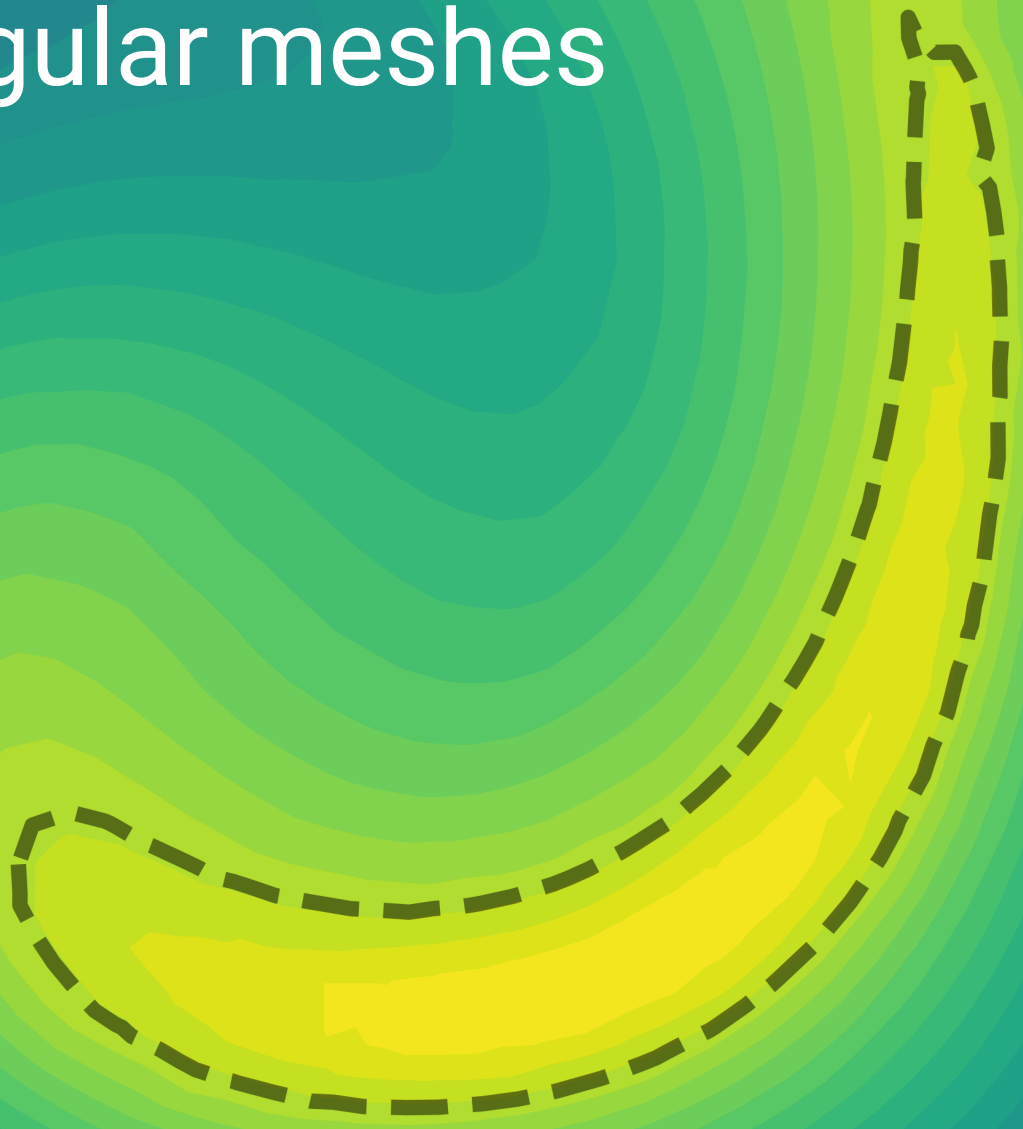
A volume-conserving interface-correction level-set method on unstructured triangular meshes

A. Kerst

Technische Universiteit Delft



TU Delft

A volume-conserving interface-correction level-set method on unstructured triangular meshes

by

A. Kerst

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Thursday November 5, 2020 at 9:00 AM.

Student number:	4482441	
Project duration:	November 18, 2019 – November 5, 2020	
Thesis committee:	Dr. ir. D. den Ouden-van der Horst,	TU Delft, supervisor
	Prof. dr. ir. C. Vuik,	TU Delft
	Dr. M. Möller,	TU Delft
	Dr. B. J. Meulenbroek,	TU Delft

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft

Abstract

The level-set (LS) method uses a signed-distance function to capture the interface in two-phase flows. Geometrical properties can be easily obtained, and merging and splitting of the interface are handled automatically by the LS method. However, it is not inherently volume conserving. Several methods found in literature that aim to solve this problem are discussed in this report, including the interface-correction level-set (ICLS) method. The ICLS method uses an additional advection step with a correction-velocity field to restore global volume loss/gain. We present the volume-of-fluid-based local interface-correction level-set (VOF-LICLS) method. This an extension to the ICLS method that aims to restore volume locally. This is achieved by coupling the ICLS method with the VoF method. In each time step we evolve both the level-set function and the volume fraction function. The VoF advection is performed using a Lagrangian-Eulerian method on a dual mesh. From the advected LS field volume fractions are constructed and compared to the advected VoF field. This allows us to use local volume fluxes for the velocity field, instead of a global volume flux. The construction of the correction-velocity is performed with the use of an analytic equation and the local volume fluxes. The novel volume correction procedure is then executed iteratively. The level-set equation is discretized in space with a finite element approach. Instabilities occur for the standard Galerkin approach for pure advection equation, so the SUPG method is used in order to obtain stable solutions. The performance of the developed method in this thesis is compared with LS and ICLS for three different test cases. We report a significant improvement for local volume conservation, and we observe that VOF-LICLS yields more accurate interface positions.

Contents

1 Introduction

1.1. Simulation of two-phase flows

Two-phase flow, which is a particular case of multiphase flow, consists of two media with possible different properties, where the media can have different phases [17]. Multiphase flow simulations are fundamental tools in many industrial applications and natural processes. Examples are rain drops in the air, free surface flows in the ocean, the dispersion of two immiscible fluids into each other to create emulsions, liquid phase sintering and inkjet printing [9, 22]. Numerical simulations of two-phase flow are difficult and far more challenging than single phase flow. Besides demanding volume conservation, accurately modelling the interface, such that it remains smooth and sharp, is also very important [10]. An accurate interface is essential, especially when the normal vector and curvature are utilized for computing the surface tension [17]. In this thesis we focus on locating the interface in an immiscible and incompressible flow.

In the last three decades several methods have been developed for solving the two-phase flow. Each method has its advantages and disadvantages. While some methods suffer from poor volume conservation properties, other methods suffer from the inability to handle complex domains with unstructured meshes. In general, the available methods can be categorized in two classes: interface-tracking methods and interface-capturing methods. Over the years steady improvements have been made by proposing various variations to the existing methods.

1.2. Interface-tracking and interface-capturing methods

The interface separating the two fluids must be either tracked or captured in the same time step as the flow field evolution [22].

Interface-tracking methods are Lagrangian, that is, the interface is explicitly represented by the mesh. Whenever the interface moves or deforms, the interface has to be rebuild [12]. These methods are very accurate and efficient for moving interfaces with small deformation, but they do not handle disconnecting and reconnecting of the interface well [14]. Interface-capturing methods are Eulerian, where the interface is described by an implicit function on a fixed mesh [14]. Although these methods are robust and have a wide range of applicability, they usually require a higher mesh resolution.

The interface-tracking techniques either employ a deforming mesh that fits to the interface, such as arbitrary Lagrangian-Eulerian (ALE) methods, or explicitly track the interface, which is done in marker-and-cell (MAC) methods. For interface-capturing techniques an auxiliary function is needed. Examples are volume-of-fluid (VoF) methods, where a volume fraction function is used, and level-set (LS) methods, where a signed-distance function is used. Several methods and variations to existing methods have been introduced for solving the two-phase flow. Approaches where the reconstruction techniques rely on the Cartesian control volumes are not able to handle geometrically complex domains. On the other hand, approaches that are applicable for triangular control volumes suffer from volume loss and/or fragility [17] due to their complexity.

Generally, the following four approaches are used for modelling the two-phase flow: front-tracking methods, marker-and-cell methods, level-set method and volume-of-fluid method. Each of these methods have there advantages and disadvantages based on volume conservation, ease of implementation and ability for handling complex geometries. Only the VoF and LS method can be used when handling interface movement

without any geometrical restrictions [17]. All other proposed methods are derived from one or a combination of the mentioned methods. These four approaches [10] are discussed in Sections 1.2.1-1.2.4.

1.2.1. Front-tracking method

Front-tracking methods are based on the movement of the surface $\Gamma(t)$. Suppose that the velocity $\mathbf{u}(x(t),t)$ is known for every point $x(t) \in \Gamma(t)$. Then we get

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{x}(t) = \mathbf{u}(\mathbf{x}(t),t). \tag{1.1}$$

This is known as the the Lagrangian formulation of the interface evolution equation [12]. Front-tracking methods can be categorized into Lagrangian and Eulerian methods. For Lagrangian methods, the mesh is constructed such that one of the boundaries corresponds to the interface. This means that the interface needs to be rebuild whenever it moves or deforms. Eulerian front-tracking methods utilize an additional data structure for describing the interface.

Numerical solutions of front-tracking methods can be unstable and may need to be stabilized by smoothing the corners of the interface. Another disadvantage is its inability to handle merging and splitting of two interface parts automatically and it is complicated to deal with these topological changes. The advantage of the front-tracking method is that the interface is sharp and explicitly given.

1.2.2. Marker-and-cell method

Marker-and-cell (MAC) methods or marker particle methods utilize marker particles. In contrast to front-tracking methods, this method marks the volume occupied by a phase rather than the interface itself [12]. The new particle location is computed by moving with the fluid velocity according to the Lagrangian formulation, Eq. (1.1). A cell containing particles while having at least one neighbour cell without any particles, is called an interface cell. The distribution of particles in interface cells is used to construct the interface.

The MAC method is able to handle merging and splitting of interface parts easily. A drawback of this method is that a large amount of the particles is needed to give an accurate approximation of the interface. When only using few particles, reconstruction of the interface is sensitive to small errors [12], resulting in a blurred interface. In order to obtain a sharp interface, far more particles than cells are needed. Furthermore, a balanced distribution of particles is required in the interface cells. Therefore, adding and removing of particles is necessary during this process, which leads to a large computational cost.

1.2.3. Level-set method

In the level-set method, the interface is captured by a signed-distance function. This function is advected according to the advection equation. During simulation it is advantageous to maintain the level-set function as a signed-distance function by applying reinitialization, however, this is computationally expensive and can be difficult to implement [14].

The level-set method handles merging and breaking of the interface automatically and geometrical information can be easily calculated. A drawback of this approach is poor volume conservation. This method is discussed in more detail in Section 2.

1.2.4. Volume-of-fluid method

The volume-of-fluid (VoF) method uses volume fractions indicating the fractional volume of a certain fluid [21]. In each time step, the volume fractions are advected according to the advection equation. The VoF method conserves volume really well, but the interface has to be reconstructed in each time step due to the discontinuity of the method. This reconstruction procedure is computationally expensive, making it a disadvantage of the method. Furthermore, it is difficult to compute geometric information. More information is given in Section 4.1.

1.2.5. Improvements to the level-set method

Several techniques can be used to solve the level-set equation, Eq. (2.2), in space and time. However, numerical methods may become unstable due to steep or flat gradients of the level set function. One solution to this problem is reinitialization. But the major drawbacks of the reinitialization process is the difficulty of maintaining the original interface location, which will often lead to volume loss/gain [14]. Furthermore, the level-set method is not inherently volume-conserving, which is shown in [10]. So even when the reinitialization procedure has no volume loss/gain, volume conservation is still not guaranteed.

Various approaches have been proposed to make the level-set method stable and volume conserving. These techniques can be divided into four methodologies [6]:

1. Improving LS discretization and reinitialization; by using a higher order scheme, minimizing displacement during reinitialization or refining the mesh near the interface, numerical errors are reduced. However, the level-set method remains inherently non-conservative [6]. In [16], two alternative FEM implementations are discussed.

2. Coupling the LS method with conservative methods, such as VoF or Lagrangian particles; by combining two methods volume can be conserved exactly. However, these hybrid methods can be very complex to implement.

3. Adding a volume constraint in the level-set (LS) or Navier-Stokes (NS) formulation or introducing a volume/mass correction procedure.

4. Modifying the LS definition, such as the (improved) conservative level-set method [24] or the assumed-gradient level-set method [15].

In this thesis we will focus on the second and third methodology; coupling with a conservative method and introducing a volume correction procedure.

1.3. Goal of this project

The goal for this project is to develop a Finite Element (FE) based method for solving the level-set equation on unstructured triangular control volumes. Ideally, the developed method should conserve volume exactly, give a continuous description of the interface and track the interface accurately.

The level-set handles merging and splitting of the interface automatically, and have shown to be a good approach for capturing the interface for two-phase flow simulations in literature. However, the level-set method is not inherently volume-conserving, so our goal is to add an extension to the method which preserves volume. We want to maintain the continuity of the level-set method, such that geometrical information can be easily calculated. Furthermore, at each time step we want a continuous description of the interface without having to reconstruct this. For geometrically flexibility we choose a FE-based method on unstructured triangular grids, since this is able to handle geometrically complex domains.

While discontinuous methods on unstructured triangular grids and continuous methods on structured rectangular have been developed, accurate continuous volume-conserving methods on unstructured triangular grids are not yet available to the authors knowledge.

1.4. Outline of the thesis

In [6] the interface-correcting level-set (ICLS) method is proposed, which performs an additional advection achieving global volume conservation. In this thesis a novel coupling between ICLS and VoF is presented: the volume-of-fluid-based local interface-correction level-set (VOF-LICLS) method. A continuous Galerkin approach is used for discretization on a unstructured triangular mesh. The original formulation and derivation of the ICLS method and the adaptation is discussed. Furthermore, the performance of VOF-LICLS is compared with LS and ICLS for different test cases.

The outline of this thesis is as follows. In Section 2 the level-set method is discussed, and the discretization of the LS field is presented in Section 3. An overview of various methods is given in Section 4. Sections 1-4 were part of the literature study [10]. Developed methods and obtained results during this thesis can be found in Sections 5-8. An extension to the ICLS method with a coupling with VoF is introduced in Section 5, where its performance is tested against ICLS. An improved local version (VOF-LICLS) is proposed in Section 6. In Section 7, results of the performance compared to ICLS and LS and a error analysis is given, which is followed by a conclusion and discussion in Section 8.

2

Classical level-set methodology

In the level-set method, the interface is captured by a signed-distance function ϕ defined by

$$\phi(\mathbf{x},t)=\begin{cases}\min\limits_{y\in\Gamma(t)}\|\mathbf{x}-\mathbf{y}\| & \text{if } \mathbf{x}\in\Omega_1(t),\\ 0 & \text{if } \mathbf{x}\in\Gamma(t),\\ -\min\limits_{y\in\Gamma(t)}\|\mathbf{x}-\mathbf{y}\| & \text{if } \mathbf{x}\in\Omega_2(t),\end{cases} \tag{2.1}$$

where $\Gamma(t)$ is the interface between the two phases $\Omega_1(t)$ and $\Omega_2(t)$ [4]. This function is advected according to the advection equation

$$\frac{\partial\phi}{\partial t}+\mathbf{u}\cdot\nabla\phi=0, \tag{2.2}$$

with $\mathbf{u}$ the velocity of the flow. The derivation is given in Section A.1.

The signed-distance function satisfies $\|\nabla\phi\|=1$ where $\nabla\phi$ is defined, which is known as the Eikonal equation. This property can be used to redistance the function, which will be discussed in Section 2.1.

2.1. Reinitialization of level-set function

In general, the level-set function ϕ does not remain a signed-distance function when ϕ is advected, which means that too steep and/or too flat gradients can occur [10]. This results in errors when derivatives are taken for computing the interface normal vector and instability problems [14]. Hence it is desirable to maintain ϕ as a signed-distance function. Therefore reinitialization is needed to modify the level-set function [22]. Reinitialization of the level-set function is not always necessary, and is only required for stability reasons which causes ϕ to diverge from the signed-distance function, or when a constant interface thickness is needed for smoothing discontinuities. If this is not the case, one should avoid reinitialization since it is computationally expensive and can be difficult to implement [14]. There are several techniques for reinitializing ϕ as a signed-distance function [14], such as the fast marching method and the fast sweeping method. Both methods solve the Eikonal equation $\|\nabla\phi\|=1$ and retain the location of the interface. Another method solves a first-order partial-differential equation, known as the time-dependent Eikonal equation, in pseudo-time, given by [6]

$$\frac{\partial\phi}{\partial\tau}+S(\phi_0)\left(\|\nabla\phi\|-1\right)=0, \tag{2.3}$$

where τ is a pseudo-time, ϕ_0 is the initial level-set field, and $S(\phi_0)$ is a smoothed sign function, defined as

$$S(\phi_0)=\begin{cases}-1 & \text{if } \phi_0<-\Delta x,\\ 1 & \text{if } \phi_0>\Delta x,\\ \dfrac{\phi_0}{\sqrt{\phi_0^2+\Delta x^2}} & \text{otherwise.}\end{cases} \tag{2.4}$$

Equation (2.3) can be rewritten in the following form

$$\begin{aligned} &\frac{\partial \phi}{\partial \tau} + S(\phi_0)\left(\|\nabla\phi\| - 1\right) = 0, \\ \implies &\frac{\partial \phi}{\partial \tau} + S(\phi_0)\|\nabla\phi\| = S(\phi_0), \\ \implies &\frac{\partial \phi}{\partial \tau} + \left(\frac{S(\phi_0)\nabla\phi}{\|\nabla\phi\|}\right)\cdot\nabla\phi = S(\phi_0), \\ \implies &\frac{\partial \phi}{\partial \tau} + \mathbf{w}\cdot\nabla\phi = S(\phi_0), \end{aligned} \tag{2.5}$$

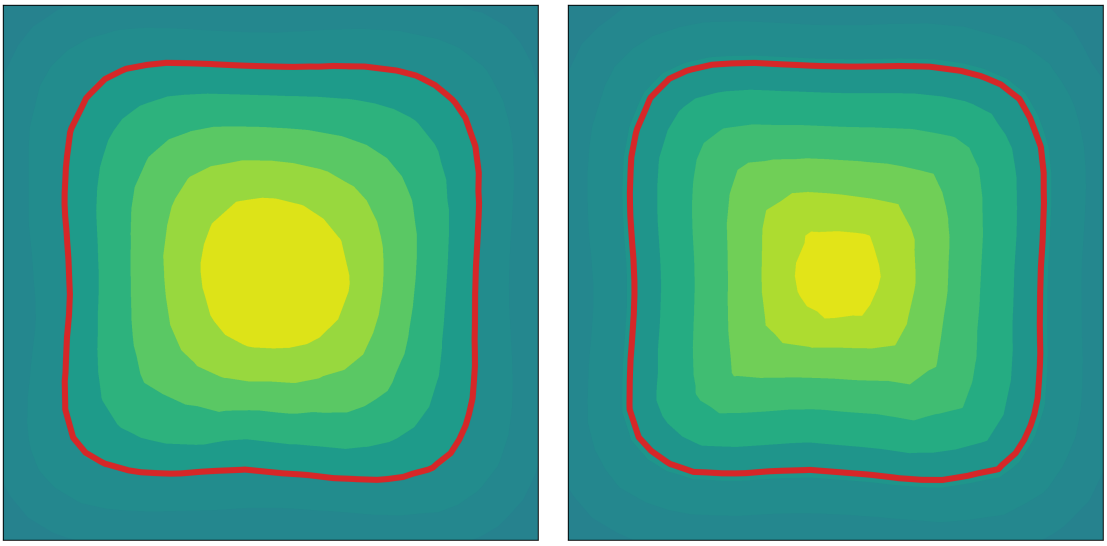
with

$$\mathbf{w} = \frac{S(\phi_0)\nabla\phi}{\|\nabla\phi\|}. \tag{2.6}$$

Equation (2.5) is the advection equation with a non-zero RHS. By experimentation we find that advection with the standard Galerkin method will lead to instabilities, therefore we choose to discretize Eq. (2.5) with the SUPG method, similar to the level-set advection discretization. The discretization for this equation can be found in Appendix B.2.

