Research Article

Kirill D. Nikitin, Maxim A. Olshanskii, Kirill M. Terekhov and Yuri V. Vassilevski

A Splitting Method for Numerical Simulation of Free Surface Flows of Incompressible Fluids with Surface Tension

Abstract: The paper studies a splitting method for the numerical time-integration of the system of partial differential equations describing the motion of viscous incompressible fluid with free boundary subject to surface tension forces. The method splits one time step into a semi-Lagrangian treatment of the surface advection and fluid inertia, an implicit update of viscous terms and the projection of velocity into the subspace of divergence-free functions. We derive several conservation properties of the method and a suitable energy estimate for numerical solutions. Under certain assumptions on the smoothness of the free surface and its evolution, this leads to a stability result for the numerical method. Efficient computations of free surface flows of incompressible viscous fluids need several other ingredients, such as dynamically adapted meshes, surface reconstruction and level set function re-initialization. These enabling techniques are discussed in the paper as well. The properties of the method are illustrated with a few numerical examples. These examples include analytical tests and the oscillating droplet benchmark problem.

Keywords: Free surface flow, surface tension, projection method, stability estimate, conservation properties

MSC 2010: 65M12, 65M06, 76D45, 76M20

Kirill D. Nikitin: Institute of Numerical Mathematics, Russian Academy of Sciences, Gubkin str. 8, 119333 Moscow, Russia, e-mail: nikitin.kira@gmail.com
Maxim A. Olshanskii: Department of Mathematics, University of Houston, 641 PGH Building, Houston, TX 77204-3008, USA, e-mail: molshan@math.uh.edu
Kirill M. Terekhov: Department of Energy Resources Engineering, Stanford University, 367 Panama Street, Stanford, CA 94305-2220, USA; and Institute of Numerical Mathematics, Russian Academy of Sciences, Moscow, Russia, e-mail: kirill.terehov@gmail.com
Yuri V. Vassilevski: Institute of Numerical Mathematics, Russian Academy of Sciences, Gubkin str. 8, 119333 Moscow; and Moscow Institute of Physics and Technology, Institutskii Lane 9, 141700, Dolgoprudny, Russia, e-mail: yuri.vassilevski@gmail.com

1 Introduction

Flows of fluids with free surfaces are ubiquitous in nature and engineering. Examples 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., [14, 36] and references therein. A variety of numerical approaches is known from the literature that are based on explicit surface tracking [40] or implicit surface capturing techniques [30, 39], as well as finite difference [16], finite volume [12], and finite element [3, 4] methods. It has also been admitted by a number of authors, see, e.g., [24, 32, 39], that employing a splitting strategy for numerical time integration of the system of fluid and surface evolution equations is a natural way for building an efficient method.

The mathematical model studied in this paper is based on the Navier–Stokes equations coupled with the level set function equation. Our numerical approach combines a time splitting algorithm with dynamic mesh adaptation. Despite significant progress in developing computational schemes for free-surface flow problems, not much of numerical analysis has been done. One possible reason is that even a partial analysis requires a scheme to reproduce a suitable balance of energy, momentum and angular momentum which is satisfied

by the solution of the flow problem. We discuss these properties of the differential solution and originating difficulties for numerical analysis in Section 3. The first objective of this paper is to present the first stability analysis of a popular splitting method for numerical time integration of the system of equations which models a free surface flow of viscous incompressible fluid subject to surface tension forces. Further, a spacial finite difference discretization is introduced and a performance of the fully discrete algorithm is studied in a series of numerical experiments.

The splitting scheme that we study 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. We note that discretizations on octree (quadtree) cartesian grids already enjoyed an employment in free surface flow computations [24, 27, 28, 32, 38], yet many aspects of using octree grids for accurate and predictive computations of free surface flows require more thorough study. To discretize divergence, gradient and Laplace operators, we use finite difference approximations with compact stencils. Particular attention is paid to the following important computational 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 and compare several re-initialization methods based on the marching cubes algorithm for free surface triangulation. 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 presents the mathematical model. Section 3 contains some preliminaries and reviews certain conservation properties of the model. In Section 4 we introduce the splitting algorithm and analyze its stability and conservation properties. A finite difference method for space discretization and further important techniques such as re-initialization methods for the level set function are considered in Section 5. Numerical results for several test problems including the freely oscillating droplet benchmark test are presented in Section 6. Section 7 contains some closing remarks.

2 Mathematical Model

Consider a Newtonian incompressible fluid flow in a bounded time-dependent domain $\Omega(t) \in \mathbb{R}^3$ for $t \in (0, T]$. The fluid dynamics is governed by the incompressible Navier–Stokes equations

$$\begin{cases} \rho\Big(\dfrac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u}\Big) - \nu\Delta\mathbf{u} + \nabla p = \mathbf{g} \\ \qquad\qquad\qquad\qquad \operatorname{div}\mathbf{u} = 0 \end{cases} \quad \text{in } \Omega(t),\ t \in (0, T], \tag{2.1}$$

where $\mathbf{u}$ is the velocity vector field, p is the kinematic pressure, $\mathbf{g}$ 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:

$$\Omega(0) = \Omega_0, \quad \mathbf{u}|_{t=0} = \mathbf{u}_0, \quad \operatorname{div}\mathbf{u}_0 = 0. \tag{2.2}$$

We assume that the entire boundary of Ω is a free surface $\Gamma(t)$ which passively evolves with the normal velocity of fluid, i.e., the following kinematic condition is valid:

$$v_\Gamma = \mathbf{u}\cdot\mathbf{n} \quad \text{on } \Gamma(t), \tag{2.3}$$

where $\mathbf{n}$ is the outward normal vector for $\Gamma(t)$ and v_Γ is the normal velocity of $\Gamma(t)$. Boundary conditions on $\Gamma(t)$ result from balancing the surface tension and the fluid stress forces:

$$\boldsymbol{\sigma}(\mathbf{u}, p)\mathbf{n}|_\Gamma = \tau\kappa\mathbf{n} - p_{\text{ext}}\mathbf{n} \quad \text{on } \Gamma(t), \tag{2.4}$$

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

For computational purposes, we shall employ the implicit definition of the free surface evolution with the help of an indicator function. Let $\Gamma(t)$ be given as the zero level of a globally defined function $\phi(t, \mathbf{x})$. A smooth (at least Lipschitz continuous) function ϕ such that

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

is called the level set function. The initial condition (2.2) allows us to define $\phi(0, \mathbf{x})$. The kinematic condition (2.3) implies that for $t > 0$ the level set function can be found as the solution to the transport equation [29]:

$$\frac{\partial \phi}{\partial t} + \tilde{\mathbf{u}} \cdot \nabla \phi = 0 \quad \text{in } \mathbb{R}^3 \times (0,T] \tag{2.5}$$

where $\tilde{\mathbf{u}}$ is any (divergence-free) smooth velocity field such that $\tilde{\mathbf{u}} = \mathbf{u}$ on $\Gamma(t)$.

A numerical method studied in this paper solves the system of equations (2.1), (2.2), (2.4), and (2.5). 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 $\mathbf{n} = \nabla\phi/|\nabla\phi|$, and the mean surface curvature is $\kappa = \operatorname{div} \mathbf{n}$. 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

$$|\nabla \phi| = 1.$$

3 Conservation Laws

A smooth solution to the free surface flow problem (2.1)–(2.4) satisfies several conservation properties and an energy balance. Before we analyze the numerical method, it is instructive to see how these properties follow from equations (2.1)–(2.4). To this end, recall some elementary integral relations. We shall assume that $\Gamma(t)$ is sufficiently smooth and closed for all $t \in [0,T]$. First, we need the Reynolds' transport theorem for a smooth vector field $\mathbf{f}$ defined in $\bigcup_{t\in[0,T]} \Omega(t) \times \{t\}$:

$$\frac{d}{dt} \int\limits_{\Omega(t)} \mathbf{f}\, d\mathbf{x} = \int\limits_{\Omega(t)} \frac{\partial \mathbf{f}}{\partial t}\, d\mathbf{x} + \int\limits_{\Gamma(t)} v_\Gamma \mathbf{f}\, d\mathbf{s}. \tag{3.1}$$

Thanks to the kinematic condition (2.3) on the normal velocity of Γ and the divergence free property of $\mathbf{u}$, one gets from (3.1) the identity

$$\frac{d}{dt} \int\limits_{\Omega(t)} \mathbf{f}\, d\mathbf{x} = \int\limits_{\Omega(t)} \Big(\frac{\partial \mathbf{f}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{f} \Big)\, d\mathbf{x} = \int\limits_{\Omega(t)} \dot{\mathbf{f}}\, d\mathbf{x}, \tag{3.2}$$

where $\dot{f}$ denotes the material derivative of f, $\dot{f} = \frac{\partial f}{\partial t} + (\mathbf{u}\cdot\nabla) f$.

