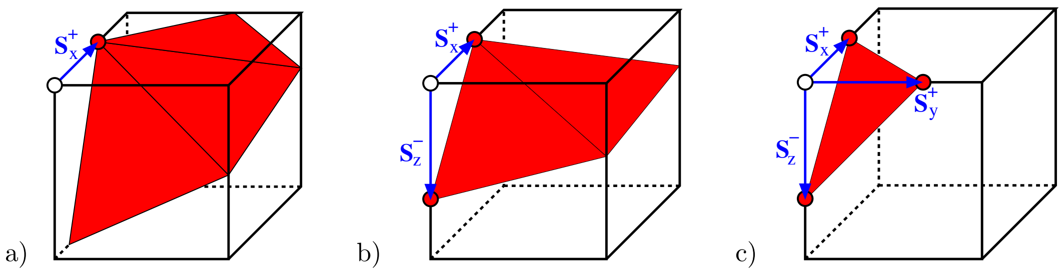


Figure 5: Possible cases for distance reconstruction in an interface cell node.

Method 1 is suggested in [1]. Consider a grid node and an interface cell sharing this node and examine three cell edges sharing the node. If an edge intersects the interface, one measures the distance from the intersection to the node. There are four possible cases induced by the number of intersections.

1. If none of three edges intersects the interface no action is needed.
2. In case of a single intersection (Figure 5a) one computes the distance to that intersection: $d = S_x^+$.
3. In case of two intersections (Figure 5b), one evaluates the distance as the height of the appropriate triangle: $d = \left(\sqrt{\left(\frac{1}{S_x^+}\right)^2 + \left(\frac{1}{S_z^-}\right)^2}\right)^{-1}$.
4. In case of three intersections (Figure 5c), the distance is evaluated as the height of the pyramid built on these points: $d = \left(\sqrt{\left(\frac{1}{S_x^+}\right)^2 + \left(\frac{1}{S_y^+}\right)^2 + \left(\frac{1}{S_z^-}\right)^2}\right)^{-1}$.

For each node $\mathbf{x}$ one computes the values d for all interface cells sharing the node and assigns the minimum value to $\phi(\mathbf{x})$.

In this paper we consider two more methods.

Method 2 computes the distance from each node of interface cells to the interface triangulation explicitly. Given a cell vertex we compute the shortest distance to every triangle of the surface triangulation (of course, for a fixed point not all triangles have to be visited). Algorithmically, it is computed by looking for the smallest distance between the vertex and (i) its projection on the plane containing the triangle, (ii) its projection on the lines containing the edges, (iii) three vertices of the triangle.

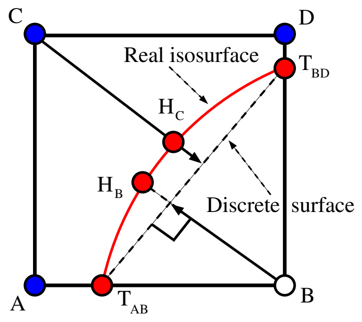


Figure 6: The difference between triangulated isosurface and the true isosurface $\{\phi_h(\mathbf{x}) = 0\}$.

Method 3 employs the same interface triangulation, but the distance to $\Gamma(t)$ is measured differently. We take into account that interface triangulation is only an approximation to the zero level of the piecewise trilinear level set function ϕ_h. For each surface triangle T and a neighboring grid node $\mathbf{x}$ we consider the line passing through $\mathbf{x}$ and orthogonal to the plane which contains T (see Figure 6). The trace of ϕ_h on the line segment contained in the cell is a cubic function $\psi(t) = f_3t^3+f_2t^2+f_1t+f_0$ where $\psi(0) = \phi_h(\mathbf{x})$. The smallest positive root of the equation $\psi(t) = 0$ defines the point $H_{\mathbf{x}}$ where the line crosses the zero isosurface of ϕ_h. If the initial value of $\phi_h(\mathbf{x})$ is greater than the computed distance to $H_{\mathbf{x}}$, we set it equal to this distance. Otherwise we update $\phi_h(\mathbf{x})$ by the distance to $H_{\mathbf{x}}$ if it does not exceed distances to the vertices of the considered triangle.

Remark 5.1 The Method 1 requires only the intersection points of surface $\{\phi_h(\mathbf{x}) = 0\}$ with cell edges, which can be easily found from the nodal values of ϕ_h. The methods 2 and 3 need the preprocessing step of the surface triangulation by the marching cubes algorithm. Although more expensive, methods 2 and 3 produce more accurate and convergent approximations to the distance function (see section 6.2) and their utilization appears to be crucial for modeling phenomena driven by surface tension forces (see the example in section 6.5)

6 Numerical experiments

To validate the accuracy of the numerical method we perform few analytical tests and consider several benchmark problems: the Zalesak's disk problem, the breaking dam problem, and the 3D oscillating droplet problem.