Solving Eq. (2.5) is a more popular choice compared to FMM, since it is easy to parallelize and higher order schemes can be used [6]. However, numerical diffusion when solving the equation can result in a shifted interface, which can lead to volume loss/gain. In order to restore this volume loss/gain, the reinitialization procedure must be applied after level-set advection and before volume correction. However, one could also pin the interface by using Dirichlet boundary conditions at the zero level-set. In this thesis we will use the former approach; applying volume correction after reinitialization.

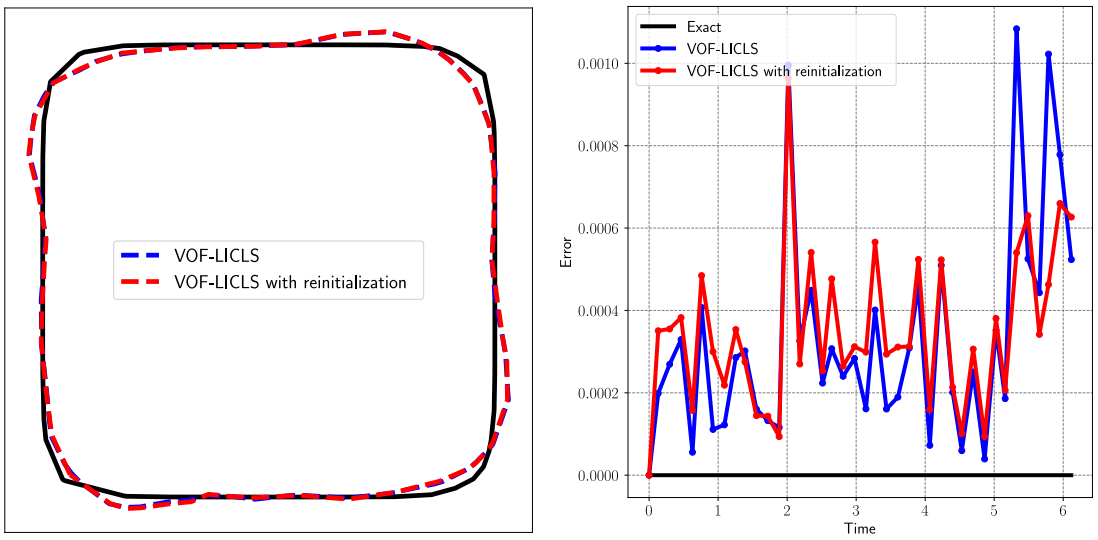
According to [6], reinitialization was required for one to two iterations per ten to a hundred time steps for their applications. Similar observations were made for methods considered in this report. In Figure 2.1 the result of a square in a rotating flow at $t = 2\pi$ for the LS method is displayed, where reinitialization is applied after every 25 time steps with 4 iterations. The iso-contours of the simulation with reinitialization corresponds more to a square than the simulation without reinitialization.



(a) Without reinitialization (b) With reinitialization

Figure 2.1: Results at $t = 2\pi$ of a square in a rotating flow without and with reinitialization.

We also simulated a droplet in a rotating flow for $t = 2\pi$ with VOF-LICLS, the developed method in this thesis which will be discussed in Section 6. The result at $t = 2\pi$ and the mean error at the interface from Eq. (5.21) are displayed in Figure 2.2. The mean error is plotted for some iterations, which are uniformly distributed across the entire time span. This is done for every graph in this report. We find that reinitialization does not improve the performance significantly. From this we can conclude that reinitialization is not



(a) Result at $t = 2\pi$.

(b) Mean error $\overline{E_\Gamma}$.

Figure 2.2: Results of droplet in rotating flow for VOF-LICLS with and without reinitialization.

necessary for advecting the LS field with a SUPG approach combined with the volume correction procedure. Therefore we choose to omit reinitialization.

2.2. Properties of level-set method

The advantage of the level-set method is its ability to handle merging and breaking of the interface automatically, without having to explicitly reconstruct the interface. Furthermore, geometrical information can be easily calculated, since ϕ is smooth near the interface

$$\mathbf{n} = \frac{\nabla\phi}{|\nabla\phi|}, \quad \kappa = \nabla \cdot \mathbf{n},$$

where $\mathbf{n}$ is the unit outward normal and κ the curvature.

The disadvantage is poor volume conservation, which can come from two aspects [24]. Firstly, it can come from numerical dissipation from discretization of the level-set equation, since a smooth distance function is advected rather than a conserved physical quantity, such as mass or volume. Therefore the level-set method has no volume conservation by design. And secondly, it can be caused by volume gain/loss from the reinitialization process, where it can occur that the interface shifts, because the signed-distance function does not provide an accurate approximation of the exact location [24].

3

Discretization of the level-set equation

3.1. Spatial discretization

We wish to discretize the level-set method with a finite element approach for its geometrical flexibility. However, we will not use the standard continuous Galerkin approach, since it is a well-known fact that naive finite-element implementations are unsatisfactory [2, 7, 16], which is shown in [10]. Standard finite-element discretizations suffer from instability issues, and in order to stabilize the numerical solution a certain amount of numerical viscosity has to be added [16]. While adding artificial diffusion solves this problem, it must be taken into account that this also affects the accuracy of the solution, which may result in volume loss.

3.1.1. Continuous Galerkin (CG) method

Let $\mathscr{T}_h$ be a triangulation of the domain Ω, and let $\mathscr{P}_k$ be the space of polynomials of degree $\leq k$. The approximation of ϕ is denoted by ϕ_h. The space of continuous piecewise polynomial functions [12] is then defined by

$$\begin{aligned} V_h^k &:= \{v_h \in \mathscr{C}(\bar{\Omega}) : v_h|_K \in \mathscr{P}_k \quad \forall K \in \mathscr{T}_h\}, \\ V_{h,g}^k &:= \{v_h \in V_h^k : v_h|_{\partial\Omega_{\text{in}}} = g_D\}. \end{aligned} \tag{3.1}$$

for $k \geq 1$. Here, $\partial\Omega_{in}$ denotes the inlet boundary, that is, the part of the boundary where the flow enters the domain.

The weak form of the standard Galerkin method is as follows. For all $t \in [0, T]$ find $\phi_h(t) \in V_h^k$ such that

$$\begin{cases} \displaystyle\int_\Omega \frac{\partial \phi_h}{\partial t} v_h \,\mathrm{d}\Omega + \int_\Omega \left(\mathbf{u} \cdot \nabla \phi_h\right) v_h \,\mathrm{d}\Omega = 0, \quad \forall v_h \in V_h^k, \\ \phi_h(0) = \phi_{0,h}. \end{cases} \tag{3.2}$$

However, this formulation will result in instabilities near the inlet boundary. Therefore we need to impose Dirichlet boundary conditions on the inlet boundary. The weak form of the standard Galerkin method for Dirichlet boundary conditions is given by: for all $t \in [0, T]$ find $\phi_h(t) \in V_{h,g}^k$ such that

$$\begin{cases} \displaystyle\int_\Omega \frac{\partial \phi_h}{\partial t} v_h \,\mathrm{d}\Omega + \int_\Omega \left(\mathbf{u} \cdot \nabla \phi_h\right) v_h \,\mathrm{d}\Omega = 0, \quad \forall v_h \in V_{h,0}^k, \\ \phi_h(0) = \phi_{0,h}. \end{cases} \tag{3.3}$$

The discretization can be found in Appendices A.2.1 and A.2.2. Results from [10] show that the standard Galerkin approach leads to instabilities, from which we can conclude that this approach is not appropriate for this application. Therefore we consider the Streamline upwind Petrov-Galerkin method in the next section.

3.1.2. Streamline upwind Petrov-Galerkin (SUPG) method

Let us define the following function spaces

$$\begin{aligned} W &:= \{w \in L_2(\Omega) : \mathbf{u}\cdot\nabla w \in L_2(\Omega)\},\\ W_0 &:= \{w \in W : w|_{\partial\Omega_{in}} = 0\},\\ W_g &:= \{w \in W : w|_{\partial\Omega_{in}} = g_D\}. \end{aligned} \tag{3.4}$$

Note that V_h^k is a subspace of W.

The streamline upwind Petrov-Galerkin (SUPG) stabilizes the standard Galerkin method by adding diffusion, which smooths the solution. Instead of simply adding a diffusion term to the level-set equation, diffusion is added in the streamline direction. In this way, less diffusion is produced. The diffusion is incorporated into the equation in such a way, that the exact solution is also a solution to the stabilized problem [12].

Note that $\mathbf{u}\cdot\nabla v \in L_2(\Omega)$ for any function $v \in W$, which means that

$$\int_\Omega \frac{\partial\phi}{\partial t}(\mathbf{u}\cdot\nabla v)\,\mathrm{d}\Omega + \int_\Omega \left(\mathbf{u}\cdot\nabla\phi\right)(\mathbf{u}\cdot\nabla v)\,\mathrm{d}\Omega = 0 \tag{3.5}$$

holds for the exact solution in the weak sense and for any $v \in W$. This is added δ times to the standard variational formulation, resulting in

$$\int_\Omega \frac{\partial\phi}{\partial t}(v+\delta\mathbf{u}\cdot\nabla v)\,\mathrm{d}\Omega + \int_\Omega \left(\mathbf{u}\cdot\nabla\phi\right)(v+\delta\mathbf{u}\cdot\nabla v)\,\mathrm{d}\Omega = 0, \tag{3.6}$$

with δ a positive constant.

When we added the diffusion term, we changed the test space. This means that the trial and test space are not the same any more, therefore this method is a Petrov-Galerkin method [12]. The SUPG test function is defined by

$$w_h(v_h) := v_h + \delta_K\mathbf{u}\cdot\nabla v_h, \tag{3.7}$$

where $v_h \in V_h^k$. The parameter δ_K is non-negative and defined element-wise on every $K \in \mathcal{T}_h$. The SUPG test space is defined by

$$V_h^{\mathrm{SUPG}} := \{w_h(v_h) : v_h \in V_h^k\}. \tag{3.8}$$

Substituting this into the standard variational equations and summing over all elements $K \in \mathcal{T}_h$ gives the desired problem. For all $t \in [0,T]$ find $\phi_h(t) \in V_{h,g}^k$ such that

$$\begin{cases} \displaystyle\sum_{K\in\mathcal{T}_h}\int_{\Omega_K} \frac{\partial\phi_h}{\partial t}\,w_h\,\mathrm{d}\Omega + \int_{\Omega_K}\left(\mathbf{u}\cdot\nabla\phi_h\right)w_h\,\mathrm{d}\Omega = 0, \quad \forall\, w_h \in V_{h,0}^{\mathrm{SUPG}},\\ \phi_h(0) = \phi_{0,h}. \end{cases} \tag{3.9}$$

A difficulty of the SUPG method is to choose an appropriate value of δ_K. Usually, the parameter is chosen to be proportional to the grid size [12]. For this problem δ_K is taken as

$$\delta_K = s\frac{h_K}{\max\{\epsilon, \|\mathbf{u}\|_{\infty,K}\}}, \tag{3.10}$$

with $\epsilon > 0$, and $s = \mathcal{O}(1)$ a scaling factor. The discretization is given in Appendix A.2.3.

Another stabilizing finite-element approach is the Galerkin Least Squares (GLS) finite-element method. However, the variational formulation is identical to the SUPG method for the hyperbolic case, or for piecewise linear elements in the general case [8]. Therefore we will not consider the GLS method.

3.1.3. Other spatial discretization methods

Besides adding diffusion in the streamline direction, other suitable stabilization method can be found in literature. The subgrid edge stabilization method [2, 16] adds a term to the standard Galerkin formulation, which penalizes the jump of the gradient over internal element boundaries. This method utilizes an additional finer triangulation with half the mesh size. Only the jump of the gradient over the subtriangles is penalized [10],

which leads to good volume conservation properties, since the stabilization is only present on the finest scale. Therefore the method will have the same conservation properties as the standard Galerkin method.

Another discretization approach is the stabilized finite element method without reinitialization, proposed in [20]. This method adds a term depended on the local residual of the Eikonal equation to the SUPG formulation of the level-set equation. This improves the interface resolution without having to reinitialize the level-set function. Furthermore, it is claimed that with the penalty term simpler and more efficient numerical schemes can be used. It has also been shown that this method enhances the numerical behaviour of the SUPG method.

Another common method is discontinuous Galerkin (DG) method [5], which utilizes ideas and techniques from both the Finite Volume method (FVM) and the Finite Element Method (FEM), resulting in a numerical scheme with advantages from FVM and FEM. Interior and boundary penalty are introduced to mimic the continuity of the approximate solution in a weaker sense. The interior penalty is required in standard conforming finite-element methods and the boundary penalty is employed for the Dirichlet boundary condition. The Runge-Kutta discontinuous Galerkin (RK-DG) method is a popular numerical technique thanks to the fact that it is accurate, compact, robust and it can handle complex geometries. Furthermore, discontinuous Galerkin finite-element approximations requires much less artificial viscosity than stabilization techniques applied to continuous Galerkin approximations [16]. Despite its good properties, we will not use this scheme since we want to maintain the continuity of the level-set function.

3.2. Temporal discretization

The semi-discrete system obtained from the SUPG method is a system of ordinary differential equations of the form

$$M(\mathbf{u})\,\frac{\mathrm{d}\boldsymbol{\phi}_h(t)}{\mathrm{d}t} = S(\mathbf{u})\boldsymbol{\phi}_h(t) + \mathbf{f}(t). \tag{3.11}$$

The fully-discrete system is acquired by discretization in time. For this project we will be using the Crank-Nicolson scheme for temporal discretization. While there are many time stepping schemes for this application, we will not consider other schemes, since the focus lies on developing a volume conserving method with an appropriate volume correction procedure. By choosing an implicit method we will obtain a stable result and besides that, we do not have to worry about the CFL-condition.

For the next part we define [12]

$$\mathbf{m} : W \times L_2(\Omega) \to \mathbb{R}, \quad \mathbf{m}(w,v) := (\mathbf{u}\cdot\nabla w, v)_{L_2(\Omega)},$$
$$\mathbf{s} : \mathscr{C}^1([0,T]; L_2(\Omega)) \times L_2(\Omega) \to \mathbb{R}, \quad \mathbf{s}(\eta, v) := (\tfrac{\mathrm{d}\eta}{\mathrm{d}t}, v)_{L_2(\Omega)}.$$

In stead of the matrix-vector formulation, we switch to a more suitable representation. For all $t \in [0,T]$ find $\eta_h(t) \in V_h^k$ such that

$$\begin{cases} \mathbf{m}(\frac{\partial \eta_h}{\partial t}, w_h) = \mathbf{s}(\eta_h(t), w_h) + (f(t), w_h), & \forall\, w_h \in V_h^{\mathrm{SUPG}}, \\ \eta_h(0) = \eta_{0,h}. \end{cases} \tag{3.12}$$

The Crank-Nicolson scheme for this system gives the fully discrete system: with $\eta_h^0 = \eta_h$, find for all time steps $n = 1, \dots N$ find $\eta_h(t) \in V_h^k$ such that

$$\left(\frac{\eta_h^n - \eta_h^{n-1}}{\Delta t}, w_h\right) = \mathbf{s}\left(\frac{1}{2}(\eta_h^n + \eta_h^{n-1}), w_h\right) + \left(\frac{1}{2}(f(t^n) + f(t^{n-1}))\right), \quad \forall\, w_h \in V_h^{\mathrm{SUPG}}. \tag{3.13}$$

For a constant velocity this gives us the fully discrete system in matrix-vector representation

$$\begin{aligned} M\frac{\boldsymbol{\phi}_h^{n+1} - \boldsymbol{\phi}_h^n}{\Delta t} &= \frac{1}{2}S^n(\boldsymbol{\phi}_h^n + \boldsymbol{\phi}_h^{n+1}) + \mathbf{f}^n, \\ \implies \boldsymbol{\phi}_h^{n+1} &= (M - \tfrac{1}{2}\Delta t S^n)^{-1}\left[(M + \tfrac{1}{2}\Delta t S^n)\boldsymbol{\phi}_h^n + \Delta t \mathbf{f}^n\right]. \end{aligned} \tag{3.14}$$

For time-dependent velocities the above formulation does not hold, and a generalized version needs to be used [12]. This version is as follows: with ϕ^0 and the velocity field given for the actual time step, compute $\tilde{\boldsymbol{\phi}}^0$

such that $M\tilde{\boldsymbol{\phi}}^0 = S\boldsymbol{\phi}^0 + \mathbf{f}^0$. For all time steps $n = 1, \ldots, N$ compute $\boldsymbol{\phi}_h^n$ and $\tilde{\boldsymbol{\phi}}_h^n$ such that

$$\begin{aligned} \frac{1}{\Delta t} M(\mathbf{u}^n)\boldsymbol{\phi}_h^n - \frac{1}{2} S(\mathbf{u}^n)\boldsymbol{\phi}_h^n + \frac{1}{2}\mathbf{f}^n &= \frac{1}{\Delta t} M(\mathbf{u}^n)\boldsymbol{\phi}_h^{n-1} + \frac{1}{2} M(\mathbf{u}^n)\tilde{\boldsymbol{\phi}}_h^{n-1}, \\ \tilde{\boldsymbol{\phi}}_h^n &= \frac{\boldsymbol{\phi}_h^n - \boldsymbol{\phi}_h^{n-1}}{\Delta t} - \frac{1}{2}\tilde{\boldsymbol{\phi}}_h^{n-1}. \end{aligned} \tag{3.15}$$

4

Overview of other methods

In this section a few interface-capturing methods are highlighted.

4.1. Volume-of-fluid method

The volume-of-fluid (VoF) method uses a marker function, denoted by ψ, indicating the fractional volume of a certain fluid [21]. The definition is given by

$$\psi_k(t) = \frac{1}{|\Omega_k|}\int_{\Omega_k} c(\mathbf{x},t)\,\mathrm{d}\Omega, \quad |\Omega_k| = \int_{\Omega_k} \mathrm{d}\Omega, \tag{4.1}$$

where the control volume is denoted by Ω_k, and the colour function $c(\mathbf{x},t) : \mathbb{R} \rightarrow \{0,1\}$ is defined by

$$c(\mathbf{x},t) = \begin{cases} 0 & \text{if phase 0 is present,} \\ 1 & \text{if phase 1 is present.} \end{cases}$$

This volume fraction function gives the ratio of phase 1 to the total ratio, which means ψ is 0 and 1 when the cell only consists of phase 0 and 1, respectively, and lies between 0 and 1 when both phases are present. In each time step, the volume fraction is advected according to

$$\frac{\partial \psi}{\partial t} + \mathbf{u}\cdot\nabla\psi = 0. \tag{4.2}$$

4.1.1. Interface reconstruction

Since the volume-of-function ψ is discontinuous across the cells, the exact interface location is unknown and has to be approximated in each cell, where the volume fraction function takes values between 0 and 1 [17]. This procedure is called 'interface reconstruction'. Common methods used for interface reconstruction are the Simple Line Interface Calculation (SLIC) and the Piecewise Linear Interface Calculation (PLIC). This is computationally expensive, which makes it a disadvantage for VoF [24]. Furthermore, the reconstructed interface is discontinuous between cells [13].