The immediate consequence of (3.2) with $\mathbf{f} = (1,0,0)^T$ is the conservation of the evolving domain volume:

$$|\Omega(t)| = |\Omega(0)| \quad \text{for all } t \in (0,T].$$

For a smooth surface Γ, recall the definition of the surface gradient and divergence:

$$\nabla_\Gamma q = \nabla q - (\mathbf{n}\cdot\nabla q)\mathbf{n} \quad \text{and} \quad \operatorname{div}_\Gamma \mathbf{f} = \operatorname{tr}(\nabla_\Gamma \mathbf{f}),$$

which are the intrinsic surface quantities and do not depend on extensions of a scalar function q and a vector function $\mathbf{f}$ off a surface, see, e.g., [14]. We shall need the following identity for integration by parts over Γ (see [11] for more details):

$$\int\limits_\Gamma \big(q(\operatorname{div}_\Gamma \mathbf{f}) + \mathbf{f}\cdot\nabla_\Gamma q \big)\, d\mathbf{s} = \int\limits_\Gamma \kappa(\mathbf{f}\cdot\mathbf{n}) q\, d\mathbf{s}. \tag{3.3}$$

Since $\Gamma(t)$ passively evolves with the fluid velocity $\mathbf{u}$, the Leibniz formula gives for sufficiently smooth f defined on $\bigcup_{t\in[0,T]} \Gamma(t)\times\{t\}$ the relation

$$\frac{d}{dt}\int\limits_{\Gamma(t)} f\,d\mathbf{x} = \int\limits_{\Gamma(t)} \left(\dot f + f\,\mathrm{div}_\Gamma\,\mathbf{u}\right) d\mathbf{x}. \tag{3.4}$$

Stability of a numerical method for fluid equations is usually based on a suitable energy estimate for the numerical solution. Let us first devise an appropriate energy balance for the free surface flow problem (2.1)–(2.4). Without loss of generality we may assume $\rho = 1$. Denote $\mathbf{Du} = [\nabla\mathbf{u} + (\nabla\mathbf{u})^T]/2$. Taking a scalar product of (2.1) with $\mathbf{u}$, integrating over Ω, using boundary conditions and employing the transport formula (3.2) yields

$$\frac{d}{dt}\int\limits_{\Omega(t)} \mathbf{u}^2\,d\mathbf{x} + 2\nu\int\limits_{\Omega(t)} |\mathbf{Du}|^2\,d\mathbf{x} + \tau\int\limits_{\Gamma(t)} \kappa(\mathbf{n}\cdot\mathbf{u})\,d\mathbf{s} = \int\limits_{\Omega(t)} (\mathbf{g}\cdot\mathbf{u})\,d\mathbf{x}, \tag{3.5}$$

where $|\mathbf{Du}|^2 = \sum_{1\le i,j\le 3} |(\mathbf{Du})_{ij}|^2$. Now we apply the surface integration by parts rule (3.3) with $q = 1$ to treat the surface term and further use the Leibniz formula (3.4) with $f = 1$:

$$\int\limits_\Gamma \kappa(\mathbf{n}\cdot\mathbf{u})\,d\mathbf{s} = \int\limits_\Gamma \mathrm{div}_\Gamma\,\mathbf{u}\,d\mathbf{s} = \frac{d}{dt}\int\limits_\Gamma 1\,d\mathbf{x} = \frac{d}{dt}|\Gamma(t)|.$$

Substituting these relations into equality (3.5) and integrating in time leads to the following energy balance for the free surface flow problem (2.1)–(2.4):

$$\int\limits_{\Omega(t)} \mathbf{u}^2\,d\mathbf{x} + 2\nu\int\limits_0^t\int\limits_{\Omega(t')} |\mathbf{Du}|^2\,d\mathbf{x}\,dt' + \tau|\Gamma(t)| = \int\limits_{\Omega(0)} \mathbf{u}^2\,d\mathbf{x} + \tau|\Gamma(0)| + \int\limits_0^t\int\limits_{\Omega(t')} (\mathbf{g}\cdot\mathbf{u})\,d\mathbf{x}\,dt' \quad \text{for all } t\in[0,T]. \tag{3.6}$$

All terms in (3.6) have a clear physical meaning: the total kinetic energy $\|\mathbf{u}\|^2_{\Omega(t)}$ is in balance with the surface free energy $\tau|\Gamma(t)|$ up to the viscous dissipation term and the external forces work.

Since there is no explicit dissipation mechanism for the free surface energy in (3.6), it is not easy to obtain from (3.6) a priori estimates for a solution which would be sufficient for showing the (local) well-posedness of the problem. Thus, to show the (local) existence of a unique solution, Solonnikov [37] applied the transformation to the Lagrangian coordinates. While such a transformation is standard in mathematical analysis of fluid problems with free surfaces, numerical methods are commonly applied to the Eulerian formulation of the free surface problem rather than to the Lagrangian formulation. For the reasons outlined above, a rigorous stability and error analysis of such Eulerian numerical methods is problematic and some simplifying assumptions have to be made.

For the sake of completeness, we recall that free surface flows with surface tension forces satisfy balances of global momentum and angular momentum. Integrating the momentum equation in (2.1) over $\Omega(t)$, employing (3.2) with $\mathbf{f} = \mathbf{u}$ and noting that $\int_\Gamma \kappa n_i\,d\mathbf{s} = 0$ yields the balance of the total momentum:

$$\int\limits_{\Omega(t)} \mathbf{u}\,d\mathbf{x} = \int\limits_{\Omega(0)} \mathbf{u}_0\,d\mathbf{x} + \int\limits_0^t\int\limits_{\Omega(t')} \mathbf{g}\,d\mathbf{x}\,dt' \quad \text{for all } t\in(0,T].$$

We need the following equalities:

$$(\mathbf{u}\dot\times\mathbf{x}) = \dot{\mathbf{u}}\times\mathbf{x} + \mathbf{u}\times(\mathbf{u}\cdot\nabla\mathbf{x}) = \dot{\mathbf{u}}\times\mathbf{x} + \mathbf{u}\times\mathbf{u} = \dot{\mathbf{u}}\times\mathbf{x} \quad\text{and}\quad \mathrm{div}([\mathbf{Du}]\times\mathbf{x}) = (\mathrm{div}\,\mathbf{Du})\times\mathbf{x}, \tag{3.7}$$

where the vector product with a matrix is understood column-wise. Taking the vector product of the momentum equation in (2.1) with $\mathbf{x}$ and using the identities in (3.7), one finds after integration over $\Omega(t)$:

$$\frac{d}{dt}\int\limits_{\Omega(t)} \mathbf{u}\times\mathbf{x}\,d\mathbf{x} + \tau\int\limits_{\Gamma(t)} \kappa(\mathbf{n}\times\mathbf{x})\,d\mathbf{x} = \int\limits_{\Omega(t)} \mathbf{g}\times\mathbf{x}\,d\mathbf{x}.$$

Denote by $[\mathbf{x}]_\times$ a 3×3 skew-symmetric matrix, such that $[\mathbf{x}]_\times \mathbf{a} = \mathbf{x} \times \mathbf{a}$ for any $\mathbf{a} \in \mathbb{R}^3$. One readily checks that $\operatorname{div}_\Gamma [\mathbf{x}]_\times = 0$, where the divergence is applied for each row of the matrix. Therefore, equality (3.3) implies that the term $\int_\Gamma \kappa(\mathbf{n} \times \mathbf{x})\, d\mathbf{s}$ vanishes. Thus we get the balance of the angular momentum:

$$\int_{\Omega(t)} \mathbf{u} \times \mathbf{x}\, d\mathbf{x} = \int_{\Omega(0)} \mathbf{u}_0 \times \mathbf{x}\, d\mathbf{x} + \int_0^t \int_{\Omega(t')} \mathbf{g} \times \mathbf{x}\, d\mathbf{x}\, dt' \quad \text{for all } t \in (0, T].$$

Both momentum and angular momentum are not sign definite quantities and so their balances do not directly yield a priori estimates useful for stability. Still, from the analysis viewpoint, they are helpful to establish the control over the L^2-norm of $\mathbf{u}$ by the L^2-norm of $\mathbf{D}\mathbf{u}$ via Korn's inequality.

Finally, we note that although the global vorticity of the free-surface flow is trivially conserved in the absence of source terms, the global balance of enstrophy or helicity should account for the generation of vorticity along free surface [23]. In our a priori estimates we do not need these balances.

4 Numerical Time Integration and Energy Stability Bound

Several numerical methods have been proposed for the time integration of (2.1) and (2.5). Numerical approaches range from fully implicit schemes to fractional step methods. Fully implicit schemes provide better stability at the expense of several nested iterative processes [14]. In this paper, we consider a semi-implicit spitting method based on semi-Lagrangian approach. The algorithm is built on the well-known splitting procedures due to Chorin, Yanenko, Pironneau and others, see, for example, [5, 31]. For the sake of presentation and analysis, in this section we suppress spacial discretization and postpone some important implementation details. We shall discuss spacial discretization and practical implementation later in Section 5.

Introduce a grid on $[0, T]$: $t_n = n\Delta t$, $n = 0, \dots, N$, with the discretization parameter $\Delta t = T/N$. We adopt the notation $\mathbf{u}^n$, p^n, ϕ^n for approximations to velocity field, pressure, and level set function at $t = t_n$. Function ϕ^n implicitly defines a numerical fluid domain at time $t = t_n$ through $\Omega_n := \{\mathbf{x} \in \mathbb{R}^3 : \phi^n(\mathbf{x}) < 0\}$. We write $\Gamma_n := \partial\Omega_n$, and $\mathbf{n}^n$, κ^n denote the outward normal vector and the mean curvature of Γ_n.