6.1 Zalesak's disk problem

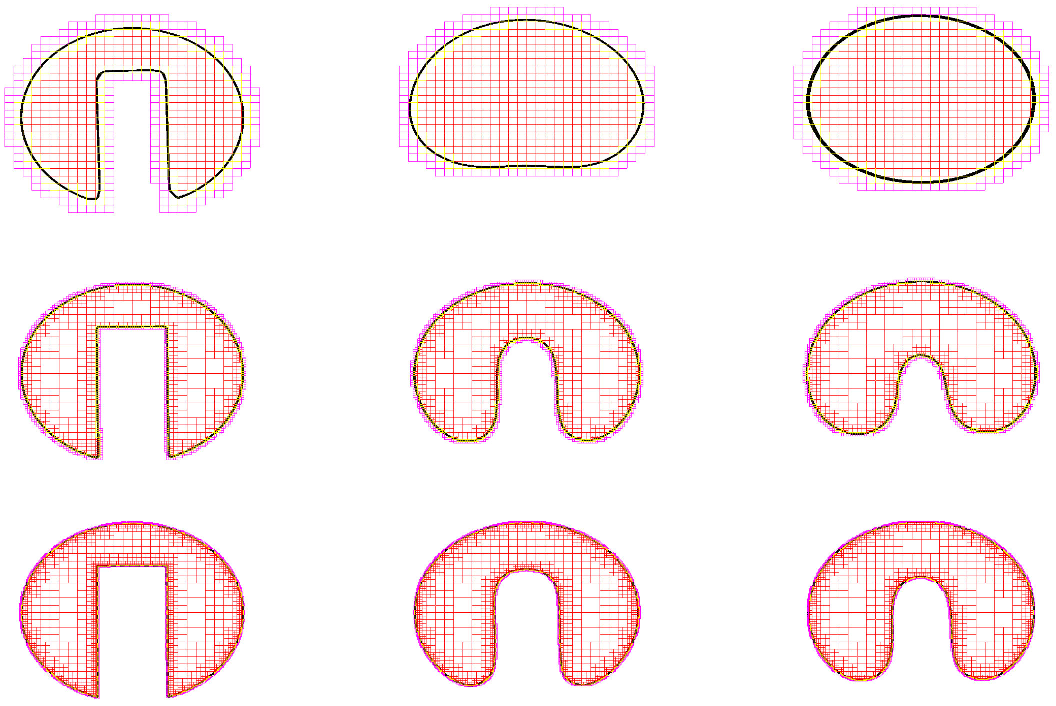


Figure 7: The effect of adaptation on the accuracy of the level set function evolution. Upper row: uniform mesh with $h_{\min} = h_{\max} = \frac{1}{64}$; middle row: octree mesh with $h_{\min} = \frac{1}{256}$, $h_{\max} = \frac{1}{32}$, $h_{\text{ext}} = \frac{1}{16}$; bottom row: octree mesh with $h_{\min} = \frac{1}{512}$, $h_{\max} = \frac{1}{32}$, $h_{\text{ext}} = \frac{1}{16}$.

In the first test we verify the accuracy of the surface capturing method. A slotted disk Z with radius 0.15 and slot width 0.05 is rotating at the center of the unit square. The computational domain is $(0,1)^3$ and Zalesak's disk is represented by the cylinder $Z \times (0,1)$. The corresponding level set function is advected by the prescribed velocity field:

$$u = \frac{\pi}{3.14}(0.5 - y), \quad v = \frac{\pi}{3.14}(x - 0.5), \quad w = 0.$$

Capturing the rotated slotted disk is referred to as the Zalesak's problem [53]. Figure 7 shows the free surface at the initial time, after one and two disk revolutions. The method is clearly too diffusive on the uniform mesh with $h_{\min} = h_{\max} = \frac{1}{64}$. The numerical diffusion reduces significantly on the adapted mesh while the computational complexity remains approximately the same. The number of vertices in the adapted mesh with $h_{\min} = \frac{1}{512}$, $h_{\max} = \frac{1}{32}$, $h_{\text{ext}} = \frac{1}{16}$ is nearly as much as one half of the number of vertices in the uniform mesh with $h = \frac{1}{64}$. The alternative approach to the reduction of the numerical diffusion is to apply higher order approximations of the transport equation on uniform meshes [8]. However, the adaptive mesh approach admits simple and efficient handling of complex geometries emerging due to free surface evolution or imbedded in problem setup.

Another well known accuracy enhancing method for level set function advection is the method of particles introduced in [12, 13]. The idea of the method is to use special massless marker particles near the interface in order to correct the level set function. Each particle has its sign, radius and is assumed to possess the dynamic of a material particle driven by the velocity field of fluid or the extended velocity field in the fluid exterior. The particle sign is the sign of the level set function in a region where the particle was initially introduced. Due to approximation errors some particles can occasionally "escape" from the phase where they have been originally seeded. If a particle is found in a region where ϕ and the particle have opposite signs, one considers a distance from the particle to the surface. If the distance is greater than the radius of the particle, then the level set function is locally adjusted.

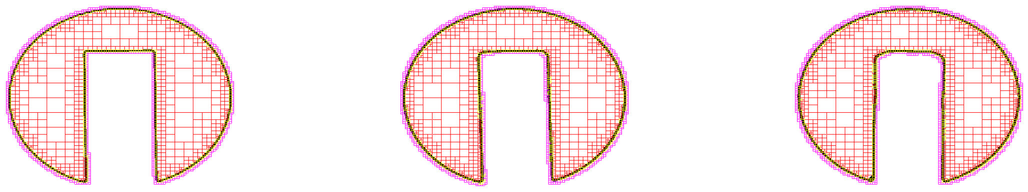


Figure 8: The effect of the particles method on the accuracy of the level set function evolution, adapted mesh with $h_{\min} = \frac{1}{256}$, $h_{\max} = \frac{1}{32}$, $h_{\text{ext}} = \frac{1}{16}$.