4.1.2. Properties of volume-of-fluid method

The VoF method conserves volume really well except for small over and under shoots. Besides the computationally intensive interface reconstruction, other drawbacks of this method are ejection of non-physical fluid, parasitic currents and the inability to compute geometric information, such as unit normal vector and curvature, accurately [13]. This is caused by the step-like behaviour of ψ [21]. Numerical diffusion arises by application of the numerical scheme for the advection equation. This results in numerical errors which are replaced by inaccurate volume motion due to volume conservation.

A common approach for hybrid methods is to combine the level-set method with the volume-of-fluid method. In Sections 4.2-4.5 a couple of methods are discussed.

4.1.3. Discretization of the volume-of-fluid method

Solving Eq. (4.2) is not easy, since standard numerical schemes can easily diffuse the interface due to the discontinuity of function ψ. A remedy to overcome this problem is to first reconstruct the interface before advecting ψ [22]. Several advection methods have been developed which utilize the reconstructed interface. For structured rectangular meshes it would be an obvious choice to discretize the VoF method with a Finite Volume approach. However, such schemes for rectangular elements cannot be easily adapted for triangular elements [12, 18, 22]. The implementation for unstructured triangular grids is challenging due to the complexity of computing edge fluxes and corner fluxes. Instead, in [17, 18, 22], the volume fraction advection is performed using a Lagrangian-Eulerian method, which is applicable for both structured rectangular grids and unstructured triangular grids. In [11], the incremental remapping approach is employed. In the next section the Lagrangian-Eulerian method is discussed, which will be the method used for discretization in this thesis.

4.1.4. Lagrangian-Eulerian advection method

The Lagrangian-Eulerian advection scheme consists of three stages: Lagrangian projection, reconstruction and remapping [22]. In the first stage the volume fraction is projected, which is equivalent to the solving Eq. (4.2) using a Lagrangian approach. The volume fraction is solely transported with the flow, so for an element K at time t^n we have

$$\tilde{\psi}_K^{n+1} = \psi_K^n, \quad \forall K \in \mathcal{T}_h, \tag{4.3}$$

where $\tilde{\psi}_K^{n+1}$ is the volume fraction after the Lagrangian projection.

If we assume that the velocity is piecewise linear over every finite element, then it is sufficient to only find the Lagrangian position for every vertex of the element. So for a vertex $\mathbf{x}$ the new position is found by solving

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{u}. \tag{4.4}$$

A second-order Runge-Kutta (RK) scheme is used to obtain the Lagrangian points

$$\begin{aligned} \tilde{\mathbf{x}}_i^{n+\frac{1}{2}} &= \mathbf{x}_i^n + \frac{\Delta t}{2}\mathbf{u}(\mathbf{x}_i^n, t^n), \quad \text{for } i = 1,\ldots,n, \\ \tilde{\mathbf{x}}_i^{n+1} &= \mathbf{x}_i^n + \Delta t\,\mathbf{u}(\tilde{\mathbf{x}}_i^{n+\frac{1}{2}}, t^{n+\frac{1}{2}}), \quad \text{for } i = 1,\ldots,n, \end{aligned} \tag{4.5}$$

with n the total grid nodes. The projected grid is called the Lagrangian grid. Note that in the second RK step the velocity $\mathbf{u}(\tilde{\mathbf{x}}_i^{n+\frac{1}{2}}, t^{n+\frac{1}{2}})$ on the Lagrangian grid is required, so this needs to be interpolated from the Eulerian grid. The projected element may deform but remains a triangle.

Theoretically, the Lagrangian projection conserves the volume in each element due to the incompressibility of the fluid. However, numerically, this may not be the case. According to [22] the volume of fluid can not be exactly conserved due to: the numerical velocity not being exactly conserved, the velocity being approximated by piecewise linear basis functions, the velocity being interpolated from the Eulerian mesh and the numerical errors from Eq. (4.2). Results from [22] show that the volume loss due to the last three reasons is exceedingly small.

After the Lagrangian projection, the volume fraction in each Lagrangian cell is known. In order to map the volume fraction back to the Eulerian mesh, we need to know which part is occupied with fluid 1 and which part is occupied with fluid 2. Therefore the interface is needed. It is common to reconstruct the interface as a line segment of the form $\mathbf{n}\cdot\mathbf{x} = \alpha$, where $\mathbf{n}$ is the unit normal vector, $\mathbf{x}$ a point on the interface and α a line constant. The unit normal vector is calculated as the gradient of the volume fraction function. This function is discontinuous, so most methods are less than second order accurate or very expensive. An available method for computing the normal vector is the differential least squares (DLS) method, discussed in [18]. The parameter α is determined by enforcing volume conservation, which is usually done iteratively. However, in [22], an analytic method is proposed for computing this line constant.

In the final stage, the new advected volume fraction ψ_K^{n+1} is obtained by mapping the projected elements back to the Eulerian grid, which is achieved by performing polygon-polygon clippings. Note that only cells near the interface need to be clipped, other cells simply remain 0 or 1.

4.2. Adaptive coupled level-set volume-of-fluid (ACLSVOF) method

In [22], the level-set method and the volume-of-fluid method are combined to obtain advantages from both methods. The adaptive coupled level-set volume-of-fluid (ACLSVOF) method is based on the CLSVOF method, introduced in [19]. The level-set function and the volume-of-fluid method are advected using a discontinuous Galerkin and Lagrangian-Eulerian method, respectively. The ACLSVOF method employs the level-set function for calculating the interface normal vector, and the volume-of-fluid function for the line constant in order to maintain mass conservation. Since ϕ is continuous, it is easy and accurate to compute the unit normal, and by the volume-conserving property of ψ, volume can be conserved accurately. In each time step the interface needs to be reconstructed for which [22] developed an analytic piecewise linear interface reconstruction. This method is developed for unstructured triangular grids, where the grid can be adapted in order to resolve complex changes in interface topology.

This method is based on the interaction between the VoF and LS functions for interface reconstruction and the interplay between the Eulerian and Lagrangian mesh for advection, making it a complex method to implement [23]. Furthermore, this method suffers from computationally expensive interface reconstruction.

4.3. Coupled volume-of-fluid and level-set method (VOSET)

In [11], a coupled volume-of-fluid and level-set method (VOSET) is introduced which combines the volume-of-fluid and level-set method on arbitrary polygon meshes. The LS function is geometrically constructed from the VoF function in order to accurately compute the geometric properties and physically sharp interface without discontinuous oscillations [23]. Since ϕ is constructed from ψ, the level-set field does not need to be advected. The new volume fractions are computed with the incremental remapping approach, a method similar to the Lagrangian-Eulerian method. The proposed VOSET method can track the interface accurately and can be applied for complex geometries.

4.4. Mass-conserving level-set (MCLS) method

Unlike the ACLSVOF method, the mass-conserving level-set (MCLS) method [17, 21] is not a combination of two or multiple methods, but uses all available information from the level-set function ϕ, rather than coupling with the VoF method. The VoF function, however, is utilized but without applying the computationally expensive interface reconstruction step. This results in a mass-conserving method, which is easy to implement and gives an explicit interface position.

In [21], an explicit relation between the level-set function ϕ and the VoF function ψ is introduced, which is the following

$$\psi = f(\phi, \nabla\phi), \tag{4.6}$$

where the assumption of piecewise linear interfaces within a computational cell is made. Now, ψ can be easily obtained from ϕ, which simplifies the advection of ψ. For a computational cell Ω_k, the Heaviside function is used for the color function, which becomes $c = H(\phi)$. This results in the following connection

$$\psi_k = \frac{1}{|\Omega_k|}\int_{\Omega_k} H(\phi)\,\mathrm{d}\Omega. \tag{4.7}$$

Because of this relation, the level-set field has mass conserving properties of the volume-of-fluid field and the interface is exactly defined by the level-set field.

As already discussed, a drawback of the level-set method is that conservation of ϕ does not imply conservation of mass, in contrast to the VoF method, where mass is conserved when ψ is conserved [21]. Therefore, MCLS method uses the fractional volume ψ to make corrections to the level-set function ϕ. First ϕ is advected and reinitialized, where the result is denoted by $\phi^{n+1,*}$. Corrections to $\phi^{n+1,*}$ for mass conservation are divided in the following three steps:

1. computation of the VoF function from the level-set function using the Heaviside step function and linearisation: $\psi = f(\phi, \nabla\phi)$;
2. conservatively advection of the VoF function with a finite volume approach, which yields ψ^{n+1};
3. corrections to $\phi^{n+1,*}$ using the inverse function of f and Picard-iterations, such that $f(\phi^{n+1}, \nabla\phi^{n+1}) = \psi^{n+1}$ holds.

The original formulation of the MCLS method in [21] is for Cartesian quadrilateral control volumes, but in [17] an extension has been made for unstructured triangular control volumes for geometrical flexibility. Here, the level-set field is advected with a discontinuous Galerkin (DG) finite element method for space discretization, which has high accuracy near boundaries and the ability to handle arbitrary geometrical complexity [17]. Runge-Kutta is used for time discretization. The advection of the VoF field is done simultaneously. The Lagrangian-Eulerian advection method is used for advection, discussed in Section 4.1.4. However, this is performed without the computationally intensive interface reconstruction procedure. Instead, the interface position from the LS field is used. The advected function is then compared to the constructed VoF function. If the error is large, the level-set function is adapted according to the advected VoF function.

We will shortly discuss the volume correction procedure introduced in [21]. In each cell a modification to ϕ^{n+1} is sought, such that

$$|f(\phi_k^{n+1}, \nabla\phi_k^{n+1}) - \psi_k^{n+1}| \leq \epsilon, \quad \forall k = 1, 2, \ldots,$$

which implies that mass is conserved in that cell. For this procedure the inverse function g of f is used to make the coupling between the mass-conserved ψ^{n+1} and the to be adapted level-set function

$$f(g(\psi, \nabla\phi), \nabla\phi) = \psi.$$

The updated level-set function is obtained by employing Picard-iterations for

$$\phi_k^{n+1,m+1} = g(\psi_k^{n+1}, \nabla\phi_k^{n+1,m}), \quad \forall k = 1, 2, \ldots, \tag{4.8}$$

where m denotes the m-th iteration of the Picard iteration.

In [17] the volume fractions in each cell are corrected in one step using the analytic inverse function of f from Eq. (4.6). The obtained LS field is discontinuous, but since the discontinuous Galerkin scheme is utilized, this does not matter.

4.5. Volume-preserving continuous Galerkin level-set approach

In [4] a volume-preserving method is proposed, where a continuous Galerkin level-set formulation based on linear triangles is coupled with the volume-of-fluid method on star-shaped polygonal finite-volume meshes. Volume preservation during advection is ensured by employing a volume-correction algorithm.

The advantage of this method is that the volume is preserved due to the coupling between the LS and VoF function and the volume correction step. Furthermore, the interface is continuous, since the interface is defined by the LS function and no costly interface construction has to be performed.

This method defines a level-set function $\phi(\mathbf{x}, t)$ on a triangular primal mesh and a volume-of-fluid method $\psi(\mathbf{x}, t)$ on a star-shaped polytopal dual mesh. Both the level-set function and the volume-of-fluid function are advected and after advection the discrete volume-of-fluid functions $\psi_h(\mathbf{x}, t)$ and $\psi_{\phi,h}(\mathbf{x}, t)$ are compared and a volume correction algorithm is applied to adapt the level-set function $\phi(\mathbf{x}, t)$. The volume correction procedure is similar to the algorithm from MCLS [21].

4.6. Interface-correction level-set method

The interface-correction level-set method uses a correction-velocity field in order to achieve global volume conservation. In this section we will discuss the method and the derivation. In [6] an approach is introduced to conserve volume globally by making small corrections near the interface. Because the corrections are done by solving the advection equation, the method is simple and easy to implement in 2D and 3D. Furthermore, this method achieves global volume correction at once, therefore multiple iterations are not necessary.

The domain is divided by the interface Γ into subdomain Ω_1 and Ω_2. The volume of Ω_1 is denoted by V. Furthermore, we have $\phi > 0$ in Ω_1 and $\phi < 0$ in Ω_2. The method uses a correction velocity, which is implicitly defined by

$$\int_\Gamma \mathbf{n} \cdot \mathbf{u}_c \, \mathrm{d}\Gamma = \frac{\delta V}{\delta t}, \tag{4.9}$$

where $\mathbf{n}$ is the outward-pointing unit normal vector from the interface Γ and $\frac{\delta V}{\delta t}$ is the rate of change of V. Assuming that $-\frac{\delta V}{\delta t}$ corresponds to the volume loss over an arbitrary period of time, then volume conservation

can be achieved by advecting ψ with the correction velocity $\mathbf{u}_c$ according to

$$\frac{\partial \phi}{\partial t} + \mathbf{u}_c \cdot \nabla \phi = 0. \tag{4.10}$$

After one time step volume loss over time period δt is restored.

For this method the correction velocity needs to be constructed, which is done as follows. The rate of change of $\mathbf{u}_c$ is given by

$$\frac{\mathrm{d}\mathbf{u}_c}{\mathrm{d}t} = -f_s \nabla p_c,$$

where f_s is a speed function and p_c an auxiliary pressure. This can also be written as

$$\mathbf{u}_c = -\int_0^{\Delta t} f_s \nabla p_c \, dt, \tag{4.11}$$

so at $t = 0$ the correction velocity is zero.

Substituting Eq. (4.11) in Eq. (4.9) results in

$$\begin{aligned} \int_\Gamma \mathbf{n} \cdot \left(-\int_0^{\Delta t} f_s \nabla p_c \, dt \right) \mathrm{d}\Gamma &= \frac{\delta V}{\delta t}, \\ \implies \int_0^{\Delta t} \int_\Gamma \mathbf{n} \cdot (-f_s \nabla p_c) \, \mathrm{d}\Gamma \, dt &= \frac{\delta V}{\delta t}. \end{aligned} \tag{4.12}$$

A smoothed Heaviside function of ϕ is introduced in order to differentiate p_c at the interface

$$H_\epsilon(\phi) = \begin{cases} 1 & \text{if } \phi > \epsilon \\ \frac{1}{2}\left[1 + \frac{\phi}{\epsilon} + \frac{1}{\pi}\sin(\frac{\pi\phi}{\epsilon})\right] & \text{if } |\phi| \le \epsilon \\ 0 & \text{if } \phi < -\epsilon, \end{cases} \tag{4.13}$$

with $\epsilon = 1.5\Delta x$ the half smoothing width. The pressure can then be written as

$$p_c = (1 - H_\epsilon(\phi)) p_0, \tag{4.14}$$

$$\implies \nabla p_c = -\delta_\epsilon(\phi) \nabla \phi p_0, \tag{4.15}$$

with $\delta_\epsilon(\phi)$ the derivative of $H_\epsilon(\phi)$), and p_0 a constant. Note that $\mathbf{n} \cdot \nabla \phi = |\nabla \phi|$ and define

$$A_f := \int_\Gamma f_s \delta_\epsilon(\phi) |\nabla \phi| \, \mathrm{d}\Gamma. \tag{4.16}$$

Then Eq. (4.15) is substituted in Eq. (4.12) and $\int_0^{\Delta t} A_f \, dt$ is approximated by $\Delta t A_f$, resulting in

$$p_0 = \frac{\delta V}{\delta t} \frac{1}{A_f \Delta t} \nabla \phi. \tag{4.17}$$

Finally Eqs. (4.11), (4.15) and (4.17) are combined, which yields

$$\mathbf{u}_c(\phi) = \frac{\delta V}{\delta t} \frac{f_s \delta_\epsilon(\phi)}{A_f} \nabla \phi. \tag{4.18}$$

The level-set field can now be advected according to Eq. (4.10) with the constructed correction-velocity from Eq. (4.18). This concludes the volume correction procedure of the ICLS method.

4.6.1. Choice of speed function

The user has the freedom of choice for selecting an appropriate speed function f_s. However, for this formulation of the ICLS method only non-negative functions are valid. Two possible choices, introduced in [6], are

$$f_s = \begin{cases} 1 & \text{uniform speed} \\ \kappa(\phi) & \text{curvature-dependent speed.} \end{cases} \tag{4.19}$$

The uniform speed can be thought of as inflating a balloon by the volume loss over time ∂t. For spherical droplets this is an appropriate choice, since it should remain a sphere. However, when the interface has sharper edges or is subject to deformations, a uniform speed will not satisfy. In such cases, a curvature-dependent speed will be a more suitable choice, since regions with a high curvature are more likely to suffer from volume loss/gain. Hence, this speed function will yield a more accurate result, since local information is used to obtain global volume conservation.

4.7. Tested methods in this project

In the course of this project several methods were proposed and te sted. In this section we will shortly discuss the tested methods. First, we tried to make the MCLS method from [17] continuous after the discontinuous volume correction procedure. This was achieved by applying an L_2-projection step to the discontinuous LS field to make the field continuous. While this did indeed improve the volume conservation, it was not sufficient. After two or three iterations convergence was achieved, but with an inaccurate result. Furthermore, we tried to apply the iterative correction process from [21] and [4] for a triangular mesh. We found that a lot of iterations were necessary to obtain an accurate solution.

Therefore, we looked for an alternative approach which requires less iterations. The ICLS method is able to handle unstructured triangular meshes. So the aim was to adapt the formulation of ICLS in order to satisfy the requirements of the goal for this project. In this thesis we developed an extension to ICLS by introducing a coupling between the VoF and the ICLS method, which is discussed in Sections 5 and 6.

5

Volume-of-fluid-based interface-correction level-set (VOF-ICLS) method

In this section we introduce the developed extension to the original ICLS method; the volume-of-fluid-based interface-correction level-set (VOF-ICLS) method. In [6] two speed functions were introduced for the ICLS method; uniform speed and curvature-dependent speed. In this thesis we propose a new speed function based on a coupling with the VoF method

$$f_s = A(\psi - \psi_\phi), \tag{5.1}$$

where ψ and ψ_ϕ are the volume-of-fluid fields obtained from advection and construction from the advected level-set, respectively, and A is the area.

The motivation for this choice of speed function is that local volume information is taken into account for maintaining global volume conservation. However, this choice is not trivial since f_s can be negative, which can not be used with the original formulation of the method. Furthermore, the volume fractions are given inside the cells of the primal mesh, but the speed function values are required at the nodes of the primal mesh. In the next part we will go more into detail on how to tackle these problems.

5.1. Dual mesh construction

The level-set function is defined on the primal mesh $\mathscr{P}$. For this method we construct a dual mesh, denoted by $\mathscr{D}$, on which we define the volume-of-fluid function. In this way, the volume fractions are given at the nodes of the primal mesh, since the primal nodes are located inside the dual cells. The dual mesh $\mathscr{D}$ is generated as follows:

1. For every element K in $\mathscr{P}$:
   For each edge of K a dual edge is defined by connecting the centroid of the edge and the centroid of the element.

2. For every node $\mathbf{v}$ in $\mathscr{P}$ with primal edges $e_1, \ldots, e_n$, define the dual cell $T_\mathbf{v}$ as the region bounded by all dual edges. When this region is open, take the primal edges placed on the boundary of the domain to create a closed region.

This gives us a dual mesh, as can be seen in Figure 5.1. So LS advection takes place on the primal mesh, and VoF advection on the dual mesh. The speed function is evaluated on the primal mesh.