Functions $\mathbf{u}^0 = \mathbf{u}(t_0)$ and $\phi^0 = \phi(t_0)$ are defined from the initial conditions. For $n = 0, \dots, N-1$ and given $\mathbf{u}^n$, ϕ^n such that $\operatorname{div} \mathbf{u}^n = 0$, we find $\mathbf{u}^{n+1}$, p^{n+1}, ϕ^{n+1} in several steps:

The semi-Lagrangian step: $\Omega_n \to \Omega_{n+1}$, $\mathbf{u}^n \to \mathbf{u}^{n+1}_{\text{aux}}$. Consider a divergence free extension of the velocity to the exterior of fluid domain: $\mathbf{u}^n|_{\Omega_n} \to \tilde{\mathbf{u}}^n|_{\mathbb{R}^3}$. In practice, the extension is performed to a bulk computational domain, rather than $\mathbb{R}^3$. For every $\mathbf{y} \in \mathbb{R}^3$, solve the characteristic equation backward in time:

$$\frac{\partial \mathbf{x}(\tau)}{\partial \tau} = \tilde{\mathbf{u}}^n(\mathbf{x}(\tau)), \quad \mathbf{x}(t_{n+1}) = \mathbf{y}, \quad \text{for } \tau \in [t_{n+1}, t_n]. \tag{4.1}$$

The mapping $\mathbf{X} : \mathbf{y} \to \mathbf{x}(t_n)$ defines an isomorphism on $\mathbb{R}^3$. Now, set

$$\phi^{n+1}(\mathbf{y}) = \phi^n(\mathbf{X}(\mathbf{y})), \quad \mathbf{u}^{n+1}_{\text{aux}}(\mathbf{y}) = \tilde{\mathbf{u}}^n(\mathbf{X}(\mathbf{y})). \tag{4.2}$$

Next we handle viscous terms and project the velocity into a (discretely) divergence-free function subspace and recover new pressure:

The viscous step: Solve for $\hat{\mathbf{u}}^{n+1}$ in Ω_{n+1}:

$$\begin{cases} -\nu \operatorname{div} \mathbf{D}\hat{\mathbf{u}}^{n+1} + \dfrac{1}{\Delta t}[\hat{\mathbf{u}}^{n+1} - \mathbf{u}^{n+1}_{\text{aux}}] = \mathbf{g}(t_{n+1}) & \text{in } \Omega_{n+1}, \\ \mathbf{D}\hat{\mathbf{u}}^{n+1} = 0 & \text{on } \Gamma_{n+1}. \end{cases} \tag{4.3}$$

The projection step: Solve for pressure p^{n+1}:

$$\begin{cases} \Delta p^{n+1} = \dfrac{1}{\Delta t} \operatorname{div} \hat{\mathbf{u}}^{n+1} & \text{in } \Omega_{n+1}, \\ p^{n+1} = \tau\kappa^{n+1} & \text{on } \Gamma_{n+1}. \end{cases} \tag{4.4}$$

Update velocity

$$\mathbf{u}^{n+1} = \widehat{\mathbf{u}}^{n+1} - \Delta t\, \nabla p^{n+1}. \tag{4.5}$$

To see that the numerical scheme (4.1)–(4.5) approximates the system of equations (2.1)–(2.5), we rewrite the scheme as a coupled implicit method for a quasi-compressible approximation of the fluid motion equations. Substituting (4.1) in (4.3) and further using (4.4), (4.5) with the index shifted $n \to n-1$, we obtain

$$\begin{cases} \dfrac{\widehat{\mathbf{u}}^{n+1}(\cdot) - \widehat{\mathbf{u}}^{n}(\mathbf{X}(\cdot))}{\Delta t} - \nu \operatorname{div} \mathbf{D}\widehat{\mathbf{u}}^{n+1} + \nabla p^{n}(\mathbf{X}(\cdot)) = \mathbf{g}(t_{n+1}) \\ \qquad \operatorname{div} \widehat{\mathbf{u}}^{n+1} - \Delta t \Delta p^{n+1} = 0 \qquad\qquad \text{in } \Omega_{n+1}, \\ \qquad \phi^{n+1}(\cdot) - \phi^{n}(\mathbf{X}(\cdot)) = 0 \\ \qquad \boldsymbol{\sigma}(\widehat{\mathbf{u}}, p)^{n+1} = \tau(\kappa \mathbf{n})^{n+1} \quad \text{on } \Gamma_{n+1}. \end{cases}$$

The $O(\Delta t)$ term $\Delta t \Delta p^{n+1}$ in the continuity equation shows the formal first order of approximation of the flow model (2.1)–(2.5). For wall-bounded flows, a well-known result (see [33, Theorem 6.1]) is that the classical Chorin scheme is first order convergent in the time-discrete $L^\infty(0,t;L^2(\Omega))$-norm for the velocity and the time-discrete $L^\infty(0,t;H^{-1}(\Omega))$ norm for the pressure. The convergence order of the scheme may decrease for some other type of boundary conditions [15]. For the tension driven free-surface flows, a rigorous convergence analysis is lacking. This is not an unexpected situation, since convergence results typically build on stability analysis, which was largely missing. It is possible to derive a formally second order splitting method following the arguments of [33, 41] for wall-bounded flows. In this paper, we analyze the first order splitting method. In numerical experiments we take time steps subject to the Courant type condition based on $h_{\min}$. We found that decreasing Δt further does not lead to much better accuracy.

We explained in the previous section that additional simplifying assumptions are expected to make the stability analysis of the numerical scheme feasible. Here we analyze the splitting method (4.1)–(4.5) assuming that (4.2) defines such evolution of Ω_n, $n = 1, \dots, N$, that all Γ_n are C^3 smooth, and the principle curvatures as well as their tangential derivatives are uniformly bounded with respect to $\Delta t \in (0, \tau_0]$ for some $\tau_0 > 0$.

Thanks to (4.2) we note that

$$\Omega_{n+1} = \{\mathbf{y} \in \mathbb{R}^3 : \phi^{n+1}(\mathbf{y}) < 0\} = \{\mathbf{y} \in \mathbb{R}^3 : \phi^{n}(\mathbf{X}(\mathbf{y})) < 0\} = \{\mathbf{X}^{-1}(\mathbf{x}) \in \mathbb{R}^3 : \phi^{n}(\mathbf{x}) < 0\} = \mathbf{X}^{-1}(\Omega_n).$$

Since the transport velocity $\tilde{\mathbf{u}}$ in (4.1) is divergence free, the Jacobian $J = |\det(\frac{d\mathbf{X}}{d\mathbf{y}})|$ of $\mathbf{X}$ satisfies $J = 1$. This implies the conservation of the fluid volume,

$$|\Omega_{n+1}| = |\Omega_n|, \tag{4.6}$$

and of the kinetic energy on the semi-Lagrangian step,

$$\int\limits_{\Omega_n} |\mathbf{u}^n(\mathbf{x})|^2\, d\mathbf{x} = \int\limits_{\Omega_{n+1}} |\mathbf{u}^n(\mathbf{X}(\mathbf{y}))|^2 J\, d\mathbf{y} = \int\limits_{\Omega_{n+1}} |\mathbf{u}^{n+1}_{\text{aux}}|^2\, d\mathbf{y}. \tag{4.7}$$

Taking the scalar product of (4.3) with $2\Delta t \widehat{\mathbf{u}}^{n+1}$ and integrating over Ω_{n+1} gives

$$\begin{aligned} \|\widehat{\mathbf{u}}\|^2_{\Omega_{n+1}} + 2\nu\Delta t \|\mathbf{D}\widehat{\mathbf{u}}^{n+1}\|^2_{\Omega_{n+1}} + \|\widehat{\mathbf{u}}^{n+1} - \mathbf{u}^{n+1}_{\text{aux}}\|^2_{\Omega_{n+1}} &= \|\mathbf{u}^{n+1}_{\text{aux}}\|^2_{\Omega_{n+1}} + 2\Delta t (\widehat{\mathbf{u}}^{n+1}, \mathbf{g})_{\Omega_{n+1}} \\ &\le \|\mathbf{u}^{n+1}_{\text{aux}}\|^2_{\Omega_{n+1}} + \Delta t \varepsilon_1 \|\widehat{\mathbf{u}}^{n+1}\|^2_{\Omega_{n+1}} + \Delta t \varepsilon_1^{-1} \|\mathbf{g}\|^2_{\Omega_{n+1}} \end{aligned} \tag{4.8}$$

for some $\varepsilon_1 > 0$ to be chosen later. We use the notation

$$\|\mathbf{D}\widehat{\mathbf{u}}\|^2_{\Omega_{n+1}} = \int\limits_{\Omega_{n+1}} |\mathbf{D}\widehat{\mathbf{u}}|^2\, d\mathbf{x}.$$