The improvement given by the method is shown in Fig. 8 where the initial disk and the disk after one and two revolutions are shown (compare with particle free computations in Fig. 7). The following set of parameters was used in our implementation of the method: particles are seeded uniformly in the interface cells at $t = 0$ with 30 particles per cell; each particle is assigned the radius r, $0.15h_{\min} \le r \le 0.65h_{\min}$. We introduce the 'continuous' reseeding, i.e. after each time step the age of every particle is increased with the probability $\frac{3}{4}$, and a particle is reseeded when it reaches a limit age. We found that such a strategy leads to a smoother level set function evolution than reseeding all particles every N time steps (N is a user defined parameter) as often recommended in the literature. We refer to [12, 13] for further details on the particle level set method. Although the improvement given by the particle level set method is evident for the case of the pure advection problem, care should be taken in the case of coupled level set – fluid systems. The reason is that the computation of mean surface curvatures is sensitive to local perturbations of the level set functions caused by the particles. A particle-free method often provides better results in modeling flows driven by surface tension forces.

6.2 Accuracy of redistancing methods

In the next experiment, we check the accuracy of the discrete Eikonal solvers (redistancing methods) introduced in section 5.2. The surface Γ is the sphere $(x-0.5)^2 + (y-0.5)^2 + (z-0.5)^2 = 0.314^2$ and the computational domain is the unit cube. The solvers accuracy is studied by comparing the discrete solutions with the analytical solution to (20). The error is measured only for the interface cells T such that $T \cap \Gamma \neq \emptyset$. Let $\mathcal{V}(V)$ denote the set of all vertices of cell V. We use the following indicators to evaluate discretization errors for discrete solutions:

$$\varepsilon_i^C = \frac{\max\limits_V \max\limits_{\mathbf{x}\in\mathcal{V}(V)} |\phi_h^i(\mathbf{x}) - \phi(\mathbf{x})|}{\max\limits_V \max\limits_{\mathbf{x}\in\mathcal{V}(V)} |\phi(\mathbf{x})|}, \qquad \varepsilon_i^L = \left(\frac{\sum\limits_V |V| \sum\limits_{\mathbf{x}\in\mathcal{V}(V)} |\phi_h^i(\mathbf{x}) - \phi(\mathbf{x})|^2}{\sum\limits_V |V| \sum\limits_{\mathbf{x}\in\mathcal{V}(V)} \phi^2(\mathbf{x})} \right)^{1/2},$$

where ϕ_h^i is the discrete signed distance function found by the ith redistancing method. The indicators resemble the scaled C and L^2 norms of the error, respectively. The scaling of the norms accounts for vanishing of ϕ in interface cell nodes when the grid is refined: $\max\limits_{\mathbf{x}\in\mathcal{V}(V)} |\phi(\mathbf{x})| = O(h_{\min})$.

$h_{\min}$	ε_1^C	ε_1^L	ε_2^C	ε_2^L	ε_3^C	ε_3^L
32^{-1}	1.4e-1	1.0e-1	5.1e-2	3.5e-2	5.0e-2	3.5e-2
64^{-1}	1.9e-1	1.1e-1	2.7e-2	1.8e-2	2.5e-2	1.8e-2
128^{-1}	1.9e-1	9.7e-2	1.5e-2	9.9e-3	1.3e-2	8.8e-3
256^{-1}	2.2e-1	9.8e-2	8.6e-3	6.7e-3	6.4e-3	4.4e-3
512^{-1}	2.2e-1	9.7e-2	5.5e-3	5.6e-3	3.3e-3	2.4e-3

Table 1: Accuracy of three redistancing methods.

Table 1 shows the errors on the sequence of refined grids. We note that in both norms method 1 exhibits no convergence, method 2 shows less than the first order convergence, and method 3 shows the first order convergence.

6.3 Approximation of surface normal vector and curvature

Now we study the accuracy of surface normal vector and curvature approximations. We recall that the unit outward normal can be computed via the level set function: $\mathbf{n}_\Gamma = \nabla\phi/|\nabla\phi|$ on $\Gamma(t)$. We derive the second order approximation of the gradient through the Taylor expansion in all possible combinations of octree cells sharing the node.

The mean curvature of the interface can be defined as the divergence of the normal vector, $\kappa(\phi) = \nabla \cdot \mathbf{n} = \nabla \cdot (\nabla\phi/|\nabla\phi|)$. Thus the approximation of the mean curvature is computed as follows. First, $\nabla_h\phi_h$ is computed in cell vertices and is averaged in face centers. Once $\nabla_h\phi_h/|\nabla_h\phi_h|$ is known in face centers, $\kappa_h(\phi_h) = \nabla_h \cdot \nabla_h\phi_h/|\nabla_h\phi_h|$ is computed in cell centers by standard second order center differences. Since $\nabla\phi$ is computed with second order accuracy, $\kappa(\phi)$ is approximated with the first order.