5.2. Volume fraction construction

After LS and VoF advection, we have ϕ defined on the primal mesh and ψ defined on the dual mesh. Identical to Eq. (4.7) from the MCLS [17] method, we construct volume fractions from the advected level-set field,

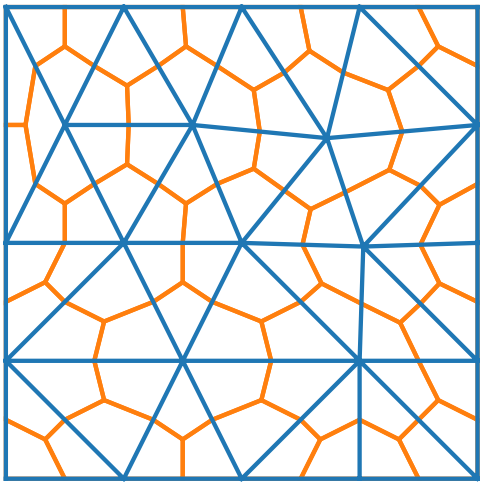


Figure 5.1: Primal mesh (blue) and constructed dual mesh (orange).

denoted by ψ_ϕ. For a primal cell K with control volume Ω_k, we have

$$\psi_{\phi,K} = \frac{1}{|\Omega_k|}\int_{\Omega_k} H(\phi)\,\mathrm{d}\Omega. \tag{5.2}$$

However, in contrast to the procedure of the MCLS method, we construct the volume fractions on the dual mesh rather than the primal mesh, as has been done in [4]. Therefore the formulation needs to be adjusted as follows

$$\psi_{\phi,T} = \frac{1}{|\Omega_t|}\int_{\Omega_t} H(\phi)\,\mathrm{d}\Omega, \tag{5.3}$$

with T a dual element and Ω_t its corresponding control volume.

The computation of the VoF function is executed with the use of barycentric transformations. For the complete derivation we like to refer to [17]. In this thesis we will only give the results.

5.2.1. Approximation

Let the control volume of element K be denoted by Ω_k with nodes $\mathbf{x}_{k,1}$, $\mathbf{x}_{k,2}$ and $\mathbf{x}_{k,3}$ and centroid $\mathbf{x}_k$. The formulation from Eq. (5.2) is inefficient to compute, so the following approximation is used

$$\begin{aligned}\psi_k(t) &= \frac{1}{|\Omega_k|}\int_{\Omega_k} H(\tilde{\phi} + c_1 h^2)\,\mathrm{d}\Omega \\ &= \frac{1}{|\Omega_k|}\int_{\Omega_k} H(\tilde{\phi}) + c_2 h^2\,\mathrm{d}\Omega = f(\phi_k, \nabla\phi_k) + c_3 h^2,\end{aligned} \tag{5.4}$$

with $\tilde{\phi}$ being the linearisation of ϕ around centroid $\mathbf{x}_k$

$$\tilde{\phi}(\mathbf{x}, t) = \phi(\mathbf{x}_k, t) + \frac{\partial\phi}{\partial x}(x - x_k) + \frac{\partial\phi}{\partial y}(y - y_k), \tag{5.5}$$

and $h = \sqrt{|\Omega_k|}$ and ϕ_k is the level-set function at the centroid $\mathbf{x}_k$, $\phi_k(t) = \phi(\mathbf{x}_k, t)$. Now we seek function $f(\phi_k, \nabla\phi_k)$ for construction of the volume fractions.

5.2.2. Barycentric transformation

The barycentric transformation for a triangle Ω_k with vertices $\mathbf{x}_{k,1}, \mathbf{x}_{k,2}$ and $\mathbf{x}_{k,3}$ is based on the areas of three sub-triangles divided by an arbitrary point $\mathbf{x} \in \Omega_k$, with the barycentric coordinates defined as

$$\xi_1 = \frac{|\triangle \mathbf{x}_{k,1}\,\mathbf{x}\,\mathbf{x}_{k,3}|}{|\Omega_k|}, \quad \xi_2 = \frac{|\triangle \mathbf{x}_{k,1}\,\mathbf{x}_{k,3}\,\mathbf{x}|}{|\Omega_k|}, \quad \xi_3 = \frac{|\triangle \mathbf{x}\,\mathbf{x}_{k,2}\,\mathbf{x}_{k,3}|}{|\Omega_k|}, \tag{5.6}$$

where $\triangle \mathbf{x}_{k,1}\,\mathbf{x}\,\mathbf{x}_{k,3}$ denotes the triangle formed by vertices $\mathbf{x}_{k,1}, \mathbf{x}$ and $\mathbf{x}_{k,3}$.

For $\mathbf{x} \in \Omega_k$ we have $\xi \in [0,1]$. This gives the following coordinate transformation on Ω_k:

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_{k,1} & \mathbf{x}_{k,2} & \mathbf{x}_{k,3} \end{bmatrix} \begin{bmatrix} \xi_1 \\ \xi_2 \\ \xi_3 \end{bmatrix}. \tag{5.7}$$

5.2.3. The volume of fluid function

For this part we assume that fluid 1 ($\psi = 1$) corresponds to $\phi \leq 0$. Since the contour lines of the linearised level-set function are straight line segments the volume fraction can be constructed by calculating the relative area of polygon P, defined by

$$P = \left\{ \mathbf{x} \in \Omega_k | \phi(\mathbf{x}) \leq 0 \right\}. \tag{5.8}$$

For polygon P the following cases are possible:

- Case I: $\phi < 0$ for one node, P is triangular.
- Case II: $\phi > 0$ for one node, P is quadrilateral.

While for both cases a derivation for the VoF function is possible, it is sufficient to only consider case I, since $1 - f(-\phi_k, -\nabla\phi_k)$.

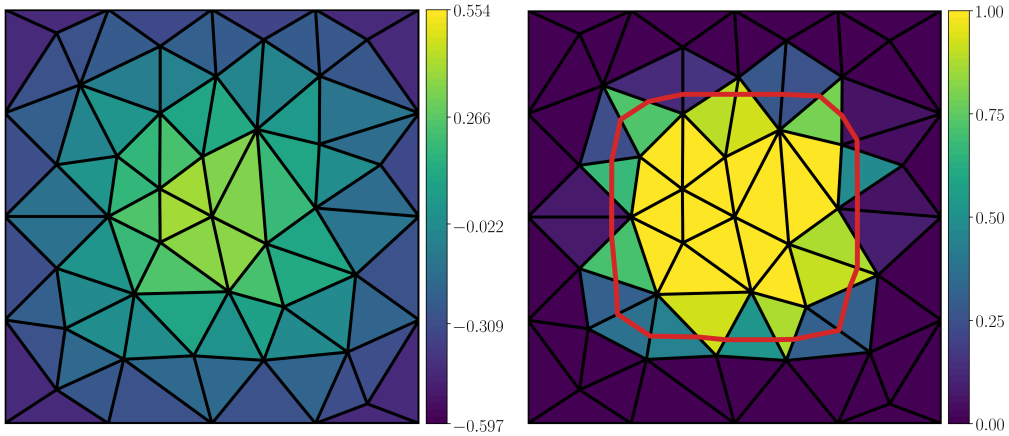
Without loss of generality, we assume for case I that $\phi(\mathbf{x}_{k,3}) < 0$. Node $\mathbf{x}_{k,3}$ is mapped to the origin and the other nodes are mapped to $(1,0)^\top$ and $(0,1)^\top$ in logical space. We define

$$D_{\xi_1} := \frac{\partial \phi}{\partial \xi_1}, \quad D_{\xi_2} := \frac{\partial \phi}{\partial \xi_2}. \tag{5.9}$$

From the derivation found in [17] we get

$$\psi : \left[-D_{\xi_2}, 0 \right] \times \mathbb{R}^2 \rightarrow \left[0, \frac{D_{\xi_2}}{D_{\xi_1}} \right], \quad \psi \left(\phi(\mathbf{x}_{k,3}), \mathbf{D}_{\boldsymbol{\xi}} \right) = \frac{\phi(\mathbf{x}_{k,3})^2}{D_{\xi_1} D_{\xi_2}}. \tag{5.10}$$

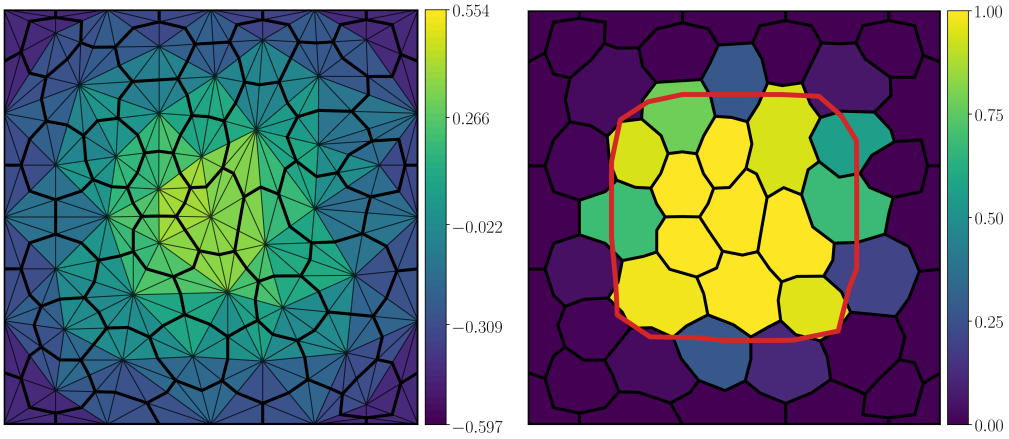
With Eq. (5.10) the volume fraction for case II can also be computed. With this approach we also yield the coordinates of the interface at the edges of the interface elements, from which we can construct the interface contour. An example of VoF construction on the primal mesh is shown in Figure 5.2.



(a) LS field (b) VoF field on primal mesh with interface

Figure 5.2: VoF construction on primal mesh.

Since the VoF field needs to be constructed on the dual mesh, as stated in Eq. (5.3), this procedure requires an extra step. The VoF construction algorithm works for triangles, however, the dual elements are star-shaped polygons. Therefore we create sub triangles in the dual elements as shown in Figure 5.3(a). For all dual elements the volume fraction for each sub triangle is computed and added together, resulting in the VoF field on the dual mesh. The VoF construction on the dual mesh can be seen Figure 5.3.



(a) LS field with sub triangles displayed inside the dual elements

(b) VoF field on dual mesh with interface

Figure 5.3: VoF construction on dual mesh.

5.3. Speed function

Now we have all the information to construct the speed function

$$\tilde{f}_{s,T} = |\Omega_t|(\psi_T - \psi_{\phi,T}), \quad \forall T \in \mathscr{D}, \tag{5.11}$$

with Ω_t the control volume of dual element T. If volume loss occurs in a cell we have $\tilde{f}_{s,T} > 0$, and $\tilde{f}_{s,T} < 0$ if there is volume gain. This speed function will tell how much volume needs to be restored relatively, so when $|\tilde{f}_{s,T}|$ is large, more volume will be restored in that cell compared to cells where $|\tilde{f}_{s,T}|$ is small. However, this does not mean that all volume loss/gain will be restored.

Since volume loss/gain can only occur at the interface, the speed function is by definition zero everywhere else. Hence, the correction-velocity will only have non-zero values at the interface. When this correction-velocity is used for advection, there will be very little correction and hence very little improvement to the level-set field. Therefore we smooth the speed function in order to obtain non-zero values near the interface. The smoothing is achieved as follows. We want to minimize the gradient norm of the speed function near the interface, which corresponds to solving the Laplace equation over the entire domain. The speed function values at the interface cells are taken as Dirichlet boundary conditions. Furthermore, we impose homogeneous Neumann boundary conditions at the boundary. The original non-smoothed speed function is denoted by $\tilde{f}_s$, and the smoothed function by f_s, which gives us the following problem

$$\begin{cases} \Delta f_s = 0, & \mathbf{x} \in \Omega, \\ \frac{\partial f_s}{\partial \mathbf{n}} = 0, & \mathbf{x} \in \partial\Omega, \\ f_s = \tilde{f}_s, & \mathbf{x} \in \Gamma. \end{cases} \tag{5.12}$$

The discretization can be found in Appendix B.1. The Laplace equation is solved on the primal mesh, hence the solution is given inside the dual cells. The smoothing result can be seen in Figure 5.4 and the effect on the correction-velocity is displayed in Figure 5.5. The correction-velocity field is depicted in orange, interface element are coloured red when the there is volume loss and coloured blue when the there is volume gain. The colours are based on a colour map, when there is less volume loss/gain, they will be more white-coloured.

5.4. Scaling

For the most part the VOF-ICLS method is similar to the ICLS method. However, in the original ICLS formulation A_f is defined as

$$A_f := \int_\Gamma f_s \delta_\epsilon(\phi) |\nabla \phi| \, \mathrm{d}\Gamma. \tag{5.13}$$

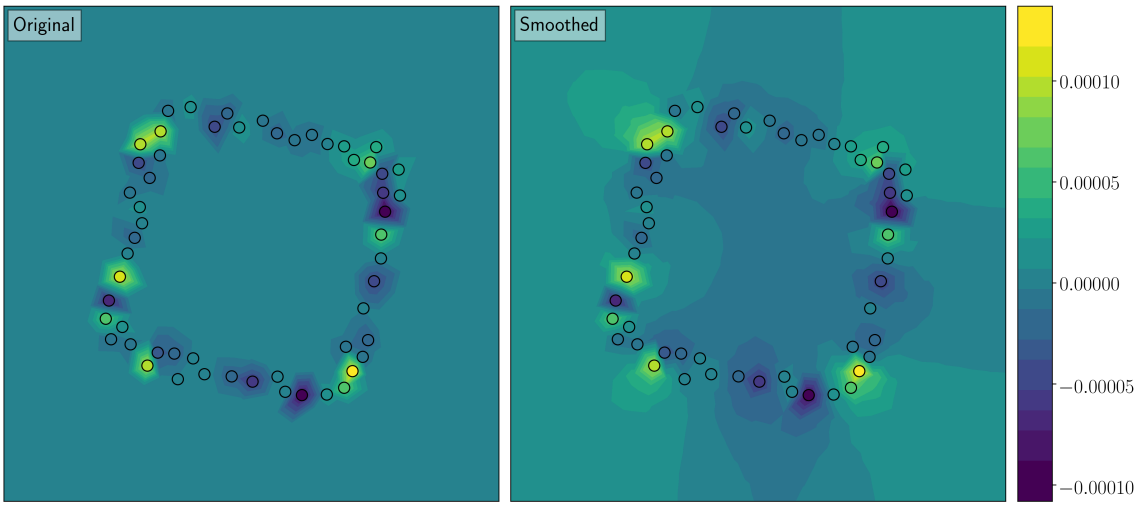
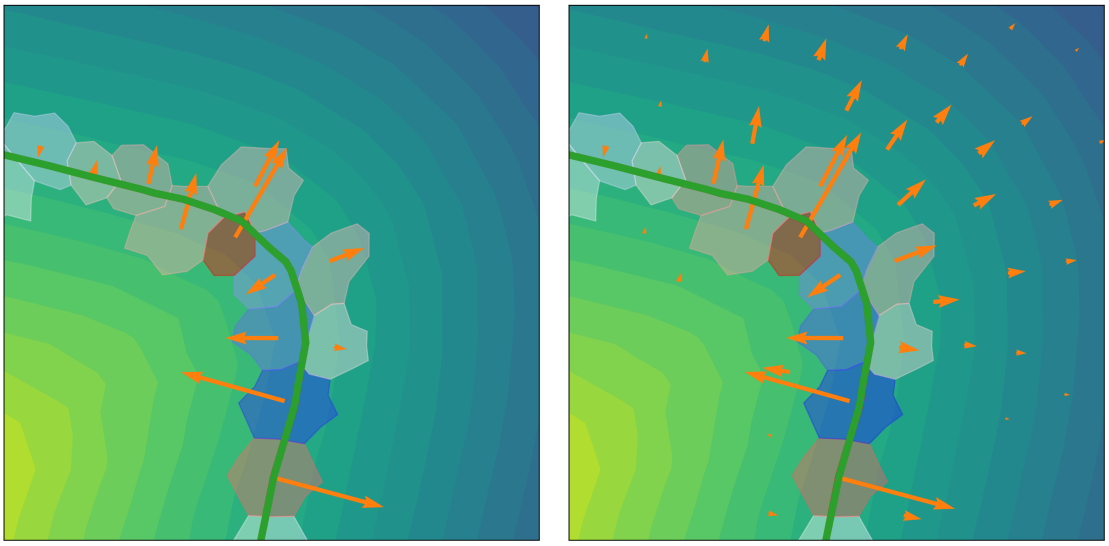


Figure 5.4: Speed function before and after smoothing with the points being the Dirichlet boundary conditions.



(a) Original speed function (b) Smoothed speed function

Figure 5.5: Correction-velocity with original and smoothed speed function.

Since the VOF-based speed function can have negative values, this formulation is no longer valid. Therefore we define

$$a(\phi, f_s) := f_s \delta_\epsilon(\phi) |\nabla \phi|,$$

and

$$a^+(\phi, f_s) := \max\{0, a(\phi, f_s)\}, \quad a^-(\phi, f_s) := \min\{0, a(\phi, f_s)\}.$$

With these definitions we can define and compute

$$A_f^+ := \int_\Gamma a_f^+ \, \mathrm{d}\Gamma, \quad A_f^- := \int_\Gamma |a_f^-| \, \mathrm{d}\Gamma. \tag{5.14}$$

Furthermore, we construct two separate velocity fields for both the outward and inward direction, where the outward direction corresponds to $f_s \geq 0$ (volume loss) and the inward direction to $f_s < 0$ (volume gain):

$$\begin{aligned} \mathbf{v}^+ &:= \begin{cases} \frac{-f_s \delta_\epsilon(\phi) \nabla \phi}{A_f^+} & \text{if } f_s \geq 0 \\ 0 & \text{if } f_s < 0, \end{cases} \\ \mathbf{v}^- &:= \begin{cases} 0 & \text{if } f_s \geq 0 \\ \frac{-f_s \delta_\epsilon(\phi) \nabla \phi}{A_f^-} & \text{if } f_s < 0. \end{cases} \end{aligned} \tag{5.15}$$

The correction-velocity is obtained by combining $\mathbf{v}^+$ and $\mathbf{v}^-$ with weights γ^+ and γ^-,

$$\mathbf{u}_c = \frac{\delta V}{\delta t} (\gamma^+ \mathbf{v}^+ + \gamma^- \mathbf{v}^-). \tag{5.16}$$

Weights γ^+ and γ^- have to be chosen such that $\int_\Gamma \mathbf{n} \cdot \mathbf{u}_c \, \mathrm{d}\Gamma = \frac{\delta V}{\delta t}$ still holds. Substituting Eq. (5.16) into Eq. (4.9) gives us

$$\begin{aligned} \int_\Gamma \mathbf{n} \cdot \mathbf{u}_c \, \mathrm{d}\Gamma &= \int_\Gamma \mathbf{n} \cdot \frac{\delta V}{\delta t} (\gamma^+ \mathbf{v}^+ + \gamma^- \mathbf{v}^-) \, \mathrm{d}\Gamma \\ &= \frac{\delta V}{\delta t} \left(\gamma^+ \int_\Gamma \mathbf{n} \cdot \mathbf{v}^+ \, \mathrm{d}\Gamma + \gamma^- \int_\Gamma \mathbf{n} \cdot \mathbf{v}^- \, \mathrm{d}\Gamma \right) \\ &= \frac{\delta V}{\delta t} \left(\gamma^+ - \gamma^- \right) = \frac{\delta V}{\delta t}, \end{aligned}$$

from which we find that $\gamma^+ - \gamma^- = 1$ is a required condition.