Taking the scalar product of (4.5) with $2\Delta t \mathbf{u}^{n+1}$, integrating over Ω_{n+1} and using the Dirichlet pressure boundary condition from (4.4) yields

$$\|\mathbf{u}^{n+1}\|^2_{\Omega_{n+1}} + \|\widehat{\mathbf{u}}^{n+1} - \mathbf{u}^{n+1}\|^2_{\Omega_{n+1}} = \|\widehat{\mathbf{u}}^{n+1}\|^2_{\Omega_{n+1}} + 2\Delta t\, \tau \int\limits_{\Gamma_{n+1}} \kappa^{n+1} (\mathbf{u} \cdot \mathbf{n})^{n+1}\, d\mathbf{s}. \tag{4.9}$$

Recall the following trace inequality (see, e.g., [13]):

$$\|\mathbf{u}\cdot\mathbf{n}\|_{H^{-\frac{1}{2}}(\Gamma)} \le C_T(\Omega)\big(\|\mathbf{u}\|_\Omega + \|\operatorname{div}\mathbf{u}\|_\Omega\big).$$

Applying this trace inequality together with the Cauchy inequality and the assumption on the smoothness of Γ_n, we treat the boundary term in (4.9) as

$$\begin{aligned}\left|\int\limits_{\Gamma_{n+1}} \kappa^{n+1}(\mathbf{u}\cdot\mathbf{n})^{n+1}\,d\mathbf{s}\right| &\le \|\kappa^{n+1}\|_{H^{\frac{1}{2}}(\Gamma_{n+1})}\|(\mathbf{u}\cdot\mathbf{n})^{n+1}\|_{H^{-\frac{1}{2}}(\Gamma_{n+1})}\\ &\le \frac{1}{4\varepsilon_2}\|\kappa^{n+1}\|^2_{H^{\frac{1}{2}}(\Gamma_{n+1})} + \varepsilon_2\|(\mathbf{u}\cdot\mathbf{n})^{n+1}\|^2_{H^{-\frac{1}{2}}(\Gamma_{n+1})}\\ &\le c\,\varepsilon_2^{-1} + C_T(\Omega_{n+1})\,\varepsilon_2\|\mathbf{u}^{n+1}\|^2_{\Omega_{n+1}}\end{aligned} \tag{4.10}$$

for some $\varepsilon_2 > 0$ to be chosen later. In the last inequality of (4.10) we also used that $\mathbf{u}^{n+1}$ is divergence free. We use (4.10) to estimate the boundary terms in (4.9) and apply the triangle inequality to obtain

$$\begin{aligned}\|\mathbf{u}^{n+1}\|^2_{\Omega_{n+1}} + \|\widehat{\mathbf{u}}^{n+1}-\mathbf{u}^{n+1}\|^2_{\Omega_{n+1}} &\le \|\widehat{\mathbf{u}}^{n+1}\|^2_{\Omega_{n+1}} + c\,\tau\varepsilon_2^{-1}\Delta t + C_T(\Omega_{n+1})\,\varepsilon_2\,\Delta t\|\mathbf{u}^{n+1}\|^2_{\Omega_{n+1}}\\ &\le \big(1 + C_T(\Omega_{n+1})\,\varepsilon_2\,\Delta t\big)\|\widehat{\mathbf{u}}^{n+1}\|^2_{\Omega_{n+1}} + c\,\tau\varepsilon_2^{-1}\Delta t + C_T(\Omega_{n+1})\,\varepsilon_2\,\Delta t\|\widehat{\mathbf{u}}^{n+1}-\mathbf{u}^{n+1}\|^2_{\Omega_{n+1}}.\end{aligned}$$

For $\varepsilon_2 \le [C_T(\Omega_{n+1})\Delta t]^{-1}$ this yields the estimate

$$\|\mathbf{u}^{n+1}\|^2_{\Omega_{n+1}} \le \big(1 + C_T(\Omega_{n+1})\,\varepsilon_2\,\Delta t\big)\|\widehat{\mathbf{u}}^{n+1}\|^2_{\Omega_{n+1}} + c\,\tau\varepsilon_2^{-1}\Delta t. \tag{4.11}$$

We employ (4.11) in (4.8) and get

$$\|\mathbf{u}^{n+1}\|^2_{\Omega_{n+1}} + 2\nu\Delta t\|\mathbf{D}\widehat{\mathbf{u}}^{n+1}\|^2_{\Omega_{n+1}} \le \|\mathbf{u}^{n+1}\|^2_{\Omega_n} + \Delta t\big((\varepsilon_1 + \varepsilon_2 C_T(\Omega_{n+1}))\|\widehat{\mathbf{u}}^{n+1}\|^2_{\Omega_{n+1}} + \varepsilon_1^{-1}\|\mathbf{g}\|^2_{\Omega_{n+1}} + c\,\varepsilon_2^{-1}\tau\big). \tag{4.12}$$

Now we need to estimate $\|\widehat{\mathbf{u}}^{n+1}\|_{\Omega_{n+1}}$ from above by the diffusion term $\|\mathbf{D}\widehat{\mathbf{u}}^{n+1}\|_{\Omega_{n+1}}$. This can be done with the help of Korn's and Poincaré's inequalities if $\widehat{\mathbf{u}}$ satisfies certain gauge conditions which exclude rigid motions. To deduce such constraints on the solution, we assume that external forces are total momentum and angular momentum free,

$$\int\limits_{\Omega(t)} \mathbf{g}\,d\mathbf{x} = \int\limits_{\Omega(t)} \mathbf{g}\times\mathbf{x}\,d\mathbf{x} = 0 \quad \text{for all } t\in[0,T]. \tag{4.13}$$

In this case, the solution of the differential problem conserves momentum and angular momentum. Below we check that our splitting scheme respects these conservation properties. Similar to the conservation of energy in (4.7), one verifies that the semi-Lagrangian step conserves momentum and angular momentum:

$$\int\limits_{\Omega_n} \mathbf{u}^n\,d\mathbf{x} = \int\limits_{\Omega_{n+1}} \mathbf{u}^{n+1}_{\text{aux}}\,d\mathbf{x}, \quad \int\limits_{\Omega_n} \mathbf{u}^n\times\mathbf{x}\,d\mathbf{x} = \int\limits_{\Omega_{n+1}} \mathbf{u}^{n+1}_{\text{aux}}\times\mathbf{x}\,d\mathbf{x}. \tag{4.14}$$

Further, we integrate (4.3) over Ω_{n+1} and use boundary conditions to show

$$\int\limits_{\Omega_{n+1}} \widehat{\mathbf{u}}^{n+1}\,d\mathbf{x} = \int\limits_{\Omega_{n+1}} \mathbf{u}^{n+1}_{\text{aux}}\,d\mathbf{x}. \tag{4.15}$$

Taking the vector product of (4.3) with $\mathbf{x}$, integrating over Ω_{n+1}, using boundary conditions and the second identity in (3.7) bring us to

$$\int\limits_{\Omega_{n+1}} \widehat{\mathbf{u}}^{n+1}\times\mathbf{x}\,d\mathbf{x} = \int\limits_{\Omega_{n+1}} \mathbf{u}^{n+1}_{\text{aux}}\times\mathbf{x}\,d\mathbf{x}. \tag{4.16}$$

Finally, integrating (4.5) and the vector product of (4.5) with $\mathbf{x}$ over $\Omega(t)$ and using the same arguments to handle the boundary terms as in the previous section leads to

$$\int\limits_{\Omega_{n+1}} \mathbf{u}^{n+1}\,d\mathbf{x} = \int\limits_{\Omega_{n+1}} \widehat{\mathbf{u}}^{n+1}\,d\mathbf{x}, \quad \int\limits_{\Omega_{n+1}} \mathbf{u}^{n+1}\times\mathbf{x}\,d\mathbf{x} = \int\limits_{\Omega_{n+1}} \widehat{\mathbf{u}}^{n+1}\times\mathbf{x}\,d\mathbf{x}. \tag{4.17}$$

Combining (4.14)–(4.17) we show the conservation property for the solution of the splitting scheme:

$$\int\limits_{\Omega(t_n)} \mathbf{u}^n \, d\mathbf{x} = \int\limits_{\Omega(0)} \mathbf{u}_0 \, d\mathbf{x}, \quad \int\limits_{\Omega(t_n)} \mathbf{u}^n \times \mathbf{x} \, d\mathbf{x} = \int\limits_{\Omega(0)} \mathbf{u}_0 \times \mathbf{x} \, d\mathbf{x} \quad \text{for all } n = 1, \dots, N. \tag{4.18}$$

Without loss of generality, we may assume that the initial velocity field is momentum and angular momentum free:

$$\int\limits_{\Omega(0)} \mathbf{u}_0 \, d\mathbf{x} = 0, \quad \int\limits_{\Omega(0)} \mathbf{u}_0 \times \mathbf{x} \, d\mathbf{x} = 0.$$

Due to the conservation property (4.17)–(4.18), this ensures that for any t_n both $\mathbf{u}$ and $\widehat{\mathbf{u}}$ have zero momentum and angular momentum. In particular, it holds

$$\int\limits_{\Omega_{n+1}} \widehat{\mathbf{u}}^{n+1} \, d\mathbf{x} = 0, \quad \int\limits_{\Omega_{n+1}} \widehat{\mathbf{u}}^{n+1} \times \mathbf{x} \, d\mathbf{x} = 0.$$