To test the approximations of $\mathbf{n}$ and κ, we take the same spherical interface and introduce the following indicators of the level set gradient and mean curvature approximation accuracy. Denote by $\mathcal{V}(V)$ the set of all vertices for every cell V such that V has nonempty intersection with $\Gamma(t)$ or a neighboring cell of V has nonempty intersection with $\Gamma(t)$. The set of all center points of the same set of cells is denoted by C. Define

$$\varepsilon^C_\nabla = \max_V \max_{\mathbf{x}\in\mathcal{V}(V)} |\nabla_h\phi(\mathbf{x}) - \nabla\phi(\mathbf{x})|, \qquad \varepsilon^L_\nabla = \left(\frac{\sum\limits_V |V| \sum\limits_{\mathbf{x}\in\mathcal{V}(V)} |\nabla_h\phi(\mathbf{x}) - \nabla\phi(\mathbf{x})|^2}{\sum\limits_V |V| \sum\limits_{\mathbf{x}\in\mathcal{V}(V)} |\nabla\phi(\mathbf{x})|^2} \right)^{1/2},$$

$$\varepsilon^C_\kappa = \max_{\mathbf{x}\in C} |\kappa_h(\phi)(\mathbf{x}) - \nabla \cdot (\nabla\phi(\mathbf{x})/|\nabla\phi(\mathbf{x})|)|, \qquad \varepsilon^L_\kappa = \left(\frac{\sum\limits_V |V| \sum\limits_{\mathbf{x}\in C} \left[\kappa_h(\phi)(\mathbf{x}) - \nabla \cdot \frac{\nabla\phi(\mathbf{x})}{|\nabla\phi(\mathbf{x})|}\right]^2}{\sum\limits_V |V| \sum\limits_{\mathbf{x}\in C} \nabla \cdot (\nabla\phi(\mathbf{x})/|\nabla\phi(\mathbf{x})|)^2} \right)^{1/2},$$

where $\kappa_h(\phi)$ is the discrete curvature operator described above.

$h_{\min}$	ε^C_∇	ε^L_∇	ε^C_κ	ε^L_κ
32^{-1}	6.8e-3	2.2e-3	1.8e-2	1.0e-3
64^{-1}	1.4e-3	5.9e-4	8.8e-3	3.6e-4
128^{-1}	3.2e-4	1.5e-4	6.6e-3	1.4e-4
256^{-1}	7.9e-5	3.9e-5	3.3e-3	6.2e-4

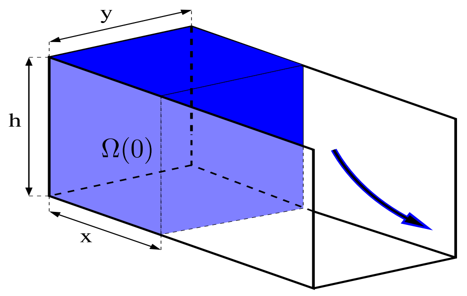
Table 2: Approximation of the gradient operator and the curvature.

Table 2 presents the approximation errors for surface curvature and gradient of ϕ. The gradient approximation shows the second order accuracy in both C- and scaled L^2-norms. As expected, the curvature is approximated only with the first order in both norms.

6.4 Breaking dam problem

This is a classic test case for free surface flows. It was adopted by several researchers as a benchmark test to validate the numerical performance of the proposed formulations in solving two-liquid interfaces or free-surface flows both in 2D and 3D, see e.g. [9, 16, 22, 23, 43, 52]. The computational problem setup is shown in figure 9 (left). In the initial state, the fluid is placed on the left-hand side as a water column with dimensions $x = y = h = 1$. Further, the water column collapses driven by the gravity force $\mathbf{f} = (0, 0, 1)^T$. On the walls we impose slip boundary conditions: $\mathbf{u} \cdot \mathbf{n} = 0$ and $\mathbf{t}_k \cdot \boldsymbol{\sigma}\mathbf{n} = 0$, where $\boldsymbol{\sigma}$ denotes the stress tensor, $\mathbf{t}_k$, $k = 1, 2$ are tangent vectors. Two statistics are of interest: the evolution of the free surface along the bottom wall and along the back wall.

In the numerical simulations the values of ν, ρ, τ were set to model the viscosity, density, and surface tension of water. The numerical results obtained for the horizontal location of the free surface front along the bottom wall



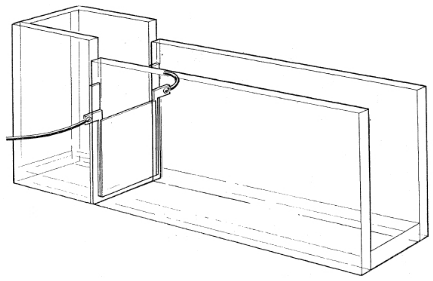
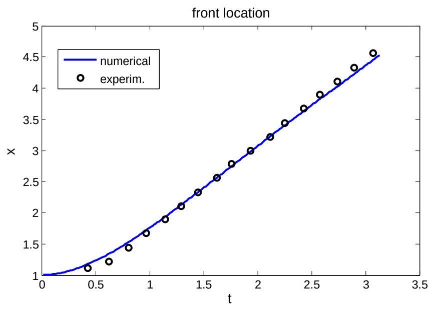


Figure 9: Left: The setup of the 'breaking dam problem' in numerical simulations; Right: The schematic apparatus from [30] for the experimental study of the collapse of liquid columns



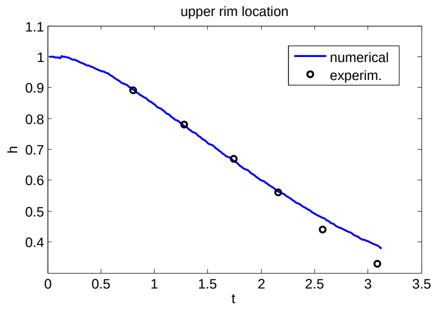


Figure 10: The computed position of the free surface bottom front and upper rim for the breaking dam problem versus experimental values from [30].

and the vertical location of the upper free surface rim along the vertical wall are compared to the experimental values from [30] in figure 10, the experiment setup from [30] is illustrated in figure 9 (right). Both computed statistics are in good agreement with the experimental values, though a minor deviation of the upper rim position is observed near the final times.