Now we need to choose appropriate values for γ^+ and γ^-. Suppose that there is only volume loss and nowhere volume gain, so there is volume loss globally. Then we would have $A_f^+ > 0$ and $A_f^- = 0$. For this situation $\gamma^+ = 1$ and $\gamma^- = 0$ would be the logic choice. However, if there is volume gain locally, we would have $A_f^+ > 0$ and $A_f^- > 0$. If we also want to restore regions with volume gain, we must have $\gamma^- > 0$. Then the area enclosed by the interface will lose volume after advecting with the correction-velocity near regions with $f_s < 0$. Therefore γ^+ needs to be chosen larger in order to compensate for the extra volume loss. An appropriate choice when there is global volume loss is

$$\gamma^+ = \frac{A_f^+ + A_f^-}{A_f^+}, \quad \gamma^- = \frac{A_f^-}{A_f^+}, \tag{5.17}$$

and for this choice we have $\gamma^+ - \gamma^- = 1$. If there is volume gain globally, we choose

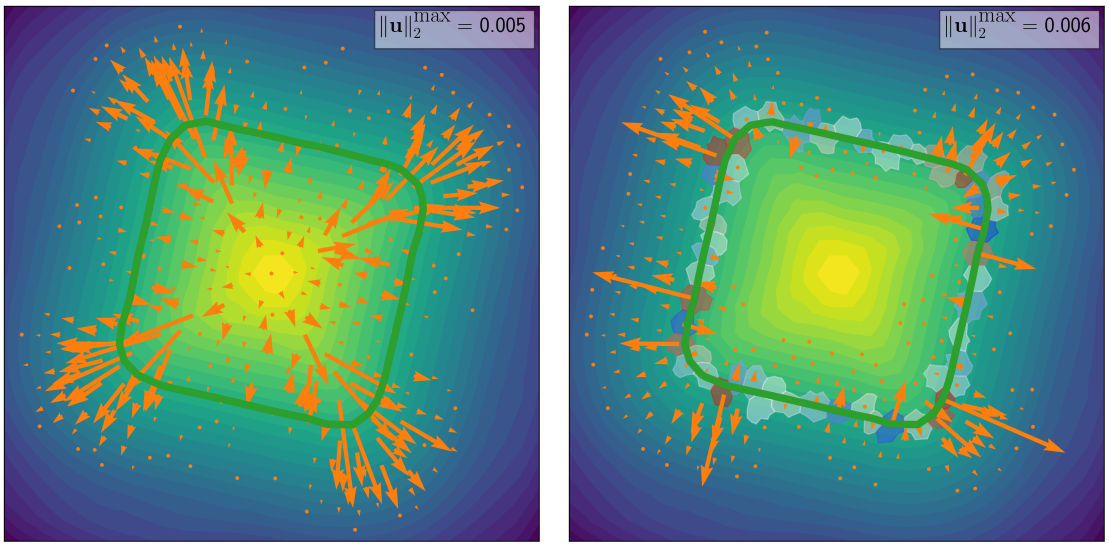
$$\gamma^+ = -\frac{A_f^+}{A_f^-}, \quad \gamma^- = -\frac{A_f^+ + A_f^-}{A_f^-}, \tag{5.18}$$

for which the condition $\gamma^+ - \gamma^- = 1$ also holds.

The level-set field can now be advected according to Eq. (4.10) with the constructed correction-velocity from Eq. (5.16).

5.5. Comparison with the ICLS method

In this section we will evaluate the performance of the original ICLS method and the developed VOF-ICLS method, and discuss their differences. The biggest difference is the choice of speed function. Figure 5.6 shows the velocity field with the curvature-dependent (ICLS) and VoF-based (VOF-ICLS) speed function. The latter speed function is more concentrated around the interface and the maximum velocity norm $\|\mathbf{u}\|_2^{\max}$ is generally larger than for the curvature-dependent speed function. Furthermore, the curvature-dependent speed function is either inward-pointed or outward-pointed, while the VoF-based function can be both.



(a) Curvature-dependent speed function

(b) VoF-based speed function

Figure 5.6: Correction-velocity with curvature-dependent and VoF-based speed function.

Their performance is investigated by simulating a droplet in a rotating flow. For the droplet shape we use a superellipse

$$\left|\frac{x-a}{r_a}\right|^n + \left|\frac{y-b}{r_b}\right|^n = 1, \quad a,b \in \mathbb{R}, n \in \mathbb{N}, \tag{5.19}$$

with r_a and r_b are the semi-major and semi-minor axes. Here we use $r_a = r_b$ and we take n even, which gives the equation

$$(x-a)^n + (y-b)^n = r^n, \tag{5.20}$$

which will be denoted as ball-n.

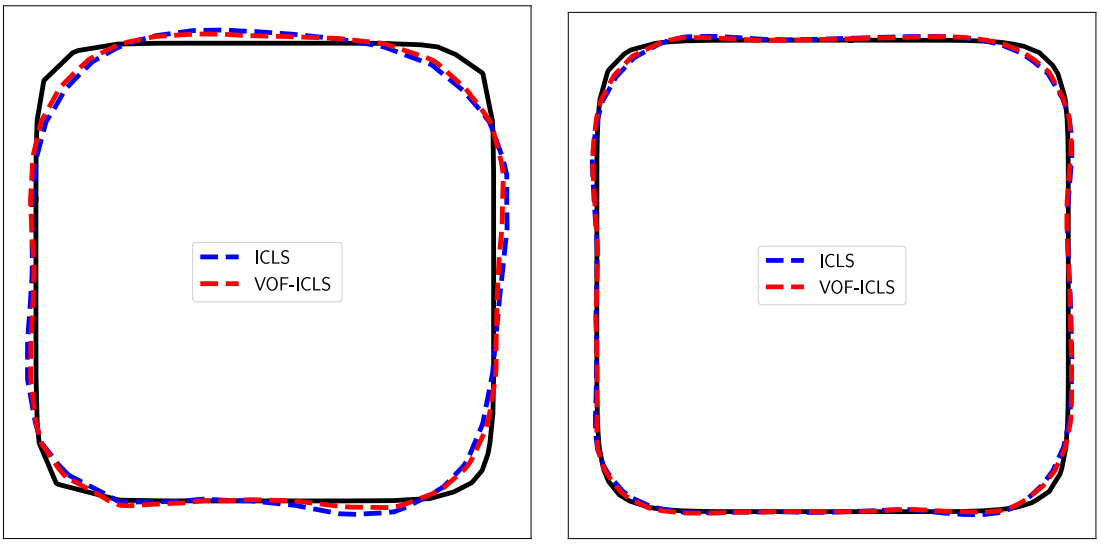
A droplet ball-8 is simulated for $t = 2\pi$, with $a = 0.15$, $b = 0.15$, $r = 0.3$ in domain $\Omega = [-1,1] \times [-1,1]$. The results can be seen in Figure 5.7 for different mesh sizes n_e. From the results we find that VOF-ICLS is only slightly better, and by increasing the mesh size this difference becomes smaller.

For a more thorough evaluation we will consider the following errors to measure the performance:

$$\begin{aligned} \overline{E_\Gamma} &:= \frac{1}{\#\mathscr{D}_\Gamma} \sum_{T \in \mathscr{D}_\Gamma} |\psi_T - \psi_{\phi,T}| \quad \text{(mean error at interface)}, \\ E_\Gamma^{\max} &:= \max_{T \in \mathscr{D}_\Gamma} |\psi_T - \psi_{\phi,T}| \quad \text{(maximum error at interface)}, \end{aligned} \tag{5.21}$$

with $\mathscr{D}_\Gamma$ the interface elements. Note that ψ_T is the advected volume fraction in cell T at the current time step (not the exact volume fraction) and $\psi_{\phi,T}$ is the corrected constructed volume fraction from the level-set field at the current time step in cell T.

From Figures 5.8-5.10 and Table 5.1 we can conclude that VOF-ICLS is slightly better at preserving volume locally, but worse at preserving volume globally. This is due to the fact that the correction-velocity is pointed inward and outward with respect to the interface. Maintaining global volume conservation is easier when the



(a) $n_e = 1451$ (b) $n_e = 5690$

Figure 5.7: Result of ICLS and VOF-ICLS method at $t = 2\pi$ compared to the exact solution (black).

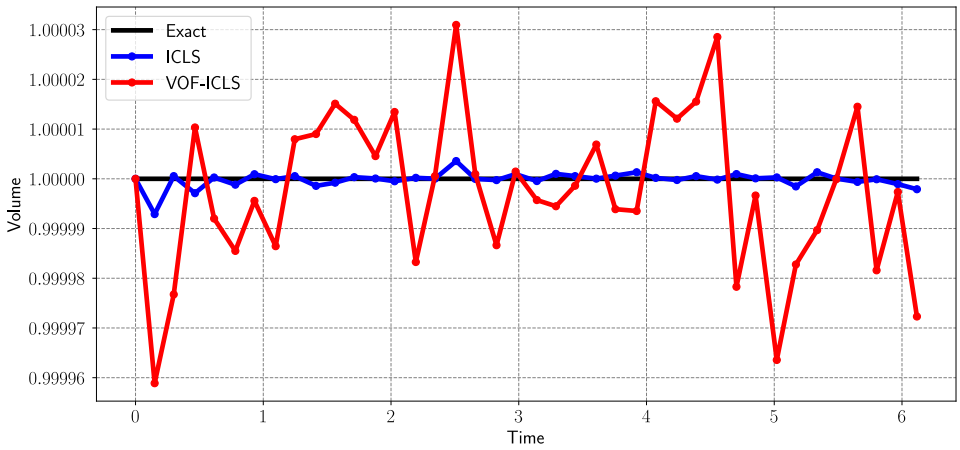


Figure 5.8: Volume of droplet ball-8 in rotating flow of ICLS and VOF-ICLS method.

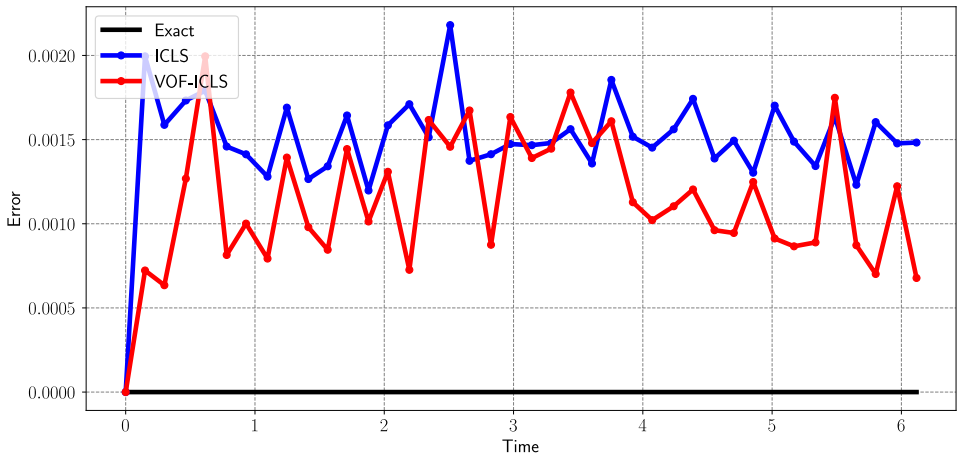


Figure 5.9: Error $\overline{E_\Gamma}$ of droplet ball-8 in rotating flow of ICLS and VOF-ICLS method.

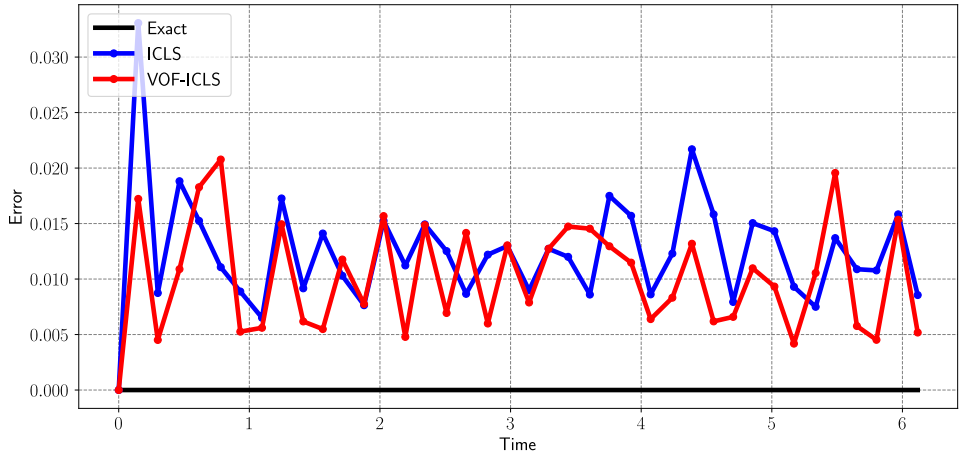


Figure 5.10: Error $E_\Gamma^{\max}$ of droplet ball-8 in rotating flow of ICLS and VOF-ICLS method.

	$\sum^{n_{\text{steps}}} \overline{E_\Gamma} / n_{\text{steps}}$	$\sum^{n_{\text{steps}}} E_\Gamma^{\max} / n_{\text{steps}}$
ICLS	2.173×10^{-3}	1.723×10^{-2}
VOF-ICLS	1.574×10^{-3}	1.309×10^{-2}

Table 5.1: Error analysis for comparison ICLS and VOF-ICLS method.

interface is just deflated or inflated, but less accurate however. Note that while the volume of the VOF-ICLS simulation is not always accurate, it never diverges too much from the exact volume (volume = 1), because in each correction step, the current volume is compared to the initial volume. This is a good property for persevering the global volume. Throughout the simulation the volume of the VOF-ICLS method diverges more from the exact volume than ICLS, but the mean and maximum error is smaller, as can be seen in Table 5.1.

From the results we find that taking local information into account has a positive effect on local conservation. This shows promising results, however, another approach is required in order to fully utilize the available local volume loss/gain. An improved version is proposed in Section 6.

6

Volume-of-fluid-based local interface-correction level-set (VOF-LICLS) method

As discussed in the previous section, the ICLS method gives no guarantee that volume is restored in each cell after applying the correction advection. In this section we aim to develop an extension to the VOF-ICLS method, such that volume is restored in each cell. While the ICLS method tries to satisfy Eq. (4.9) over the entire interface, the volume-of-fluid-based local interface-correction level-set (VOF-LICLS) method attempts to construct a correction-velocity field such that for every interface cell $\int_{\Gamma_T} \mathbf{n} \cdot \mathbf{u}_c \, \mathrm{d}\Gamma$ equals the local volume flux of the interface cell.

6.1. Improved algorithm

Instead of calculating the global volume flux, the local volume flux is computed for every dual interface cell $T \in \mathscr{D}_\Gamma$:

$$V_{\tau,T} := \left(\frac{\partial V}{\partial \tau}\right)_T = \frac{|\Omega_t|(\psi_T - \psi_{\phi,T})}{\partial \tau}, \quad \forall T \in \mathscr{D}_\Gamma. \tag{6.1}$$

The correction-velocity is constructed using weights, denoted by w, for each cell

$$\mathbf{u}_c = w\tilde{\mathbf{v}}, \tag{6.2}$$

with

$$\tilde{\mathbf{v}} = \delta_\epsilon(\phi)\nabla\phi.$$

Here, $\tilde{\mathbf{v}}$ is essentially the unscaled field of Eq. (4.18). The weights are obtained by scaling the local volume fluxes, such that

$$\int_{\Gamma_T} \mathbf{n} \cdot \mathbf{u}_c \, \mathrm{d}\Gamma = V_{\tau,T} \tag{6.3}$$

holds for every interface cell. Here, Γ_T is defined as $\Gamma_T := \Gamma \cap T$, i.e. the part of the interface contained in dual cell T. This gives us

$$\begin{aligned} \int_{\Gamma_T} w\left(V_{\tau,T}\right)\delta_\epsilon(\phi)|\nabla\phi| \, \mathrm{d}\Gamma &= V_{\tau,T}, \\ \implies w\left(V_{\tau,T}\right) &= \frac{V_{\tau,T}}{\int_{\Gamma_T} \delta_\epsilon(\phi)|\nabla\phi| \, \mathrm{d}\Gamma}, \quad \forall T \in \mathscr{D}_\Gamma. \end{aligned} \tag{6.4}$$

However, when either $|\Gamma_T|$ is small or $\delta_\epsilon(\phi(\mathbf{x}))|\nabla\phi(\mathbf{x})|$ is close to zero for $\mathbf{x} \in T$, the value of the line integral will be close to zero. This causes the weight from Eq. (6.4) to blow up. Therefore this formulation is not valid

for numerical simulation. For the next formulation we scale the local volume fluxes with respect to the total outward and inward volume fluxes. In order to achieve this we define the sets of dual interface cells with volume loss and volume gain,

$$\begin{aligned} \mathscr{D}_\Gamma^+ &:= \{T \in \mathscr{D}_\Gamma : V_{\tau,T} > 0\}, \\ \mathscr{D}_\Gamma^- &:= \{T \in \mathscr{D}_\Gamma : V_{\tau,T} \leq 0\}. \end{aligned} \tag{6.5}$$

The total outward and inward volume flux are calculated as follows

$$V_\tau^+ := \sum_{T \in \mathscr{D}_\Gamma^+} V_{\tau,T}, \quad V_\tau^- := \sum_{T \in \mathscr{D}_\Gamma^-} V_{\tau,T}, \tag{6.6}$$

respectively. Now we want to scale the volume fluxes such that

$$\begin{cases} \int_{\Gamma^+} \mathbf{n} \cdot \mathbf{u}_c \, \mathrm{d}\Gamma &= V_\tau^+, \\ \int_{\Gamma^-} \mathbf{n} \cdot \mathbf{u}_c \, \mathrm{d}\Gamma &= V_\tau^-, \end{cases} \tag{6.7}$$

is satisfied, where Γ^+ and Γ^- (see Figure 6.1) are defined as

$$\Gamma^+ := \bigcup_{T \in \mathscr{D}_\Gamma^+} \{\Gamma \cup T\}, \quad \Gamma^- := \bigcup_{T \in \mathscr{D}_\Gamma^-} \{\Gamma \cup T\}. \tag{6.8}$$

First we consider the discrete field $\mathbf{v}$, with

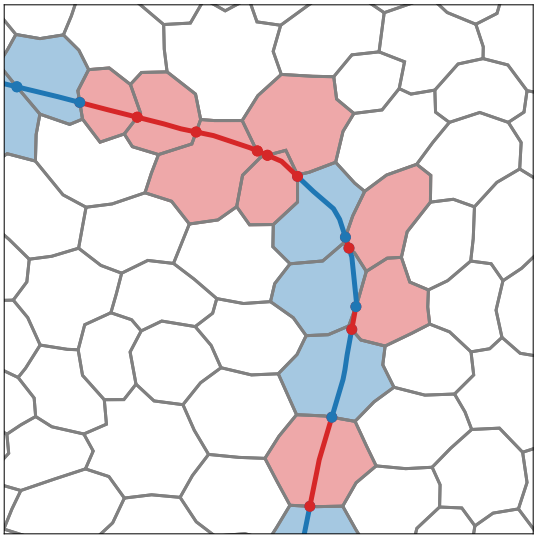


Figure 6.1: Interface divided into Γ^+ (red line segments) and Γ^- (blue line segments). The dual elements with outward and inward flux ($\mathscr{D}_\Gamma^+$ and $\mathscr{D}_\Gamma^-$) are displayed by the red and blue elements, respectively.

$$\mathbf{v}_T = V_{\tau,T} \delta_\epsilon(\mathbf{x}) \nabla\phi(\mathbf{x}), \quad \mathbf{x} \in T, \forall T \in \mathscr{D}, \tag{6.9}$$

This discrete field will be called the 'volume flux field'. The volume flux field is integrated over Γ^+ and Γ^-. These line integrals are evaluated as follows

$$\begin{aligned} \int_{\Gamma^+} \mathbf{n} \cdot \mathbf{v} \mathrm{d}\Gamma &= \sum_{T \in \mathscr{D}_\Gamma^+} \int_{\Gamma_T} \mathbf{n} \cdot \mathbf{v} \mathrm{d}\Gamma, \\ &= \sum_{T \in \mathscr{D}_\Gamma^+} V_{\tau,T} \int_{\Gamma_T} \delta_\epsilon(\phi) |\nabla\phi| \, \mathrm{d}\Gamma \end{aligned} \tag{6.10}$$

and the last integral is numerically computed by interpolation. Then the scaling factor can be calculated as follows

$$\xi^+ = \frac{V_\tau^+}{\int_{\Gamma^+} \mathbf{n} \cdot \mathbf{v} \mathrm{d}\Gamma} = \frac{V_\tau^+}{\sum_{T \in \mathscr{D}_\Gamma^+} V_{\tau,T} \int_{\Gamma_T} \delta_\epsilon(\phi) |\nabla\phi| \, \mathrm{d}\Gamma}. \tag{6.11}$$