This constraint on all rigid motions of the medium is sufficient for the Korn inequality to hold (see [17, Theorem 4]). This and the Poincaré inequality imply

$$\|\widehat{\mathbf{u}}^{n+1}\|_{\Omega_{n+1}} \le C_K(\Omega_{n+1}) \|\mathbf{D}\widehat{\mathbf{u}}^{n+1}\|_{\Omega_{n+1}}. \tag{4.19}$$

We shall assume the constants $C_T(\Omega_n)$, $C_K(\Omega_n)$ are uniformly bounded for all $\Delta t \in (0, \tau_0]$ and $n = 1, \dots, N$. Therefore we can apply (4.19) in (4.12) and choose ε_1, ε_2 sufficiently small but independent of n and Δt, such that

$$\|\mathbf{u}^{n+1}\|^2_{\Omega_{n+1}} + \nu \Delta t \|\mathbf{D}\widehat{\mathbf{u}}^{n+1}\|^2_{\Omega_{n+1}} \le \|\mathbf{u}^n\|^2_{\Omega_n} + c\, \Delta t (\|\mathbf{g}\|^2_{\Omega_{n+1}} + \tau). \tag{4.20}$$

Summing (4.20) over $n = 0, \dots, M-1$, with any M, $1 \le M \le N$, we obtain the stability estimate for our semi-discrete scheme:

$$\|\mathbf{u}^M\|^2_{\Omega_M} + \sum_{n=1}^{M} \nu \Delta t \|\mathbf{D}\widehat{\mathbf{u}}^n\|^2_{\Omega_n} \le C\, \tau + \|\mathbf{u}_0\|^2_{\Omega_0} + c \sum_{n=1}^{M} \Delta t \|\mathbf{g}\|^2_{\Omega_n}, \tag{4.21}$$

with a constant C independent of Δt, $\mathbf{g}$, $\mathbf{u}_0$, τ, ν (compare with the energy balance (3.6) of the smooth solution to the differential flow problem).

We summarize the results in the following theorem.

Theorem 4.1. *Assume that* (4.2) *defines such evolution of* Ω_n, $n = 1, \dots, N$, *that all* Γ_n *are* C^3 *smooth. Assume that the principle curvatures, their tangential derivatives and constants* $C_T(\Omega_n)$, $C_K(\Omega_n)$ *from the trace and Korn's inequalities are uniformly bounded with respect to* $\Delta t \in (0, \tau_0]$ *for some* $\tau_0 > 0$. *If the volume forces satisfy* (4.13), *then the solution to the numerical splitting scheme* (4.1)–(4.5) *satisfies the conservation of momentum, angular momentum* (4.18) *and the energy stability bound* (4.21).

Remark 4.1. In [3], Bänsch analyzed a finite element method for the free boundary Navier–Stokes problem subject to surface tension forces. Let us comment on similarities and differences of that analysis and the one presented here. Both stability analyses rely on a priori assumptions on the regularity of free surface evolution. The assumptions in [3] are somewhat weaker than ours. This is partially because that paper considered an implicit coupling of surface evolution and fluid dynamics on every time step in the framework of a space-time finite element method. The method studied here decouples these processes on every time step. Furthermore, in [3] conservation properties of the discretization method were not addressed and, therefore, the author had to use a Gronwall argument to obtain an energy bound similar to (4.21). This led to an exponential growing stability constant, which is not the case in the present analysis. Time stepping schemes studied in both papers have consistency error $O(\Delta t)$.

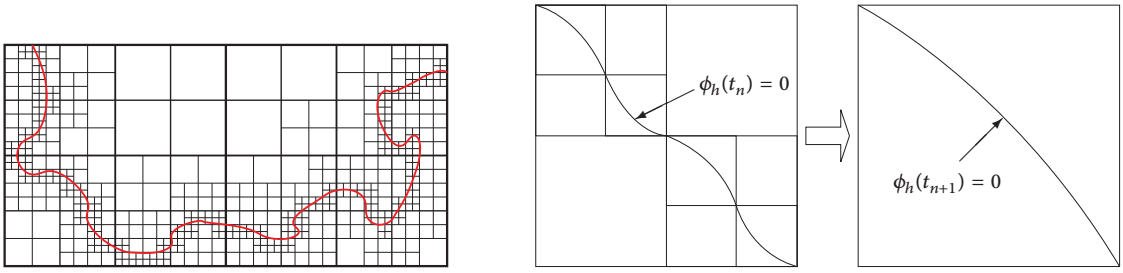

Figure 1. Left: 2D quadtree grid adapted to free boundary. Right: The loss of discrete free surface geometric information when ϕ_h is transported from a region with finer mesh to the one with a coarser mesh.

5 Spacial Discretization and Implementation Details

Practical implementation of the time-splitting method for computing free surface flows needs a choice of a numerical integrator for the semi-Lagrangian step, spacial discretization and several further important ingredients such as an approximation of surface tension forces, mesh adaptation, and handling a discrete level set function. This section addresses these and a few other implementation details. It should be noted right away that the conservation properties of the splitting scheme may be disturbed once a spacial discretization is introduced. Despite some existing interesting work on conservative (fully discrete) schemes based on the semi-Lagrangian method (see [21] and references therein), building a fully discrete method for incompressible viscous flows that conserves momentum, angular momentum, and yields a correct balance of energy, is largely an open question.

5.1 Spacial Discretization and Mesh Adaptation

A possibly complex geometry of the free surface and accurate approximation of the 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 need less computational resources. However, such meshes have to be dynamically refined and coarsened if the free surface evolves. The remeshing is, in general, a CPU time and memory demanding procedure for consistent regular tetrahedra divisions. 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. Details on how quadtree/octree structures can be efficiently handled computationally are found in [35]. Note that due to the inherited hierarchical structure, data interpolation between two consecutive meshes is straightforward and efficient multilevel techniques can be adopted for the solution of resulting algebraic linear systems, e.g. [2, 6].

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 Γ_{n+1} if the characteristic equation (4.1) is solved on a current grid 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 Γ_{n+1} 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 Γ_{n+1} in the level set part of the algorithm, since the mesh adaptation step is performed *before* the velocity is 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 general, an adaptation strategy may rely on estimators of flow complexity such as local vorticity magnitude. We do not implement such possibly more advanced estimators in the present paper. Rather, on each time iteration and before the semi-Lagrangian step we refine all cubic cells intersected by Γ_n and by (predicted) Γ_{n+1} so that all these cells have the width $h_{\min}$. All other cells are marked for coarsening. The coarsening is performed in such a way that the octree remains balanced (two neighboring cells may differ in size at most by

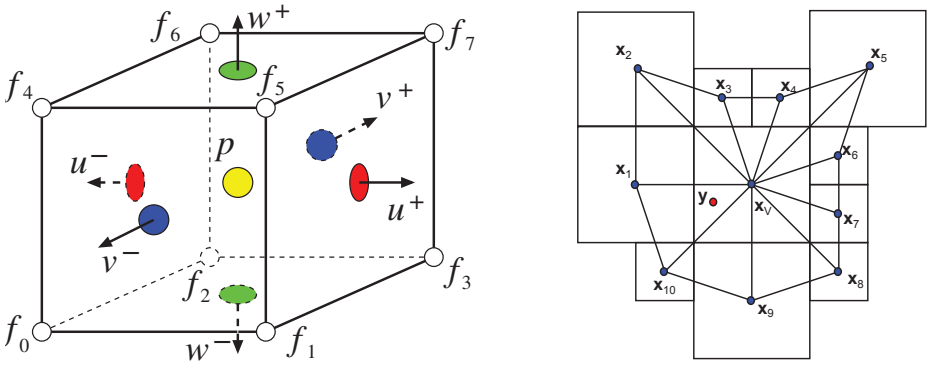

Figure 2. Left: Location of the variables in each cubic cell: p is the pressure, $\{u^{\pm}, v^{\pm}, w^{\pm}\}$ are the velocity components, f are the scalar functions, e.g. level set function. Right: $u_h(\mathbf{y})$ is defined by a linear interpolation based on the fan triangulation with the center 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.

a factor of two) and the maximum cell width in the fluid domain is $h_{\max}$ and in the rest of the computational domain it is h_{ext}. Parameters $h_{\min}$, $h_{\max}$, and h_{ext} are provided in advance.

For the spacial discretization, we apply a finite difference method. The use of cubic cells appeals to the staggered distribution of velocity and pressure degrees of freedom: The pressure is approximated in cell centers, velocity components are approximated in face centers; the level set function is approximated in cell vertices (see Figure 2). The resulting discretization can be observed as an extension to octree meshes of a classical MAC scheme for fluid equations [16, 20] and was studied in [28]. For the discrete gradient, divergence and diffusion terms we use second order approximations with compact stencil as described in [26–28].