6.5 Oscillating droplet problem

We consider a droplet for which evolution is driven only by surface tension forces. The fluid is assumed to be in rest at time $t = 0$ and $\mathbf{f} = \mathbf{0}$. The initial shape of the droplet is a perturbation of a sphere. In spherical coordinates (r, θ, φ) the initial shape is given by

$$r = r_0(1 + \varepsilon S_2(\frac{\pi}{2} - \theta)),$$

where S_2 is the second spherical harmonic. In all experiments we set $r_0 = 1$, $\tau = 1$, $\varepsilon = 0.3$. At $t = 0$ the mean curvature of the surface is not constant, and an unbalanced surface tension force causes droplet oscillation. In this experiment the fluid motion is solely driven by the surface tension forces. Therefore, the quality of the numerical solution is sensitive to the accuracy of the level set function and surface curvature approximations. This is the challenging test for the surface tension approximation and free surface capturing method where an accurate and smooth curvature field is critical. The oscillating droplet problem often serves as a benchmark test for free surface and two-phase fluid flow solvers, see e.g. [3, 10, 15, 17, 18, 40]. Two statistics are of common interest: The droplet oscillation period T and the damping factor δ. Assuming the perturbation is small ($\varepsilon \ll 1$), a linear stability analysis from [24] predicts the period and the damping factor according to

$$T_{\text{ref}} = 2\pi\sqrt{\frac{\rho r_0^3}{8\tau}}, \qquad \delta_{\text{ref}} = \frac{r_0^2}{5\nu}. \tag{21}$$

We solved the problem on a sequence of meshes with $h_{\min} \in \{\frac{1}{32}, \frac{1}{64}, \frac{1}{128}, \frac{1}{256}\}$ and the constant coarse mesh

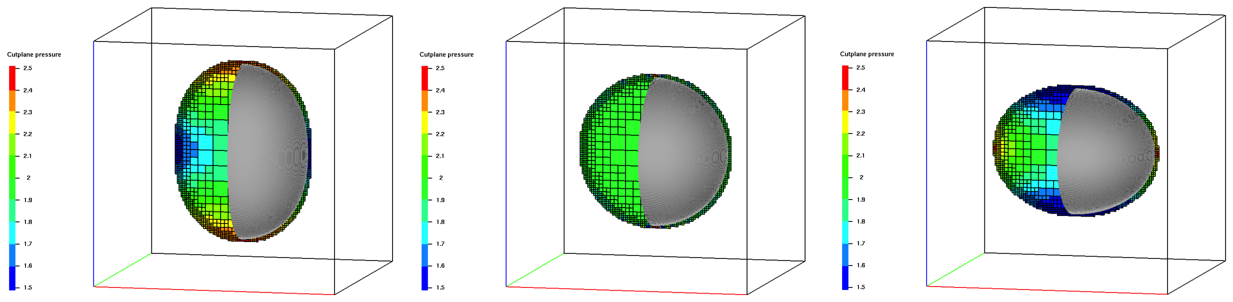


Figure 11: The droplet shape, grid and pressure distribution for $t = \{0, 0.726, 1.286\}$.

size $h_{\max} = \frac{1}{16}$ (mesh size in the fluid domain interior) and $h_{\text{ext}} = \frac{1}{16}$ (mesh size in the fluid domain exterior). We also repeated experiments with gradually refined $h_{\max} = h_{\min}/2$ and observed similar results, although the timings increased significantly. The third method (see § 5.2) of discrete level set function re-initialization was used. While the second method gave qualitatively similar results, the first one resulted in non-physical oscillatory solutions.

We experiment with $\nu = 0$ which allows both to look at the quality of surface tension forces approximation and to quantify the effect of numerical diffusion of the scheme. Figure 11 illustrates the droplet shape, the

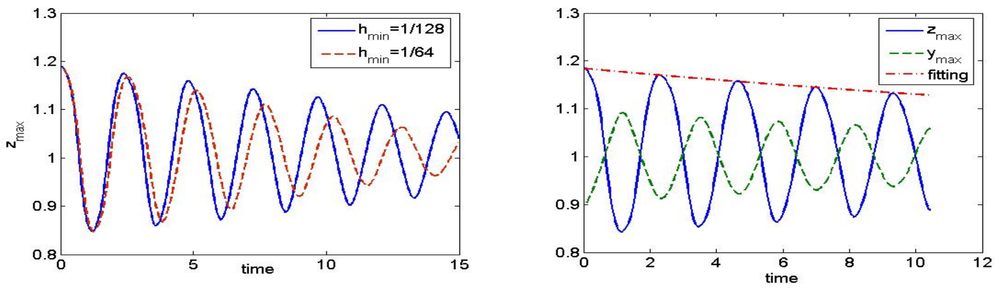


Figure 12: The left picture compares the top tip trajectory on the z axes for $h_{\min} = \frac{1}{64}$ and $h_{\min} = \frac{1}{128}$. The right picture shows the droplet tips trajectories for $h_{\min} = \frac{1}{256}, h_{\max} = \frac{1}{16}$ and the fitted curve.