Then the weights become

$$\tilde{w}^{+}\left(V_{\tau,T}\right) = \xi^{+} V_{\tau,T}, \quad \forall T \in \mathscr{D}_{\Gamma}^{+}, \tag{6.12}$$

for the outward flux. The procedure is the same for the inward flux, which gives $\tilde{w}^{-}$. The correction-velocity can be constructed as follows

$$\tilde{\mathbf{u}} = \begin{cases} \tilde{w}^{+}\left(V_{\tau,T}\right)\delta_{\epsilon}(\phi)\nabla\phi & \text{for } T \in \mathscr{D}_{\Gamma}^{+} \\ \tilde{w}^{-}\left(V_{\tau,T}\right)\delta_{\epsilon}(\phi)\nabla\phi & \text{for } T \in \mathscr{D}_{\Gamma}^{-} \\ 0 & \text{elsewhere.} \end{cases} \tag{6.13}$$

This formulation of the correction-velocity has only non-zero values at the interface. The interface values are extended to the whole domain by solving the Laplace equation with Dirichlet boundary condition, see Figure 6.2. This is similar to the procedure from Eq. (5.12),

$$\begin{cases} \Delta w = 0, & \mathbf{x} \in \Omega, \\ \frac{\partial w}{\partial \mathbf{n}} = 0, & \mathbf{x} \in \partial\Omega, \\ w = \tilde{w}, & \mathbf{x} \in \Gamma. \end{cases} \tag{6.14}$$

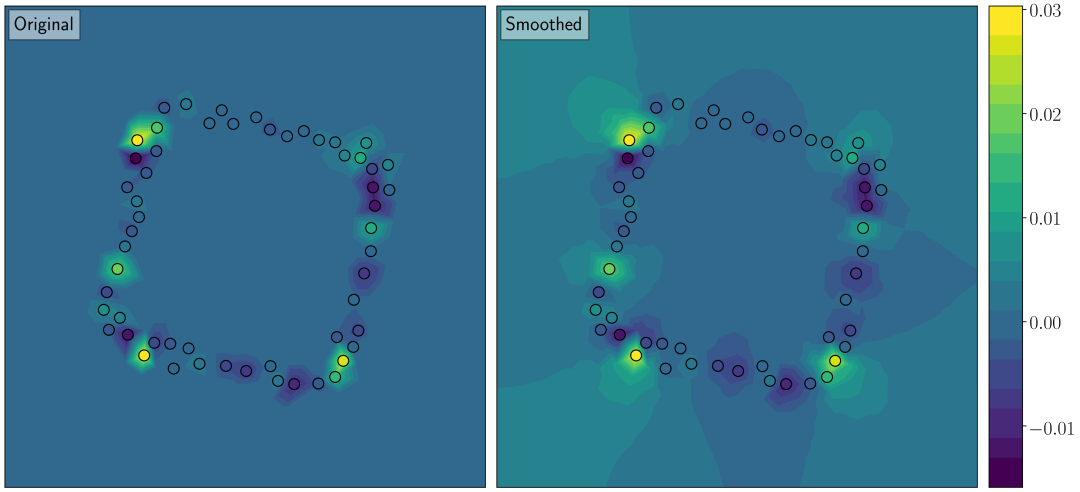


Figure 6.2: Original and smoothed weights obtained by solving the Laplace equation.

And finally, we get the correction-velocity

$$\mathbf{u}_c = w\left(V_{\tau,T}\right)\delta_{\epsilon}(\phi)\nabla\phi, \tag{6.15}$$

which satisfies Eq. (6.7). In Figure 6.3 correction-velocity $\mathbf{u}_c$ of the VOF-LICLS method is displayed. Compared to Figure 5.6, $\|\mathbf{u}\|_2^{\max}$ is larger, hence yielding a more aggressive volume correction.

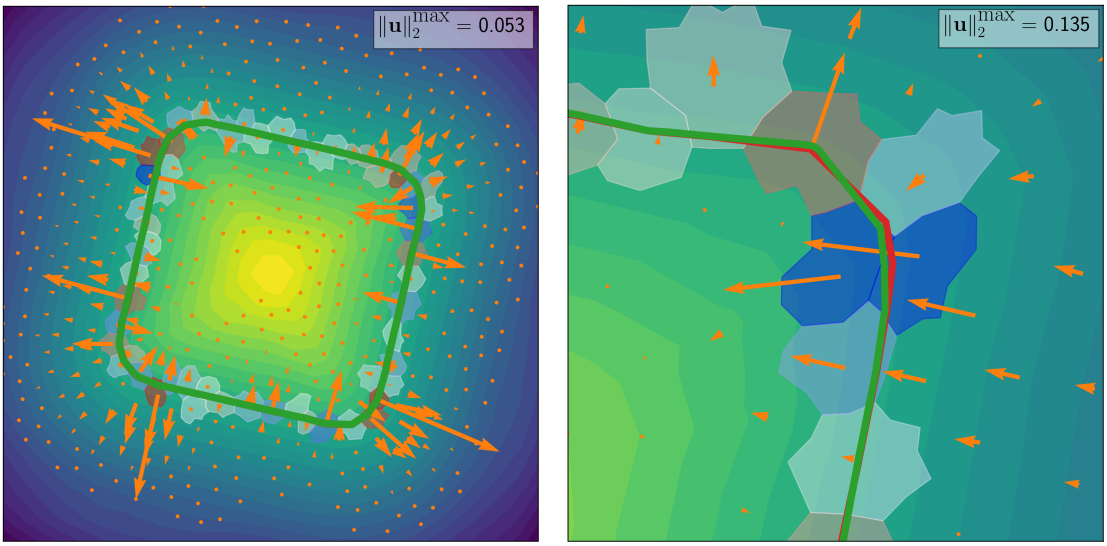
Note that this method does not aim to conserve volume globally. However, by preserving volume locally the total volume loss/gain should also be restored. In Section 7 we will go more into detail about the performance.

6.2. Implementation details

6.2.1. Choices made for the implementation

In this section we discuss some of the implementation details for the VOF-LICLS method. While the proposed method should work theoretically, some choices can be made to improve the numerical behaviour and in this way enhance the performance of the method.

While the original ICLS method could restore volume over an arbitrary number of time steps, this method is not able to do that, since its goal is to restore volume locally. So the correction procedure must be executed in every time step.



(a) Ball-8 (b) Square

Figure 6.3: Correction-velocity of VOF-LICLS method for droplet ball-8 and square. The corrected interface is depicted in green and the interface before correction in red.

By experimentation we found that by choosing $\Delta\tau = \Delta t$ overshoots occur, that is, cells with volume loss will have volume gain after advection with the correction-velocity and vice versa. This results in unstable behaviour of the interface location. Furthermore, not all volume is restored after one pseudo time step. Therefore we choose to execute the procedure iteratively by restoring volume over multiple pseudo time steps. This allows us to use a smaller pseudo time step in order to avoid overshoots and obtain a more stable result. For the pseudo time step $\Delta t = 0.5\Delta\tau$ turns out to be an appropriate choice.

As discussed, the VOF-LICLS method does not guarantee to retain global volume conservation. While in every time step the total volume is preserved considerably well, the accumulation of errors can cause the total volume to slowly diverge from the initial volume. Therefore we choose to advect the level-set with the correction-velocity from the VOF-ICLS method from Section 5 at the end of the correction algorithm. This additional step will make sure global volume conservation is maintained. Since the VOF-ICLS method uses local information and the magnitude of correction is generally small, it is very unlikely that this will worsen the result obtained from the previous VOF-LICLS time steps. This justifies the choice of performing one time step of the VOF-ICLS method at the end of the correction procedure.

The derivation of the correction-velocity assumes that volume loss/gain can only occur at interface elements. Assume that after advection the exact interface is located at the edge of an element K. Since errors occur while advecting, it can happen that the interface 'jumps' to another element. In this way, the volume error of element K is not taken into account. This problem is solved by labelling this element also as an interface cell and by scaling this with scaling factor from Eq. (6.11).

As discussed earlier in this report, the standard Galerkin method for pure advection equations leads to instabilities. Therefore we also tried to advect Eq. (4.10) with the SUPG scheme from Section 3.1.2. However, this resulted in an inaccurate interface and mediocre volume corrections. Therefore we chose to advect Eq. (4.10 with standard scheme. Since correction-advection is executed for only a few pseudo time steps, and since the correction-velocity is very small in magnitude compared to the flow-velocity, this will have no significant effect on the stability of the solution.

6.2.2. Calculating the gradient

For constructing the correction-velocity and reinitializing the level-set field the gradient of ϕ must be computed. However, this is not straightforward for unstructured triangular meshes. We present the differential least-squares method [18], which will be used for computing $\nabla\phi$ and the normal vector $\mathbf{n}$.

The DLS method makes Taylor series expansions of $\tilde{\phi}_i$ for each node $\mathbf{x}_i$ to each neighbour ϕ_k at node $\mathbf{x}_k$. The sum $(\tilde{\phi}_i - \phi_k)^2$ over all adjacent nodes is then minimized in the least-squares sense. In Figure 6.4 the reference node with its adjacent nodes is displayed. The gradient of the level-set field is then obtained with

the L_2 norm minimization [18] by solving the linear system

$$A^\top A\mathbf{x} = A^\top \mathbf{b}, \tag{6.16}$$

with

$$A = \begin{bmatrix} w_{k_1}(x_{k_1} - x_i) & w_{k_1}(x_{k_1} - x_i) \\ \vdots & \vdots \\ w_{k_n}(x_{k_n} - x_i) & w_{k_n}(x_{k_n} - x_i) \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} w_{k_1}(\phi_{k_1} - \phi_i) \\ \vdots \\ w_{k_n}(\phi_{k_n} - \phi_i) \end{bmatrix}, \tag{6.17}$$

where $w_k = 1/\|\mathbf{x}_k - \mathbf{x}_i\|^2$. The solution to Eq. 6.16 yields

$$\mathbf{x} = \begin{bmatrix} \nabla_x \phi_i \\ \nabla_y \phi_i \end{bmatrix}. \tag{6.18}$$

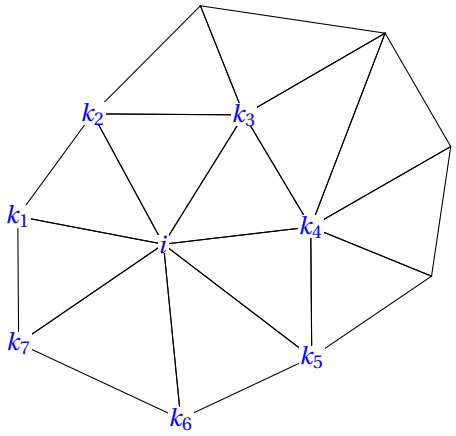


Figure 6.4: Mesh with adjacent nodes $k_1, \ldots, k_7$ to reference node i.

This method gives only first order accurate results. If one desires a better gradient evaluation the geometric least-squares (GLS) method [18] can be used, yielding a second order gradient approximation of the gradient.

6.2.3. Stopping criterium for volume correction procedure

As discussed in Section 6 the volume correction step is an iterative procedure, and therefore it requires a stopping criterium. The choice is made based on experimentation and observation of the results.

The iteration process stops when either a certain threshold value for the maximum interface error $E_\Gamma^{\max}$ is reached or when a maximum number of iterations n_{iter} is reached. When either

$$E_\Gamma^{\max,n} < 10^{-3} \quad \text{or} \quad \left(E_\Gamma^{\max,k-1} - E_\Gamma^{\max,k}\right) < 10^{-5}$$

holds, the procedure stops. Here, $E_\Gamma^{\max,k}$ denotes the maximum error at correction time step k. If this condition is not satisfied after n_{iter} iterations, the procedure also stops. In this thesis we choose $n_{\text{iter}} = 10$.

6.3. Full solution procedure

We summarize the full solution procedure as follows for one iteration at time step n.

1. The VoF field ψ^n is constructed from the LS field ϕ^n.
2. The LS and VoF field are advected simultaneously, resulting in $\tilde{\phi}^{n+1}$ and ψ^{n+1}.
3. The VoF field ψ_ϕ^{n+1} is constructed from the advected LS field $\tilde{\phi}^{n+1}$.
4. The correction-velocity $\mathbf{u}_c$ is constructed from the LS field $\tilde{\phi}^{n+1}$ and the difference of ψ^{n+1} and ψ_ϕ^{n+1}.

5. The LS field $\tilde{\phi}^{n+1}$ is advected with the correction-velocity $\mathbf{u}_c$.
6. Steps 3-5 are repeated, until a threshold value or a maximum number of iterations is reached.
7. We set $\phi^{n+1} = \tilde{\phi}^{n+1}$ and advance to the next time step.

7

Results

7.1. Verifying experiments

In this section we verify the developed method in this thesis by evaluating its performance against the standard LS and ICLS method. This is done by simulating various test cases and computing the errors.

Besides the volume errors introduced in Eq. (5.21) , we will also consider the L_2-norm and the relative L_2-norm to analyse the accuracy of the interface location at $t = T$, as has been done in [20]

$$\begin{aligned} E_1 &= \sqrt{\int_{\tilde{\Omega}} \left(\phi(\mathbf{x},0) - \phi(\mathbf{x},T)\right)^2 \mathrm{d}\Omega}, \\ E_2 &= \frac{E_1}{\sqrt{\int_{\tilde{\Omega}} \left(\phi(\mathbf{x},0)\right)^2 \mathrm{d}\Omega}}, \end{aligned} \tag{7.1}$$

where E_1 denotes L_2-norm and E_2 the relative L_2-norm.

Theoretically, the interface location at $t = T$ should be identical to the initial position for the rotating flow and the reverse-vortex flow, hence $\phi(\mathbf{x},0)$ is the exact solution for $t = T$. During advection the LS field diverges from the initial LS field, and since we focus on the interface position rather than the entire field we will only consider the region near the interface $\tilde{\Omega}$. We take $\tilde{\Omega}$ as the region near the interface with bandwidth $1.5 \times h$ from the interface, with h being the maximum cell width. In Figure 7.4(b) an example of the discrete region for computing the position errors is displayed.

We consider three test cases for comparing VOF-LICLS with the ICLS and standard LS method; simple droplet in rotating flow, Zalesak's disk in rotating flow and circular droplet in reverse vortex flow.

7.1.1. Simple droplet in rotating flow

First we consider a simple droplet ball-8 with radius $r = 0.3$ centred at $(0.15, 0.15)$ in a rotating flow, given by

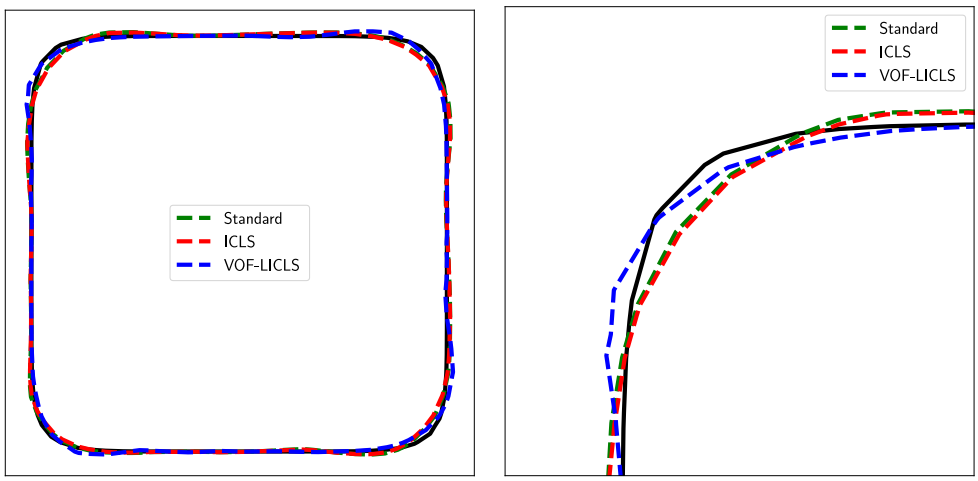
$$\mathbf{u} = \begin{bmatrix} y \\ -x \end{bmatrix}.$$

This test case is simulated until $t = 2\pi$ in domain $\Omega = [-1,1] \times [-1,1]$ with mesh size $n_e = 11145$.

	E_1	E_2
Standard	2.802×10^{-3}	1.578×10^{-1}
ICLS	2.983×10^{-3}	1.680×10^{-1}
VOF-LICLS	2.415×10^{-3}	1.360×10^{-1}

Table 7.1: Position errors for droplet ball-8 in rotating flow.

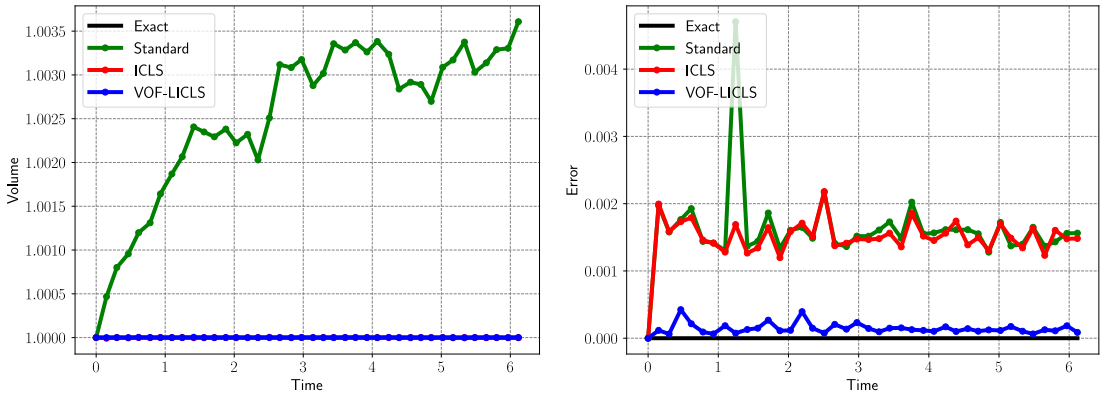
Since this is a simple case, the results (see Figure 7.1) are very similar to each other. The exact interface position is best approached by VOF-LICLS, which is in agreement with the position errors shown in Table 7.1, but the differences are very small.



(a) (b)

Figure 7.1: Result at $t = 2\pi$ of droplet ball-8 in rotating flow of standard, ICLS and VOF-LICLS method.

From the volume errors displayed in Figure 7.2 we see a clear difference in performance. The global volume is well conserved for the ICLS and VOF-LICLS method, however, the local volume conservation of ICLS is almost as bad as the standard LS method. The VOF-LICLS method seems to handle local volume preservation very well, while still maintaining global volume conservation.



(a) Volume (b) Mean error $\overline{E_\Gamma}$

Figure 7.2: Volume and error $\overline{E_\Gamma}$ of droplet ball-8 in rotating flow of standard, ICLS and VOF-LICLS method.

7.1.2. Zalesak's disk in rotating flow

A common benchmark to validate the level-set methods is the Zalesak's disk, which is a slotted disk undergoing a rotating flow [6]. The thin slot and sharp corners make it difficult to be advected accurately, making the effect of the volume correction scheme clear to see.

Zalesak's disk is initialized as follows; the disk is centred at $(0, 0.375)$ in domain $[-1,1] \times [-1,1]$. It has a radius of 0.375 where a slot of length 0.45 and width 0.09375 is taken out. We consider a mesh with $n_e = 11145$. The results are displayed in Figure 7.3 and 7.4.