5.2 Numerical Semi-Lagrangian Step

The characteristic equation (4.1) is integrated numerically with the second order accuracy:

$$\mathbf{x}\Big(t_n + \frac{\Delta t}{2}\Big) = \mathbf{y} - \frac{\Delta t}{2}\widetilde{\mathbf{u}}^n(\mathbf{y}), \quad \mathbf{x}(t_n) = \mathbf{x}\Big(t_n + \frac{\Delta t}{2}\Big) - \frac{\Delta t}{2}\Big[3\widetilde{\mathbf{u}}^n\Big(\mathbf{x}\Big(t_n + \frac{\Delta t}{2}\Big)\Big) - \widetilde{\mathbf{u}}^{n-1}\Big(\mathbf{x}\Big(t_n + \frac{\Delta t}{2}\Big)\Big)\Big]. \tag{5.1}$$

The space point $\mathbf{x}(t_n + \frac{\Delta t}{2})$ is not necessarily a grid point, therefore the computation of $\widetilde{\mathbf{u}}^n(\mathbf{x}(t_n + \frac{\Delta t}{2}))$ and $\widetilde{\mathbf{u}}^{n-1}(\mathbf{x}(t_n + \frac{\Delta t}{2}))$ requires an interpolation of velocity values. This interpolation procedure is described below. Once (5.1) is computed, we assign $\phi^{n+1}_{\text{aux}}(\mathbf{y})$ and $\mathbf{u}^{n+1}_{\text{aux}}(\mathbf{y})$ according to

$$\phi^{n+1}_{\text{aux}}(\mathbf{y}) = \phi^n(\mathbf{X}(\mathbf{y})), \quad \mathbf{u}^{n+1}_{\text{aux}}(\mathbf{y}) = \mathbf{u}^n(\mathbf{x}(t_n)). \tag{5.2}$$

To compute $\phi^n(\mathbf{x}(t_n))$ and $\mathbf{u}^n(\mathbf{x}(t_n))$ in (5.2) the interpolation is used. Note that at this step the spacial discretization error may perturb the sign distance property of ϕ and the volume conservation (4.6).

A natural choice of interpolation procedure would be, first, to define velocity values in cell vertices by averaging the corresponding velocity nodal values in the neighboring cell faces and, second, to compute the velocity value in any point of a cell as the trilinear interpolation of the vertex values. We use this simple (and fast) procedure when the level set function advection step is performed (level set function is defined in cell vertices and no averaging is needed). 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.

For a given point $\mathbf{y}$ in the 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 $\mathbf{x}_k$, $k = 1, \ldots, m$, $m \le 12$, 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 $u_h(\mathbf{x}_k)$, $k = 1, \dots, m$, 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 $\mathbf{x}_k$, $k = 1, \dots, m$, then $u_h(\mathbf{y})$ is defined by a linear interpolation between the values of u_h in the vertices of this triangle, cf. Figure 2. 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 $\{\mathbf{x}_V, \mathbf{x}_k\}$, $k = 1, \dots, m$.

Remark 5.1. It was found beneficial in some cases to integrate (4.1) more accurately, dividing the time interval $[t_n, t_{n+1}]$ into l subintervals and applying (5.1) on each subinterval. Using $l = 2, 3$ we observed a notable overall accuracy improvement compared to $l = 1$.

5.3 Free Surface Handling

Now we discuss how to handle the free surface and compute its geometric quantities. To find the evolution of the free surface, we apply the semi-Lagrangian method as described in Section 5.2.

5.3.1 Volume Correction

The numerical integration of (2.5) may cause a divergence (loss or gain) of the fluid volume, i.e., the violation of (4.6). 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 (2.5). We use the second order accurate method with the integration step $\frac{\Delta t}{l}$ with $l = 2, 3$. Further increase of l was not found to produce more accurate results. (iii) Correction of the level set function by the method of particles [7]. However, the particle level set method alone improves, but does not guarantee a complete volume conservation [27]. 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 value δ the equation

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

and correcting $\phi^{\mathrm{new}} = \phi - \delta$. The bisection algorithm was used to find δ and a Monte Carlo method was applied to evaluate $\mathrm{meas}\{\mathbf{x} : \phi(\mathbf{x}) < \delta\}$. We note that the volume correction method can be extended to the case of multi-connected fluid domains and more general adjustment functions [25].

5.3.2 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:

$$|\nabla \phi(\mathbf{x})| = 1, \quad \mathbf{x} \in \mathbb{R}^3, \tag{5.3}$$

with the boundary condition on the free surface: $\phi(\mathbf{x}) = 0$ for $\mathbf{x} \in \Gamma(t)$. Keeping the discrete level set function close to the signed distance function 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 the discrete level set function in cell vertices) a local internal surface triangulation is built using the marching cubes technique [18, 22]. The triangulated global approximation of the surface turns out to be a conformal triangulation in space.

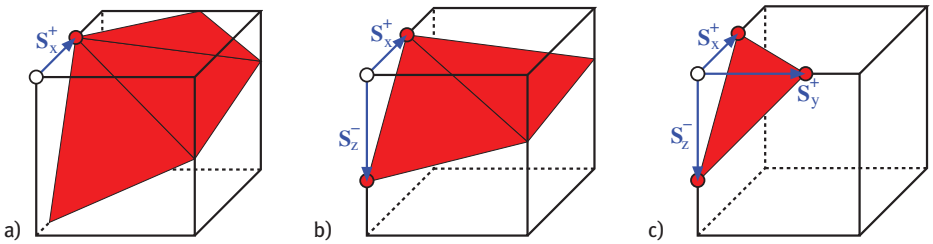

Figure 3. Possible cases for distance reconstruction in an interface cell node.

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 a solution to a discrete counterpart of (5.3) 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.

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.

(i) If none of three edges intersects the interface, no action is needed.

(ii) In case of a single intersection (Figure 3a), one computes the distance to that intersection: $d = S_x^+$.

(iii) In case of two intersections (Figure 3b), one evaluates the distance as the height of the appropriate triangle:

$$d = \left(\sqrt{\left(\frac{1}{S_x^+}\right)^2 + \left(\frac{1}{S_z^-}\right)^2}\right)^{-1}.$$

(iv) In case of three intersections (Figure 3c), 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.

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 4). 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 5.2. 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 6.1) and their utilization appears to be crucial for modeling phenomena driven by surface tension forces (see the example in Section 6.3).

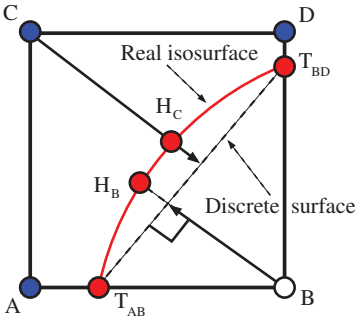

Figure 4. The difference between triangulated isosurface and the true isosurface $\{\phi_h(\mathbf{x}) = 0\}$.

5.3.3 Extension of $\mathbf{u}_h$ from $\Gamma_h(t)$

Finally, for updating the level set function we need an extension of the velocity field from $\Gamma_h(t)$ (t is fixed) to the entire computational domain. We build the normal extension of the 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 $\mathbf{y}^0 = \mathbf{x}$ and 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 $|\mathbf{y}^{n+1} - \mathbf{y}^n| \le \varepsilon$ 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 $\alpha = (\sqrt{5} - 1)/2$.

6 Numerical Experiments

To illustrate the performance of the numerical method, we perform several tests with known analytical solutions and consider the standard benchmark problem for free-surface flow numerical solvers, the 3D oscillating droplet problem.

6.1 Accuracy of Redistancing Methods

In the first experiment, we study the accuracy of the discrete Eikonal solvers (redistancing methods) introduced in Section 5.3.2. We choose the surface Γ to be 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. We compare numerical solutions to the exact solution of (5.3) and compute the error only in the interface cells T such that $T \cap \Gamma \neq \emptyset$. Let $\mathcal{V}(V)$ denote the set of all vertices of a cell V. We use the following indicators to evaluate discretization errors of numerical solutions:

$$\varepsilon_i^C = \frac{\max_V \max_{\mathbf{x}\in\mathcal{V}(V)} |\phi_h^i(\mathbf{x}) - \phi(\mathbf{x})|}{\max_V \max_{\mathbf{x}\in\mathcal{V}(V)} |\phi(\mathbf{x})|}, \quad \varepsilon_i^L = \left(\frac{\sum_V |V| \sum_{\mathbf{x}\in\mathcal{V}(V)} |\phi_h^i(\mathbf{x}) - \phi(\mathbf{x})|^2}{\sum_V |V| \sum_{\mathbf{x}\in\mathcal{V}(V)} \phi^2(\mathbf{x})} \right)^{1/2},$$

where ϕ_h^i is the discrete signed distance function found by the redistancing method number i, $i = 1, 2, 3$. The indicators resemble the scaled C- and L^2-norms of the error, respectively, while the scaling factors account for vanishing of ϕ in interface cell nodes when the grid is refined: $\max_{\mathbf{x}\in\mathcal{V}(V)} |\phi(\mathbf{x})| = O(h_{\min})$.

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.

6.2 Accuracy of Normal Vectors and Mean Curvature Reconstruction

Recall that the unit outward normal can be computed via the level set function: $\mathbf{n}_\Gamma = \nabla\phi/|\nabla\phi|$ on $\Gamma(t)$. The mean curvature of the interface can be defined as the divergence of the normal vector, $\kappa(\phi) = \operatorname{div}\mathbf{n} = \operatorname{div}(\nabla\phi/|\nabla\phi|)$. Thus the numerical approximation of the mean curvature is computed as follows. First, $\nabla_h\phi_h$ is

$h_{\min}$	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.