grid ($h_{\min} = \frac{1}{128}$, $h_{\max} = \frac{1}{16}$) and the pressure distribution at times equal 0, 0.726, and 1.286. Further, the trajectories of the droplet tips are shown in Figure 12. One can see that the decay of the oscillation amplitude becomes lesser if the mesh is refined towards the surface (left picture), though remains visible for the finer mesh we used (right picture). The exponent curve fitted to the local maximums of the droplet top tips (right picture) quantifies the decay factor as explained below. With $h_{\min} = \frac{1}{32}$ the method failed to recover the correct physical behavior of solution for times more than one period of the droplet oscillation.

The computed period and the decay factor are given in Table 3. We observe that adaptive grid refinement leads to the convergence of the computed period to the reference one, while the decay factor grows slowly. This suggests that the scheme remains somewhat dissipative. We note that in general the particle method is believed to reduce the numerical dissipation, however for the present benchmark test the particle correction of level set function was found to produce large enough disturbance of ϕ_h leading to inaccurate surface tension forces modeling. To estimate the amount of the numerical dissipation in the scheme we compute the damping factor δ by the least square fitting of the function $c\exp(-\frac{t}{\delta})$ to the computed maximum values of the droplet top tip: $z_{\max}(t_n) - r_\infty$, where t_n are the times when $z_{\max}(t)$ attains a local maximum, r_∞ is the radius of a spherical droplet with the same volume as the initial droplet. We evaluate the effective numerical viscosity of the scheme ν_{num} assuming that similar to (21) it holds $\delta \approx r_0^2(5\nu_{\text{num}})^{-1}$.

$h_{\min}$	T	$T-T_{\text{ref}}$	δ	ν_{num}	# cells	$Time_{\Delta t}$
32^{-1}	2.80	0.579	–	–	3,248	0.21
64^{-1}	2.475	0.254	12.75	0.0157	14,544	0.89
128^{-1}	2.378	0.157	21.63	0.0092	61,976	4.28
256^{-1}	2.275	0.054	28.59	0.0070	258,552	18.4

Table 3: The period, the decay factor of the computed solution, estimated effective numerical viscosity of the scheme; also CPU times (sec.) per one time step versus the total number of cells in $\Omega_h(0)$.

Further, consider the energy balance for the problem:

$$\frac{1}{2}\int_{\Omega(t)}|\mathbf{u}(t)|^2\mathrm{d}\mathbf{x}+\nu\int_0^t\int_{\Omega(t)}\mathbf{Du}:\mathbf{Du}\,\mathrm{d}\mathbf{x}\,\mathrm{d}t'+\tau|\Gamma(t)|=\frac{1}{2}\int_{\Omega(t)}|\mathbf{u}(0)|^2\mathrm{d}\mathbf{x}+\tau|\Gamma(0)|\quad\forall\,t\geq 0, \tag{22}$$

where $\mathbf{Du}=[\nabla\mathbf{u}+(\nabla\mathbf{u})^T]/2$, $|\Gamma(t)|$ denotes the measure of the surface $\Gamma(t)$. For $\nu=0$ and $\mathbf{u}(0)\equiv 0$ the kinetic energy peaks and amplitude should not decrease in time. If they do decrease (the decay of kinetic energy is even more visible, since the kinetic energy is *quadratic* invariant, see Fig. 13), then the energy balance (22) is not preserved by the numerical scheme. To the best of our knowledge, building (energy) conservative adaptive FD

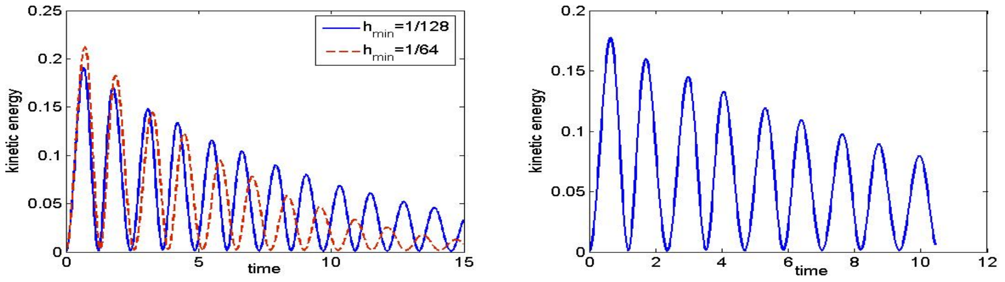


Figure 13: The left picture compares the kinetic energy evolution for $h_{\min}=\frac{1}{64}$ and $h_{\min}=\frac{1}{128}$. The right picture shows the kinetic energy evolution for $h_{\min}=\frac{1}{256}, h_{\max}=\frac{1}{16}$.

schemes for the Navier-Stokes coupled with level set equations model is a largely open problem which deserves further studies.