From Figure 7.4 we find that VOF-LICLS looks the most accurate. For all three methods the corners are smoothed out, for the ICLS method even more than the standard level-set method. While the interface location of VOF-LICLS is the most accurate, its position inside the slot is shifted to the left. This may be caused by applying the volume correction procedure on an inaccurate LS field, which is discussed in Section 7.3.1. Table 7.2 confirms the observation, the position of VOF-LICLS is the most accurate and ICLS is least accurate, however, the difference is small.

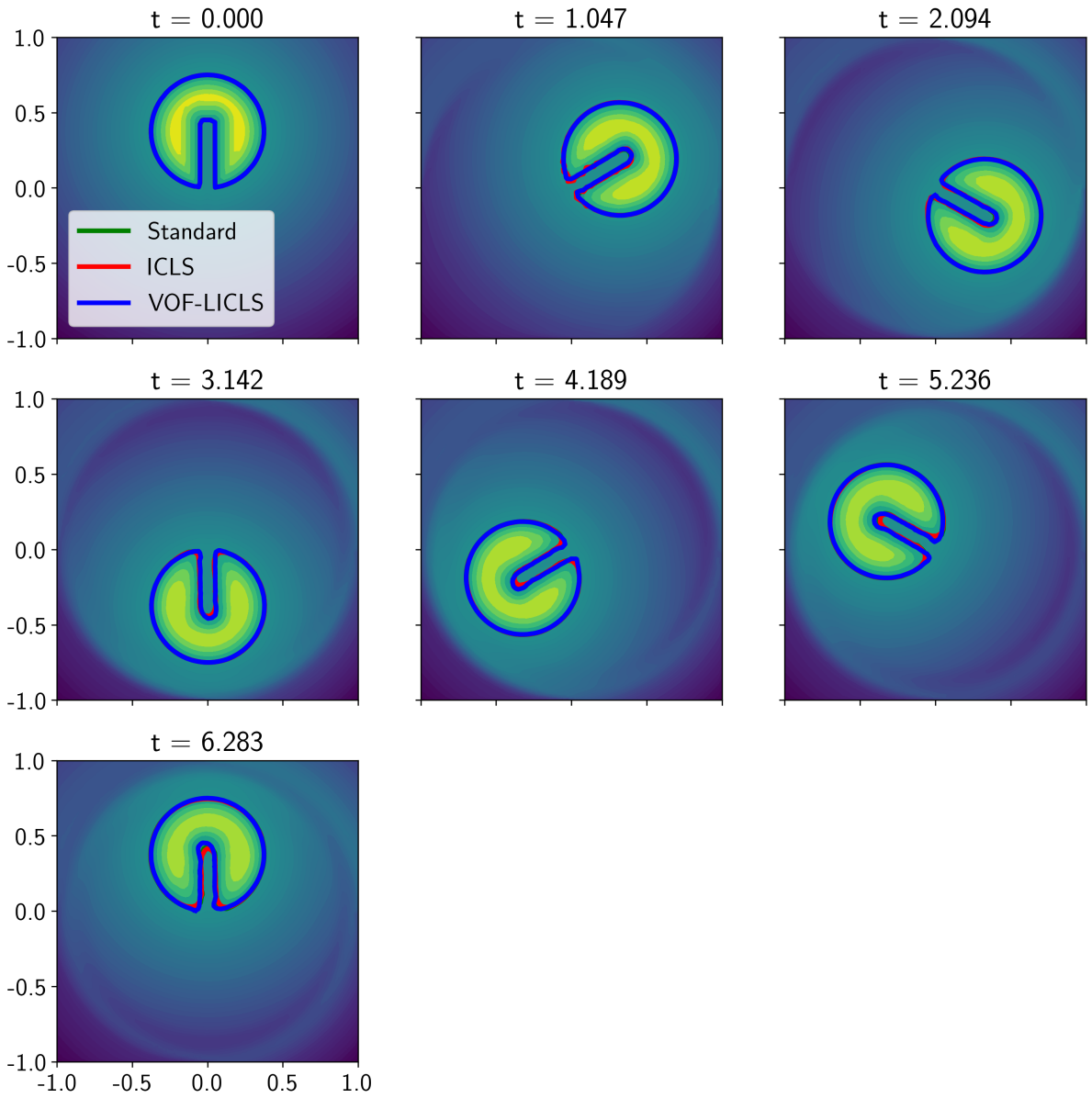
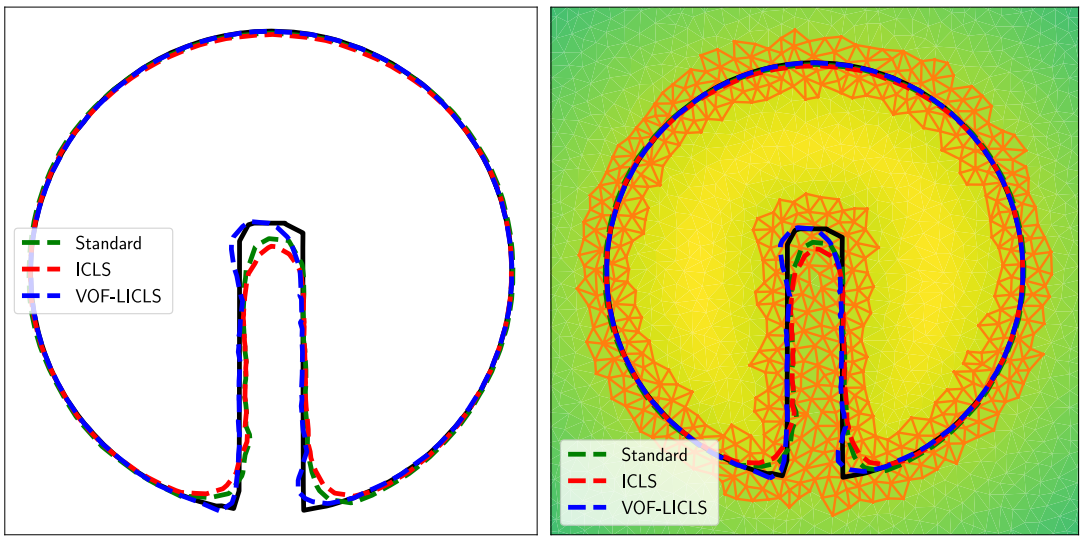


Figure 7.3: Evolution of Zalesak's disk in rotating flow of standard, ICLS and VOF-LICLS method with.

	E_1	E_2
Standard	6.354×10^{-3}	3.058×10^{-1}
ICLS	7.386×10^{-3}	3.555×10^{-1}
VOF-LICLS	5.747×10^{-3}	2.767×10^{-1}

Table 7.2: Position errors for Zalesak's disk.



(a) Result at $t = 2\pi$.

(b) Region $\tilde{\Omega}$ (depicted in orange) for computation E_1 and E_2.

Figure 7.4: Result of Zalesak's disk in rotating flow of standard, ICLS and VOF-LICLS method.

We consider the volume errors over time to investigate their behaviours in more detail, which can be found in Figures 7.5-7.7.

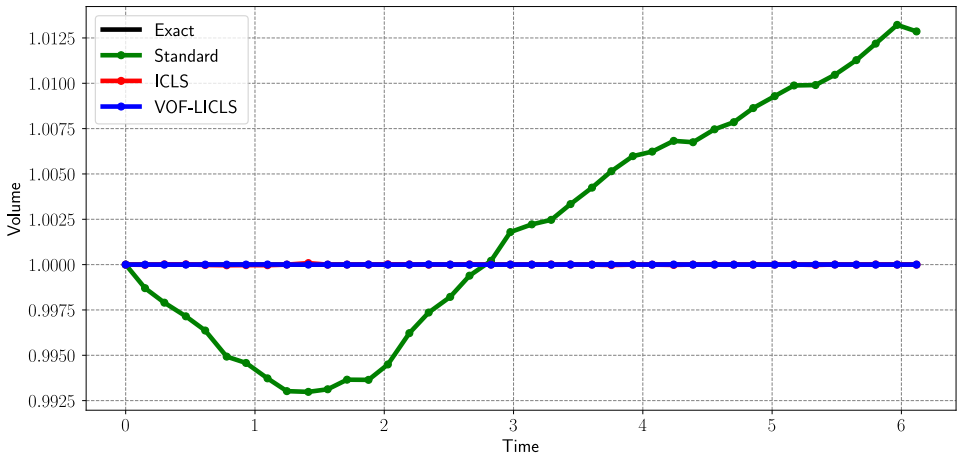


Figure 7.5: Volume of Zalesak's disk in rotating flow of standard, ICLS and VOF-LICLS method. Note that red line is located behind blue line.

	$\sum^{n_{\text{steps}}} \overline{E_\Gamma} / n_{\text{steps}}$	$\sum^{n_{\text{steps}}} E_\Gamma^{\max} / n_{\text{steps}}$
Standard	3.196×10^{-3}	2.967×10^{-2}
ICLS	3.864×10^{-3}	3.379×10^{-2}
VOF-LICLS	2.302×10^{-4}	5.182×10^{-3}

Table 7.3: Average volume errors for Zalesak's disk.

For Zalesak's disk both ICLS and VOF-LICLS manage to conserve volume globally, since during the entire simulation the total scaled volume is almost always 1, as shown in Figure 7.5. Furthermore, from Table 7.3 we see that VOF-LICLS does a good job to preserve volume locally. However, the maximum error at the interface $E_\Gamma^{\max}$ can also be large for VOF-LICLS.

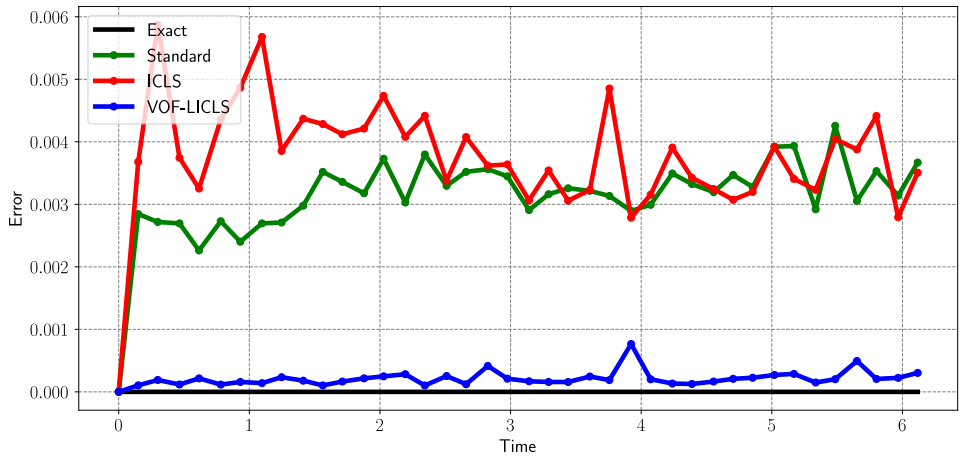


Figure 7.6: Mean error $\overline{E_\Gamma}$ of Zalesak's disk in rotating flow of standard, ICLS and VOF-LICLS method.

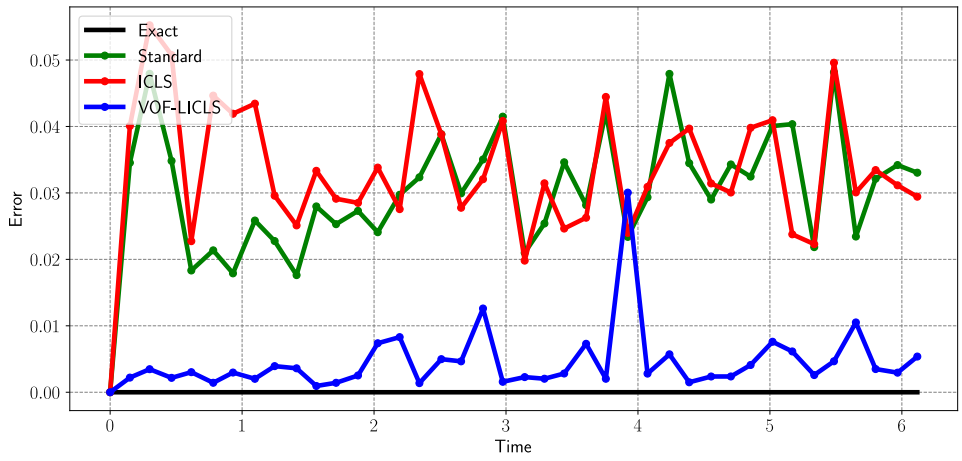


Figure 7.7: Max error $E_\Gamma^{\max}$ of Zalesak's disk in rotating flow of standard, ICLS and VOF-LICLS method.

7.1.3. Circular droplet in reverse vortex flow

To verify the proposed method, a circular droplet in a reverse vortex flow is conducted. The reverse vortex flow is given by [3]

$$\mathbf{u} = \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} -\sin(2\pi y)\sin^2(\pi x)\cos(\pi t/T) \\ \sin(2\pi x)\sin^2(\pi y)\cos(\pi t/T) \end{bmatrix}. \tag{7.2}$$

The initial circular droplet is located in domain $[0,1] \times [0,1]$ with its centre at $(0.5, 0.75)$ and radius 0.25 with mesh size $n_e = 6029$.

During the simulation the droplet is stretched out and gets thinner as it spirals to the centre of the domain. The fluid body gets thinner for larger end times T. At $t = T/2$ the flow reverses and the droplet transforms back to the initial position. This is a good test case, since it tests how well the methods are able to handle deformations. All three methods are tested for the reverse vortex flow with $T = 1$ and $T = 2$. The results at $t = T$ is shown in Figure 7.9 and the evolution of simulation with $T = 2.0$ is displayed in Figure 7.8.

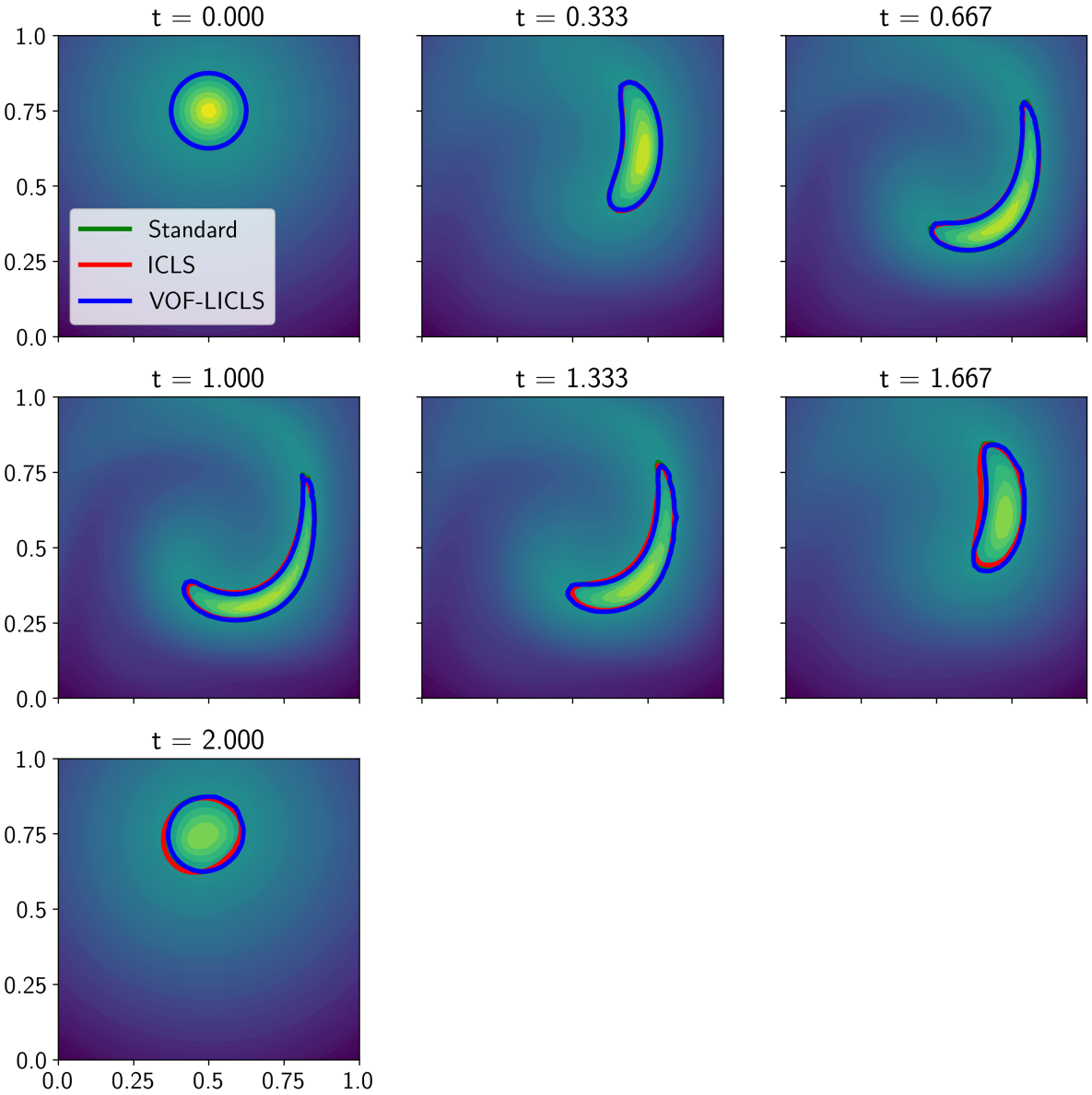
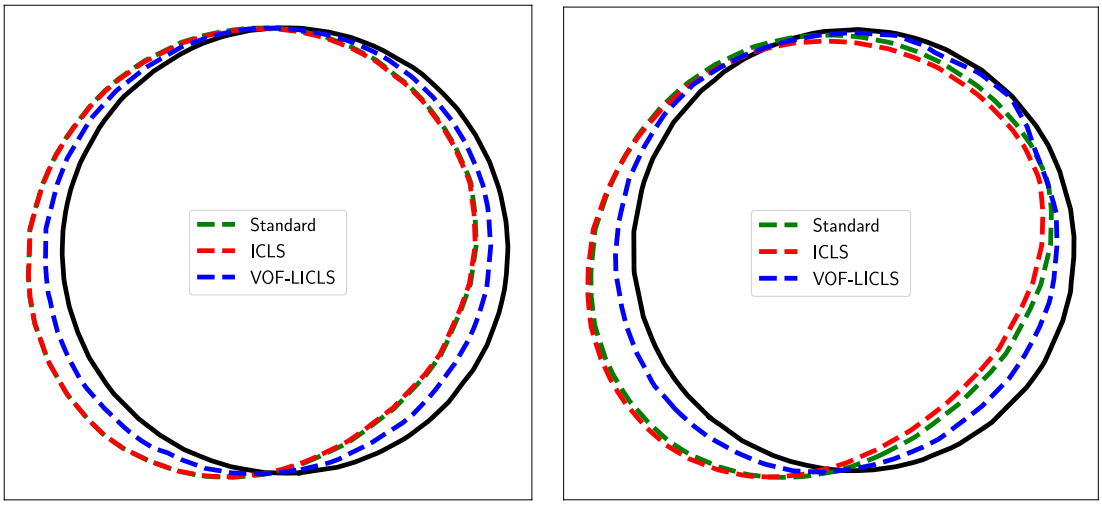


Figure 7.8: Evolution of circular droplet in reverse vortex flow of standard, ICLS and VOF-LICLS method with $T = 2.0$.

From Figure 7.9 and Tables 7.4 and 7.5 we find that the interface position of VOF-LICLS is significantly better than the others. For the previous two test cases, the position errors were similar. So it turns out that



(a) T=1.0 (b) T=2.0

Figure 7.9: Result at $t = T$ of circular droplet in reverse vortex flow of standard, ICLS and VOF-LICLS method with $T = 1.0$ and $T = 2.0$.