$h_{\min}$	ε_∇^C	ε_∇^L	ε_κ^C	ε_κ^L
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−5

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

computed in cell vertices using the second order approximation of the gradient through the Taylor expansion in all possible combinations of octree cells sharing the node. Further, the computed values are averaged in face centers. Once $\nabla_h\phi_h/|\nabla_h\phi_h|$ is known in face centers, $\kappa_h(\phi_h) = \operatorname{div}_h \nabla_h\phi_h/|\nabla_h\phi_h|$ is computed in cell centers by standard second order central differences. Since $\nabla\phi$ is computed with second order accuracy, $\kappa(\phi)$ is approximated with the first order.

To test the discretization errors for $\mathbf{n}$ and κ, we take the same spherical interface and introduce the following indicators of the level set gradient and mean curvature discretization 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

$$\varepsilon_\nabla^C = \max_V \max_{\mathbf{x}\in\mathcal{V}(V)} \left|\nabla_h\phi(\mathbf{x}) - \nabla\phi(\mathbf{x})\right|, \qquad \varepsilon_\nabla^L = \left(\frac{\sum_V |V| \sum_{\mathbf{x}\in\mathcal{V}(V)} |\nabla_h\phi(\mathbf{x}) - \nabla\phi(\mathbf{x})|^2}{\sum_V |V| \sum_{\mathbf{x}\in\mathcal{V}(V)} |\nabla\phi(\mathbf{x})|^2}\right)^{1/2},$$

$$\varepsilon_\kappa^C = \max_{\mathbf{x}\in C} \left|\kappa_h(\phi)(\mathbf{x}) - \nabla\cdot\left(\nabla\phi(\mathbf{x})/|\nabla\phi(\mathbf{x})|\right)\right|, \quad \varepsilon_\kappa^L = \left(\frac{\sum_V |V| \sum_{\mathbf{x}\in C} \left[\kappa_h(\phi)(\mathbf{x}) - \nabla\cdot\frac{\nabla\phi(\mathbf{x})}{|\nabla\phi(\mathbf{x})|}\right]^2}{\sum_V |V| \sum_{\mathbf{x}\in C} \nabla\cdot(\nabla\phi(\mathbf{x})/|\nabla\phi(\mathbf{x})|)^2}\right)^{1/2},$$

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

Table 2 shows the discretization errors for surface curvature and gradient of ϕ. The gradient discretization demonstrates the second order accuracy in both the C-norm and the scaled L^2-norm. Naturally, the mean curvature approximation is only first order accurate in both norms.

6.3 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 $\mathbf{g} = 0$. The initial shape of the droplet is a perturbation of a sphere: In spherical coordinates (r, θ, φ) the initial shape is given by

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

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. The fluid motion in this experiment is solely driven by the surface tension forces. This oscillatory behavior illustrates the continuous transition between kinetic and surface free energy, as seen from the energy balance (3.6), while the expected exponential decay of oscillations is due to the dissipation energy through diffusion in (3.6).

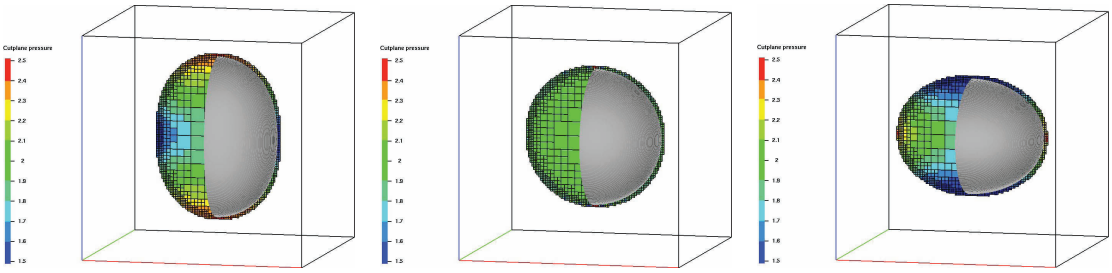

Figure 5. The droplet shape, grid and pressure distribution for $t = \{0, 0.726, 1.286\}$.

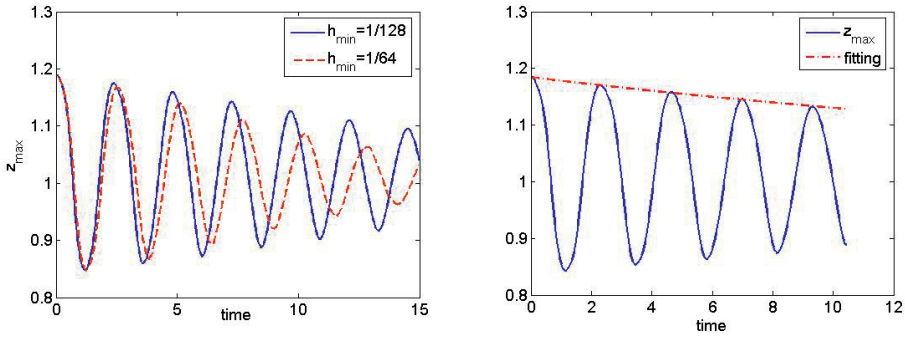

Figure 6. The left picture compares the top tip trajectories on the z axis for $h_{\min} = \frac{1}{64}$ and $h_{\min} = \frac{1}{128}$. The right picture shows the droplet tip trajectory on the z axis for $h_{\min} = \frac{1}{256}$, $h_{\max} = \frac{1}{16}$ and the fitted curve.

Therefore, the quality of the numerical solution is sensitive to how accurate the conservation laws are enforced. This test is also challenging for the surface tension approximation and the free surface capturing method where an accurate and smooth curvature field is critical. For these reasons, the oscillating droplet problem often serves as a benchmark test for free surface and two-phase fluid flow solvers, see, e.g., [3, 8–10, 34]. Two statistics are of common interest: The droplet oscillation period T and the oscillations damping factor δ. Assuming the perturbation is small ($\varepsilon \ll 1$), a linear stability analysis from [19] predicts the period and the damping factor according to

$$T_{\text{ref}} = 2\pi\sqrt{\frac{\rho r_0^3}{8\tau}}, \quad \delta_{\text{ref}} = \frac{r_0^2}{5\nu}. \tag{6.1}$$

We solve 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 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 repeat experiments with gradually refined $h_{\max} = h_{\min}/2$ and observed similar results (not shown), although the timings increased significantly. The third method (see Section 5.3.2) of discrete level set function re-initialization was used. We note that the second method gave qualitatively similar results, whereas the first one resulted in non-physical oscillatory solutions.

We experiment with $\nu = 0$ which allows us both to look at the quality of surface tension forces approximation and to quantify the effect of numerical diffusion of the scheme. Figure 5 illustrates the droplet shape, the grid ($h_{\min} = \frac{1}{128}$, $h_{\max} = \frac{1}{16}$) and the pressure distribution at times 0, 0.726, and 1.286. Further, the trajectories of the droplet tips are shown in Figure 6. 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 which is 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 fails 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 shown 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.

$h_{\min}$	T	$T - T_{\text{ref}}$	δ	ν_{num}	#cells	$\text{Time}_{\Delta t}$
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 time step versus the total number of cells in $\Omega_h(0)$.

This suggests that the scheme remains somewhat dissipative. We note that in general the particle method can be used to reduce the numerical dissipation, however for the present benchmark test the particle correction of the level set function was found to produce large enough disturbance of ϕ_h leading to inaccurate modeling of the surface tension forces. 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, similarly to (6.1), it holds $\delta \approx r_0^2(5\nu_{\text{num}})^{-1}$.

The energy balance (3.6) suggests that 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 Figure 7), then the total energy is not completely conserved by the numerical scheme. Recall that for the energy we were able to show only a stability estimate rather than a suitable (discrete) balance property. Spacial discretization also introduces some extra numerical dissipation which destroys the energy balance. In Section 4 we checked that the time-splitting scheme conserves momentum and angular momentum. The spatial discretization is not conservative with respect to these quantities. Nevertheless, Figure 8 shows that in computations the deviation from initial total momentum and angular momentum (both are zero for the oscillating droplet problem) remains reasonably small. To the best of our knowledge, building conservative (energy, momentum and angular momentums) adaptive FD 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 2 × 2.66 GHz Intel Quad-Core Xeon X5355 processors with 8 Gb 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 simulation of free surface flows. The numerical model is based on the Navier–Stokes equations coupled with a level set function equation discretized on dynamically refined/coarsened octree cartesian grids. Other key components of the approach are the splitting method for time discretization, the redistancing algorithm for the discrete level set function, and accurate approximation of free surface normal vector and mean curvature. Several redistancing and normal vector/mean curvature approximation techniques were comparatively studied. The redistancing method based on the 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. A stability estimate for semi-discrete solutions of the time splitting method was proved. The proof is based on the semi-discrete balance of kinetic and surface free energy. It is interesting to note that the conservation of the momentum and angular momentum property turn out to be important to show the stability of the numerical method. This is in contrast to the situation with enclosed flows, where the stability usually directly follows from an (discrete) energy balance. Probably for this reason, analyzing and improving momentum and angular momentum conservation

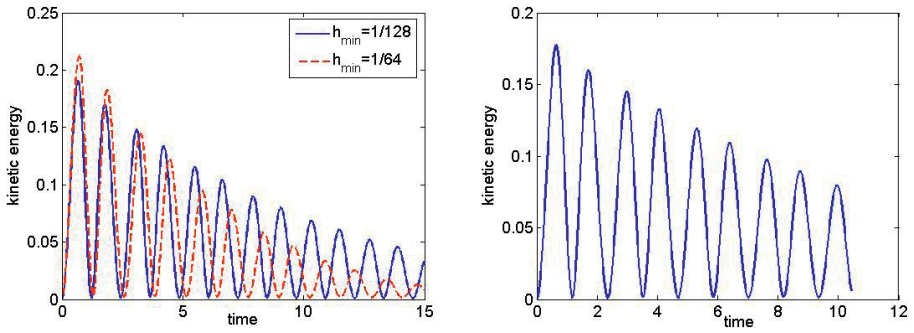

Figure 7. The left picture compares the kinetic energy evolution for $h_{\min} = \frac{1}{64}$ and $h_{\min} = \frac{1}{128}$. The right picture shows the kinetic energy evolution for $h_{\min} = \frac{1}{256}$ and $h_{\max} = \frac{1}{16}$.