Finally, to check the scalability of the method we show in the right column of Table 3 the CPU times obtained on Altix XE310 node with 2x 2.66 GHz Intel Quad-Core Xeon X5355 processors with 8Gb memory. The average timings are shown for one time step versus the total number of 'active' cells at $t=0$. The data demonstrates almost linear dependence of the CPU time on the number of grid cells.

7 Conclusions

We considered a computational approach for free surface flow simulation based on the Navier-Stokes equations coupled with level set function equation discretized on dynamically refined/coarsened octree cartesian grids. Other key components of the approach are splitting method for time discretization, redistancing algorithm for the discrete level set function and accurate approximation of free surface normal vector and mean curvature. The method is computationally efficient and capable for handling complex surface topologies and surface geometries. Several redistancing and normal vector / mean curvature approximation techniques were comparatively studied. The redistancing method based on marching cubes algorithm for free surface reconstruction was shown to provide accurate approximation to the distance function and hence leads to efficient surface tension forces treatment. We performed several numerical tests with analytical solutions and benchmark problems which illustrate the performance of the method. The reference [54] can be used to download the animated numerical solutions for the Zalesak's disk, the breaking dam, and the oscillating droplet problems from the paper as well as few other animations of computed free surface fluid flows, which illustrate the flexibility of the approach studied in the paper.

The authors thank A.Suleimanov for his help in implementing the Fast Marching Method.

References

[1] D. Adalsteinsson, J.A. Sethian, The fast construction of extension velocities in level set methods, J. Comput. Phys. 148 (1999) 2–22.

[2] R.E. Bank, T.F. Dupont, H. Yserentant, The hierarchical basis multigrid method, Numer. Mathem. 52 (1988) 427–458.

[3] E. Bänsch, Finite element discretization of the Navier-Stokes equations with a free capillary surface, Numer. Math. 88 (2001) 203–235.

[4] M. Behr, Stabilized space-time finite element formulations for free surface flows, Comm. Numer. Meth. Engrg. 11 (2001) 813–819.

[5] E. Bertakis, S. Gross, J. Grande, O. Fortmeier, A. Reusken, A. Pfennig, Validated simulation of droplet sedimentation with finite-element and level-set methods, Chem. Eng. Science 65 (2010) 2037–2051.

[6] J. Bramble, J. Pasciak, J. Xu, Parallel multilevel preconditioners. Math.Comp. 55 (1990) 1–22.

[7] A. Chorin, Numerical solution of the Navier-Stokes equations. Math. Comp., 22 (1968) 745–762.

[8] R. Croce, M. Griebel, M. A. Schweitzer, Numerical simulation of bubble and droplet deformation by a level set approach with surface tension in three dimensions, Int. J. Numer. Meth. Fluids 62 (2010) 963–993.

[9] M. A. Cruchaga, D. J. Celentano, T. E. Tezduyar, Collapse of a liquid column: numerical simulation and experimental validation, Comput. Mech. (2007) 39 453–476.

[10] M. Dai, H. Wang, J. B. Perot, D. P. Schmidt, Direct Interface Tracking of Droplet Deformation, Atomization and Sprays, 12 (2002) 721–735.

[11] V.G. Dyadechko, Y.I. Iliash, Y.V. Vassilevski, Structuring preconditioners for unstructured meshes, Russ.J.Numer.Anal.Math.Modelling 11 (1996) 139–154.

[12] D. Enright, R. Fedkiw, J. Ferziger, I. Mitchell, A hybrid particle level set method for improved interface capturing. J. Comput. Phys. 183 (2002) 83–116.

[13] D. Enright, F. Losasso, R. Fedkiw, A fast and accurate semi-Lagrangian particle level set method, Comput. Struct. 83 (2005) 479–490.

[14] P. Esser, J. Grande, A. Reusken, An extended finite element method applied to levitated droplet problems, Int. J. for Numer. Meth. in Engineering (2010). DOI: 10.1002/nme.2913

[15] M.M. Francois, S.J. Cummins, E.D. Dendy, D.B. Kothe, J.M. Sicilian, M.W. Williams, A balanced-force algorithm for continuous and sharp interfacial surface tension models within a volume tracking framework, J. Comput. Phys. 213 (2006) 141–173.

[16] T. P. Fries, The intrinsic XFEM for two-fluid flows, Int. J. Numer. Meth. Fluids 60 (2009) 437–471.

[17] S. Ganesan, L. Tobiska, An accurate finite element scheme with moving meshes for computing 3D-axisymmetric interface flows, Int. J. Numer. Meth. Fluids 57 (2008) 119–138.

[18] S. Ganesan, L. Tobiska, A coupled arbitrary Lagrangian-Eulirian and Lagrangian method for computation of free surface flows with insoluble surfactants, J. Comput. Phys. 228 (2009) 2859–2843.

[19] I. Ginzburg, G. Wittum, Two-Phase Flows on Interface Refined Grids Modeled with VOF, Staggered Finite Volumes, and Spline Interpolants, J. Comput. Phys. 166 (2001) 302–335.

[20] S.Gross, V.Reichelt, A.Reusken, A finite element based level set method for two-phase incompressible flows, Computing and Visualization in Science, 9 (2006) 239–257.

[21] F. Harlow, J. Welch. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. Phys. Fluids, 8 (1965) 2182-2189.