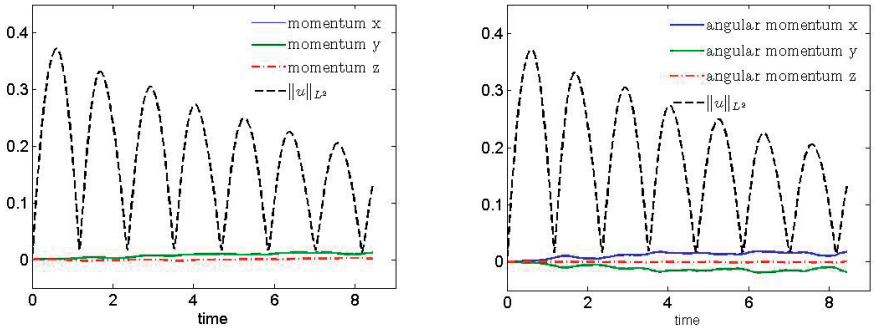

Figure 8. The left picture shows x, y and z momentums. The right picture shows x, y and z angular momentums. The L^2-norm of the velocity is plotted to provide appropriate scaling.

properties of numerical schemes is often overlooked in the literature. Although our stability analysis involved additional assumptions on the smoothness of the evolution of free surface, we consider it as a first important step towards a more complete numerical analysis of free surface flows with surface tension.

Funding: Part of the research on analysis of conservation laws for the free surface flow model and analysis of the conservation and stability properties of the time integration scheme have been supported in part by RFBR grants 12-01-00283, 12-01-33084, 14-01-00830, and the Russian President grant MK 7159.2013.1. The development of discretizations in space and time, free surface handling methods, and numerical studies of the developed solver have been supported by the Russian Science Foundation grant 14-11-00434.

References

[1] D. Adalsteinsson and J. A. Sethian, The fast construction of extension velocities in level set methods, *J. Comput. Phys.* **148** (1999), 2–22.
[2] R. Bank, T. Dupont and 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] A. Chorin, Numerical solution of the Navier–Stokes equations, *Math. Comp.* **22** (1968), 745–762.
[6] V. Dyadechko, Y. Iliash and Y. Vassilevski, Structuring preconditioners for unstructured meshes, *Russian J. Numer. Anal. Math. Modelling* **11** (1996), 139–154.

[7] D. Enright, F. Losasso and R. Fedkiw, A fast and accurate semi-Lagrangian particle level set method, *Comput. Struct.* **83** (2005), 479–490.
[8] M. Francois, S. Cummins, E. Dendy, D. Kothe, J. Sicilian and M. 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.
[9] S. Ganesan and 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.
[10] S. Ganesan and 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.
[11] D. Gilbarg and N. S. Trudinger, *Elliptic Partial Differential Equations of Second Order*, Springer, Berlin, 1998.
[12] I. Ginzburg and 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.
[13] V. Girault and P. A. Raviart, *Finite Element Methods for Navier–Stokes Equations: Theory and Algorithms*, Springer Ser. Comput. Math. 5, Springer, Berlin, 1986.
[14] S. Gross and A. Reusken, *Numerical Methods for Two-Phase Incompressible Flows*, Springer, Berlin, 2011.
[15] J. Guermond, P. Minev and J. Shen, An overview of projection methods for incompressible flows, *Comput. Methods Appl. Mech. Engrg.* **195** (2006), no. 44, 6011–6045.
[16] F. Harlow and J. Welch, Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface, *Phys. Fluids* **8** (1965), 2182–2189.
[17] V. A. Kondrat'ev and O. A. Oleinik, Boundary-value problems for the system of elasticity theory in unbounded domains. Korn's inequalities, *Russ. Math. Surv.* **43** (1988), 65–119.
[18] J.-O. Lachaud, Topologically defined iso-surfaces, in: *Discrete Geometry for Computer Imagery* (DGCI'96), Lecture Notes in Comput. Sci. 1176, Springer, Berlin (1996), 245–256.
[19] H. Lamb, *Hydrodynamics*, Cambridge University Press, Cambridge, 1932.
[20] 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** (1964), no. 3, 69–92.
[21] M. Lentine, J. T. Gretarsson and R. Fedkiw, An unconditionally stable fully conservative semi-Lagrangian method, *J. Comput. Phys.* **230** (2011), 2857–2879.
[22] W. Lorensen and H. Cline, Marching cubes: A high resolution 3D surface construction algorithm, *Computer Graphics* **21** (1987), 163–169.
[23] T. Lundgren and P. Koumoutsakos, On the generation of vorticity at a free surface, *J. Fluid Mech.* **382** (1999), 351–366.
[24] C. Min and F. Gibou, A second order accurate level set method on non-graded adaptive cartesian grids, *J. Comput. Phys.* **225** (2007), 300–321.
[25] F. Mut, G. Buscaglia and E. Dari, A new mass-conserving algorithm for level set redistancing on unstructured meshes, *Mecanica Computacional* **23** (2004), 1659–1678.
[26] K. D. Nikitin, K. M. Terekhov, M. A. Olshanskii and Y. V. Vassilevski, A numerical method for the simulation of free surface flows of viscoplastic fluid in 3D, *J. Comput. Math.* **29** (2011), 605–622.
[27] K. D. Nikitin and Y. V. Vassilevski, Free surface flow modelling on dynamically refined hexahedral meshes, *Russian J. Numer. Anal. Math. Modelling* **23** (2008), 469–485.
[28] M. A. Olshanskii, K. M. Terekhov and Y. V. Vassilevski, An octree-based solver for the incompressible Navier–Stokes equations with enhanced stability and low dissipation, *Comput. Fluids* **84** (2013), 231–246.
[29] S. Osher and R. Fedkiw, *Level Set Methods and Dynamic Implicit Surfaces*, Appl. Math. Sci. 153, Springer, New York, 2002.
[30] J. E. J. Pilliod and E. G. Puckett, Second-order accurate volume-of-fluid algorithms for tracking material interfaces, *J. Comput. Phys.* **199** (2004), 465–502.
[31] O. Pironneau, On the transport-diffusion algorithm and its applications to the Navier–Stokes equations, *Numer. Math.* **28** (1982), 309–332.
[32] S. Popinet, An accurate adaptive solver for surface-tension-driven interfacial flows, *J. Comput. Phys.* **228** (2009), 5838–5866.
[33] A. Prohl, *Projection and quasi-compressibility methods for solving the incompressible Navier–Stokes equations*, BG Teubner, Stuttgart, 1997.
[34] S. Quan and D. Schmidt, A moving mesh interface tracking method for 3D incompressible two-phase flows, *J. Comput. Phys.* **221** (2007), 761–780.
[35] H. Samet, *Applications of Spatial Data Structures: Computer Graphics, Image Processing and GIS*, Addison-Wesley, Reading, 1990.
[36] R. Scardovelli and S. Zaleski, Direct numerical simulation of free-surface and interfacial flow, *Annu. Rev. Fluid Mech.* **31** (1999), 567–603.
[37] V. A. Solonnikov, Solvability of a problem on the evolution of a viscous incompressible fluid, bounded by a free surface, on a finite time interval, *Algebra i Analiz* **3** (1991), 222–257.
[38] J. Strain, Tree methods for moving interfaces, *J. Comput. Phys.* **151** (1999), 616–648.

[39] M. Sussman, P. Smereka and S. Osher, A level set approach for computing solutions to incompressible two-phase flow, *J. Comput. Phys.* **114** (1994), 146–159.
[40] S. O. Unverdi and G. Tryggvason, A front-tracking method for viscous, incompressible multi-fluid flows, *J. Comput. Phys.* **100** (1992), 25–37.
[41] J. van Kan, A second-order accurate pressure-correction scheme for viscous incompressible flow, *SIAM J. Sci. Stat. Comput.* **7** (1986), no. 3, 870–891.

Received July 11, 2014; revised August 18, 2014; accepted August 26, 2014.