[22] M.S. Kim, W.I. Lee, A new VOF-based numerical scheme for the simulation of fluid flow with free surface. Part I: New free surface- tracking algorithm and its verification, Int. J. Numer. Methods Fluids 42 (2003) 765–790.

[23] Kohno H., Tanahashi T., Numerical analysis of moving interfaces using a level set method coupled with adaptive mesh refinement, Int. J. Numer. Methods Fluids 45 (2004) 921–944.

[24] H. Lamb, Hydrodynamics, Cambridge University Press, 1932.

[25] J.-O. Lachaud, Topologically defined iso-surfaces. DGCI (1996) 245–256.

[26] V. Lebedev, Difference analogues of orthogonal decompositions, basic differential operators and some boundary problems of mathematical physics, I,II. U.S.S.R. Comput. Math. Math. Phys., 4(3) (1964) 69–92, 4(4) (1964) 36–50.

[27] K. Lipnikov, M. Shashkov, D. Svyatskiy, The mimetic finite difference discretization of diffusion problem on unstructured polyhedral meshes, J. Comput. Phys. 211 (2006) 473–491.

[28] W. Lorensen, H. Cline, Marching Cubes: A High Resolution 3D Surface Construction Algorithm, Computer Graphics, 21 (1987) 163–169.

[29] F. Losasso, F. Gibou, R. Fedkiw, Simulating water and smoke with an octree data structure, ACM Transactions on Graphics (TOG), 23 n.3, August 2004.

[30] J. Martin, W. Moyce, An experimental study of the collapse of liquid columns on a rigid horizontal plane, Philos.Trans.R.Soc.Lond.Ser.A 244 (1952), 312–324.

[31] D. Meagher, Geometric modeling using octree encoding, Computer Graphics and Image Processing, 19 (1982) 129–147.

[32] C. Min, F. Gibou, A second order accurate level set method on non-graded adaptive cartesian grids, J. Comput. Phys. 225 (2007) 300–321.

[33] F. Mut, G.C. Buscaglia, E.A. Dari, A new mass-conserving algorithm for level set redistancing on unstructured meshes, Mecanica Computacional 23 (2004) 1659–1678.

[34] K.D. Nikitin, Y. V. Vassilevski, Free surface flow modelling on dynamically refined hexahedral meshes, Rus. J. Numer. Anal. Math. Model., 23 (2008) 469–485.

[35] Olshanskii M.A., Analysis of semi-staggered finite-difference method with application to Bingham flows, Comp. Meth. Appl. Mech. Eng. 198 (2009) 975–985.

[36] S. Osher, R. Fedkiw, Level Set Methods and Dynamic Implicit Surfaces. Springer-Verlag, 2002.

[37] J. E. Jr. Pilliod, E. G. Puckett. Second-order accurate volume-of-fluid algorithms for tracking material interfaces, J. Comput. Phys. 199 (2004) 465–502.

[38] S. Popinet, Gerris: a tree-based adaptive solver for the incompressible Euler equations in complex geometries, J. Comput. Phys. 190 (2003) 572–600.

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

[40] S. Quan, D. Schmidt, A moving mesh interface tracking method for 3D incompressible two-phase flows, J J. Comput. Phys. 221 (2007) 761–780.

[41] H. Samet, The Design and Analysis of Spatial Data Structures, Addison-Wesley, New York, 1989.

[42] H. Samet, Applications of Spatial Data Structures: Computer Graphics, Image Processing and GIS, Addison-Wesley, New York, 1990.

[43] Sochnikov V, Efrima S., Level set calculations of the evolution of boundaries on a dynamically adaptive grid, Int. J. Numer. Methods Eng. 56 (2003) 1913–1929

[44] R. Scardovelli, S. Zaleski, Direct numerical simulation of free-surface and interfacial flow, Annual Review of Fluid Mechanics 31 (1999) 567–603.

[45] J.A. Sethian, Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science, Cambridge University Press, Cambridge, 1999.

[46] J. Strain, Tree Methods for Moving Interfaces, J. Comput. Phys. 151 (1999) 616–648.

[47] M. Sussman, P. Smereka, S. Osher, A level set approach for computing solutions to incompressible two-phase flow, J. Comput. Phys. 114 (1994) 146–159.

[48] J.Strain, Semi-Lagrangian methods for level set equations. J. Comput. Phys., 151 (1999) 498–533.

[49] R. Szeliski, Rapid octree construction from image sequences, CVGIP: Image Understanding, 58 (1993) 23–32.

[50] G. Tryggvason, B. Bunner, A. Esmaeeli, D. Juric, N. Al-Rawahi, W. Tauber, J. Han, S. Nas, and Y.-J. Jan. A front-tracking method for the computations of multiphase flow. J. Comput. Phys., 169 (2001) 708–759.

[51] S. O. Unverdi and G. Tryggvason. A front-tracking method for viscous, incompressible multi-fluid flows. J. Comput. Phys., 100 (1992) 25–37.

[52] Yue W., Lin C.L., Patel V.C., Numerical simulation of unsteady multidimensional free surface motons by level set method. Int J Numer Methods Fluids 42 (2003) 853–884.

[53] S.T. Zalesak, Fully multidimensional flux-corrected transport algorithms for fluids, J. Comput. Phys. 31 (1979) 335–362.

[54] http://www.inm.ras.ru/research/freesurface/jcp