	E_1	E_2
Standard	5.107×10^{-3}	7.656×10^{-1}
ICLS	5.518×10^{-3}	7.733×10^{-1}
VOF-LICLS	2.783×10^{-3}	4.172×10^{-1}

Table 7.4: Position errors for reverse vortex with $T = 1.0$.

	E_1	E_2
Standard	6.255×10^{-3}	6.909×10^{-1}
ICLS	7.161×10^{-3}	7.910×10^{-1}
VOF-LICLS	3.246×10^{-3}	3.674×10^{-1}

Table 7.5: Position errors for reverse vortex with $T = 2.0$.

when deformations occur VOF-LICLS outperforms ICLS. Furthermore, we see that the interface at $t = T$ is deformed to an elliptic shape for the standard LS and ICLS method, while for VOF-LICLS the interface remains a circle and is only shifted.

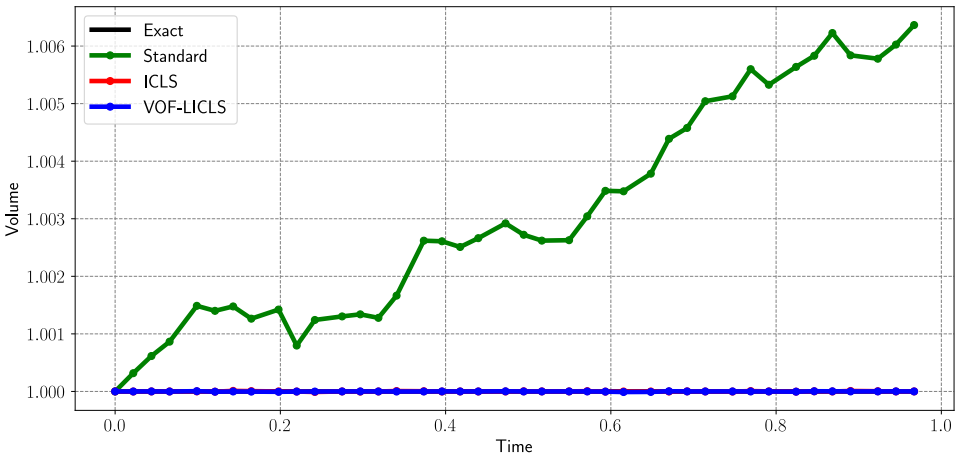


Figure 7.10: Volume of circular droplet in reverse vortex flow of standard, ICLS and VOF-LICLS method with $T = 1.0$. Note that red line is located behind blue line.

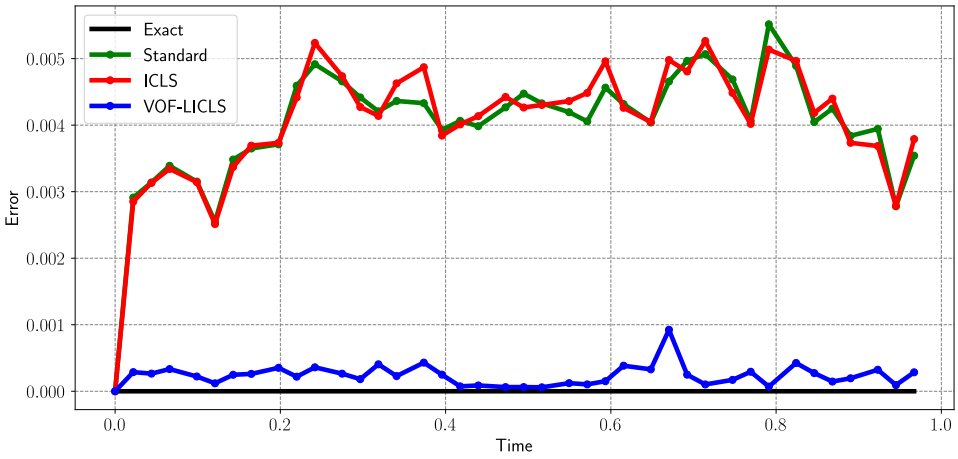


Figure 7.11: Mean error $\overline{E_\Gamma}$ of circular droplet in reverse vortex flow of standard, ICLS and VOF-LICLS method with $T = 1.0$.

	$\sum^{n_{\text{steps}}} \overline{E_\Gamma} / n_{\text{steps}}$	$\sum^{n_{\text{steps}}} E_\Gamma^{\max} / n_{\text{steps}}$
Standard	4.075×10^{-3}	1.446×10^{-2}
ICLS	4.130×10^{-3}	1.451×10^{-2}
VOF-LICLS	2.374×10^{-4}	2.273×10^{-3}

Table 7.6: Average volume errors for circular droplet in reverse vortex flow with $T = 1.0$.

	$\sum^{n_{\text{steps}}} \overline{E_\Gamma} / n_{\text{steps}}$	$\sum^{n_{\text{steps}}} E_\Gamma^{\max} / n_{\text{steps}}$
Standard	4.152×10^{-3}	2.468×10^{-2}
ICLS	4.533×10^{-3}	2.389×10^{-2}
VOF-LICLS	6.961×10^{-4}	1.180×10^{-2}

Table 7.7: Average volume errors for circular droplet in reverse vortex flow with $T = 2.0$.

By studying the volume errors of the simulation with $T = 1$ shown in Figures 7.10-7.12 and Table 7.6 we find the same results as the previous two test cases. However, the reverse vortex case with $T = 2$ is relatively

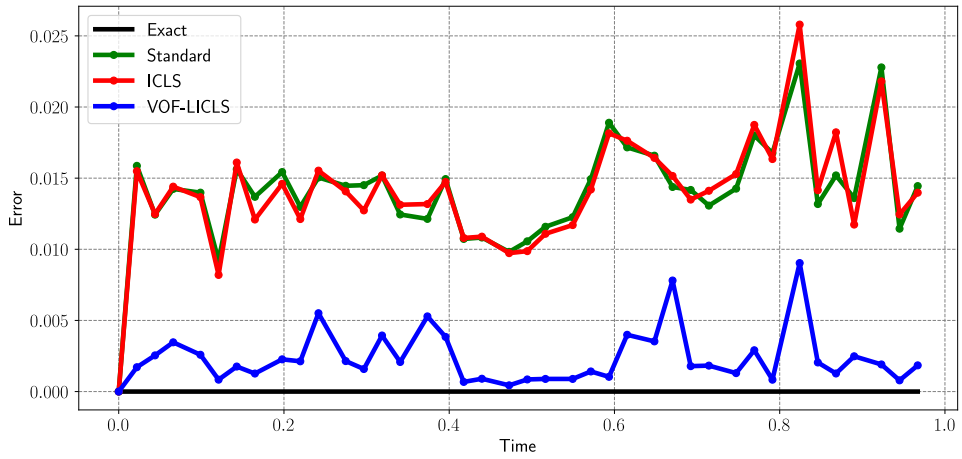


Figure 7.12: Max error $E_\Gamma^{\max}$ of circular droplet in reverse vortex flow of standard, ICLS and VOF-LICLS method with $T = 1.0$.

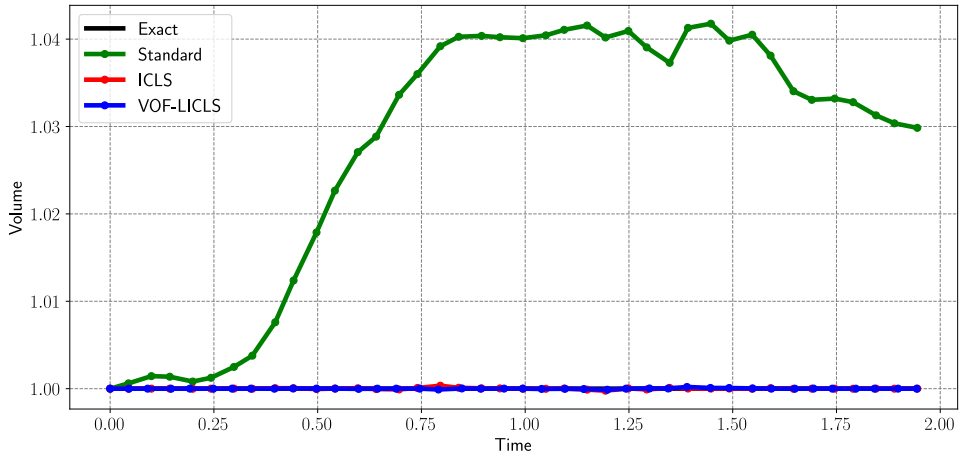


Figure 7.13: Volume of circular droplet in reverse vortex flow of standard, ICLS and VOF-LICLS method with $T = 2.0$.

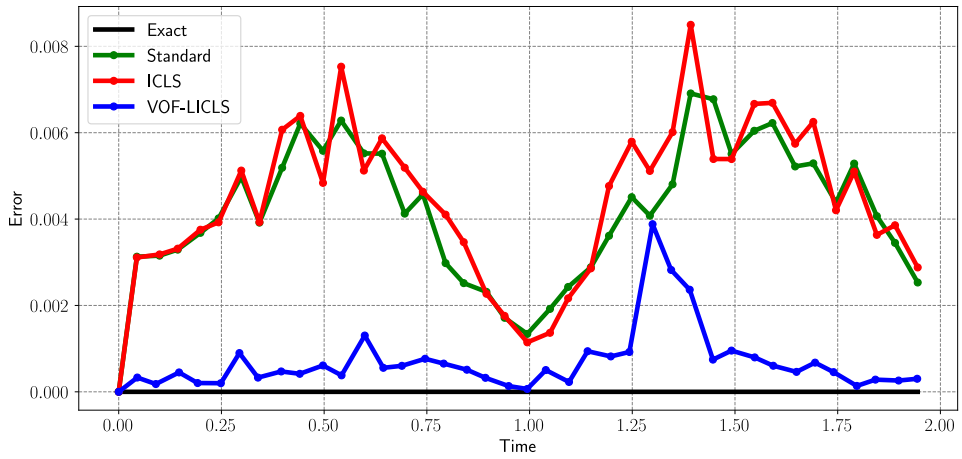


Figure 7.14: Mean error $\overline{E_\Gamma}$ of circular droplet in reverse vortex flow of standard, ICLS and VOF-LICLS method with $T = 2.0$.

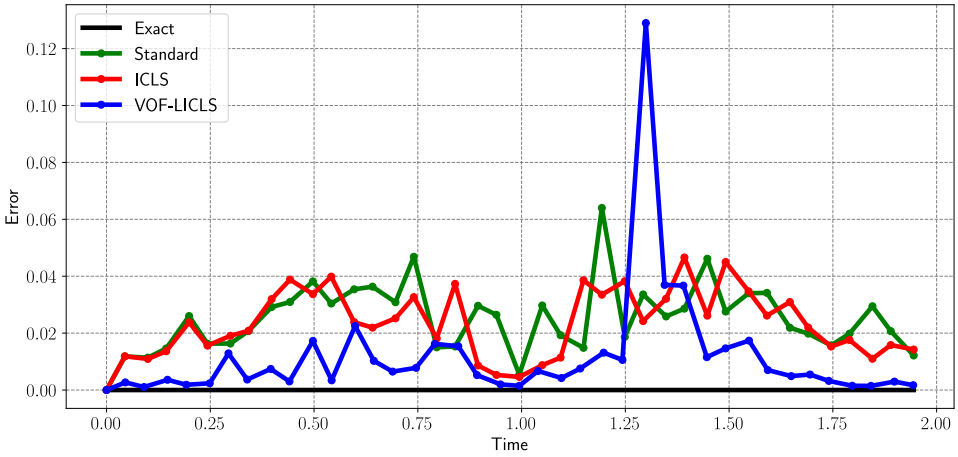


Figure 7.15: Max error $E_\Gamma^{\max}$ of circular droplet in reverse vortex flow of standard, ICLS and VOF-LICLS method with $T = 2.0$.

more difficult for VOF-LICLS as can be seen in Figures 7.13-7.15 and Table 7.7. While still yielding better results, the average maximum error of VOF-LICLS is close to that of ICLS. So VOF-LICLS has more trouble successfully correcting volume when the droplet is thin and sharp-cornered.

7.1.4. Conclusions from verifying experiments

From the test cases we found that ICLS improves the global volume conservation, but slightly worsens the local volume conservation, while VOF-LICLS improves both. For each case VOF-LICLS yielded the best results for preserving volume locally. The difference was less clear for the position errors, but VOF-LICLS also managed to obtain the most accurate interface location. So volume can be corrected accurately, but not always in the right manner, which can result in a slight deformed interface. From the results we can conclude that the proposed method is definitely an improvement to the standard LS method and the ICLS method.

7.2. Error analysis

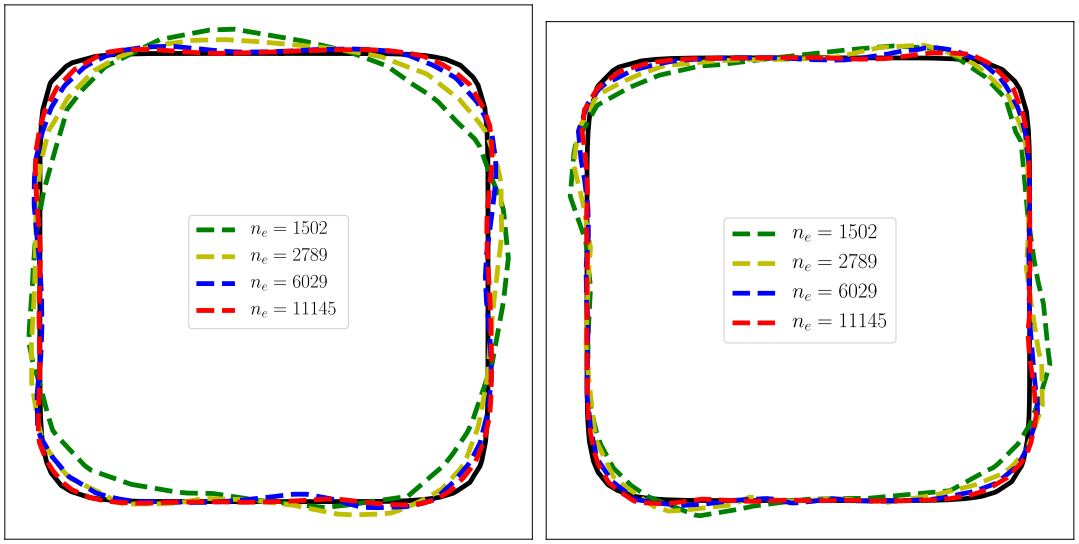
In this section we will investigate the behaviour of VOF-LICLS in more detail by analysing the error. We will study the effect of the mesh size and the number of iterations on the performance of VOF-LICLS. Since the maximum error, defined by Eq. (5.21) is roughly the same for every simulation no matter the mesh size, we will keep the threshold value at 10^{-3}.

7.2.1. Mesh sizes

First we will consider different mesh sizes. A droplet ball-8 with radius 0.3 with its centre located at $(0.15, 0.15)$ in domain $[-1,1] \times [-1,1]$ in a rotating flow is simulated with mesh sizes $n_e = 1502$, $n_e = 2789$, $n_e = 6029$ and $n_e = 11145$, with n_e being the number of elements. For this experiment we choose $n_{\text{iter}} = 10$. We also simulated this for the standard LS method, see Figure 7.16(a) and Table 7.8, from which we can clearly see that a finer mesh leads to more accurate results, which is to be expected. The results of VOF-LICLS are displayed in Figure 7.16(b) and Table 7.9 which also show that increasing the number of elements results in a better performance, so the method is consistent with the expected behaviour. We observe that peaks in interfaces still occur even for finer meshes, which is a drawback of the method. While the errors of VOF-LICLS are smaller, we observe by looking at the relative L_2-norm that increasing the mesh size has relatively less effect for VOF-LICLS than for the standard method. The solution of the LS field converges with around second order accuracy. The convergence order for VOF-LICLS is slightly worse than for LS.

	E_1	E_2	Order
$n_e = 1502$	2.471×10^{-2}	3.218×10^{-1}	-
$n_e = 2789$	1.351×10^{-2}	$2.633 \times 10^{2.4-1}$	1.95
$n_e = 6029$	5.660×10^{-3}	2.028×10^{-1}	2.10
$n_e = 11145$	2.802×10^{-3}	1.578×10^{-1}	2.52

Table 7.8: Position errors for droplet ball-8 in rotating flow for different mesh sizes of standard LS method.



(a) Standard (b) VOF-LICLS

Figure 7.16: Result of droplet ball-8 in rotating flow at $t = 2\pi$ for different mesh sizes of standard LS and VOF-LICLS method.

	E_1	E_2	Order
$n_e = 1502$	1.526×10^{-2}	1.987×10^{-1}	-
$n_e = 2789$	9.326×10^{-3}	1.818×10^{-1}	1.59
$n_e = 6029$	4.366×10^{-3}	1.565×10^{-1}	1.97
$n_e = 11145$	2.415×10^{-3}	1.360×10^{-1}	1.92

Table 7.9: Position errors for droplet ball-8 in rotating flow for different mesh sizes of VOF-LICLS method.

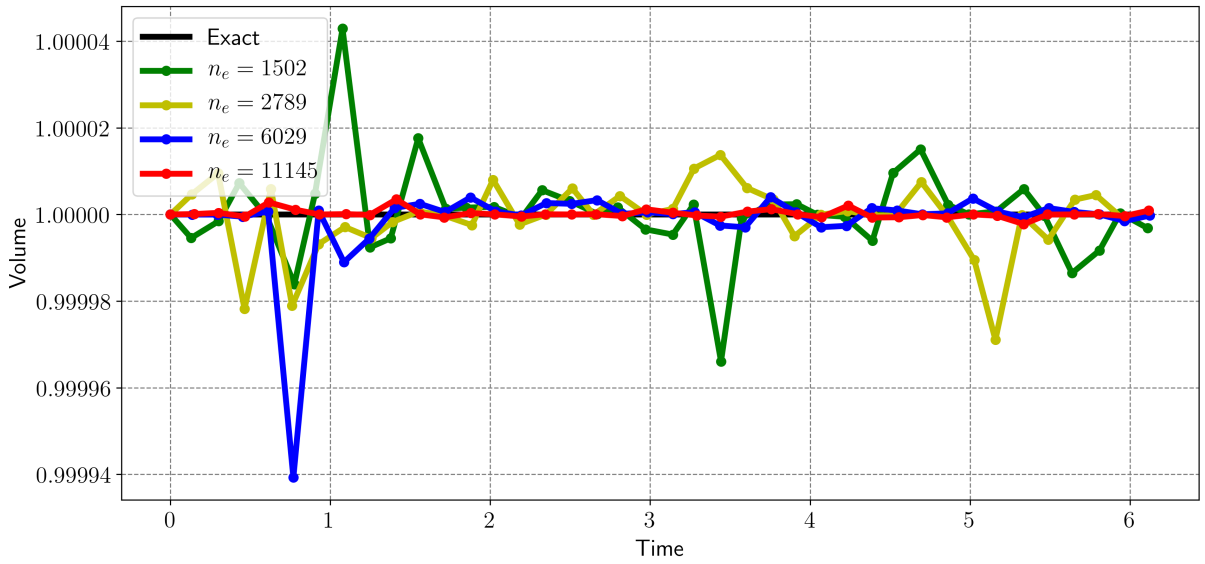


Figure 7.17: Volume for ball-8 in rotating flow for different mesh sizes.

The volume errors are shown in Figures 7.17-7.19 and Table 7.10. The results show that volume is conserved better globally and locally for larger mesh sizes. We also see that the maximum error $E_\Gamma^{\max}$ does not differ much for different mesh sizes, and the average maximum error for $n_e = 6029$ is even larger than for $n_e = 2789$.

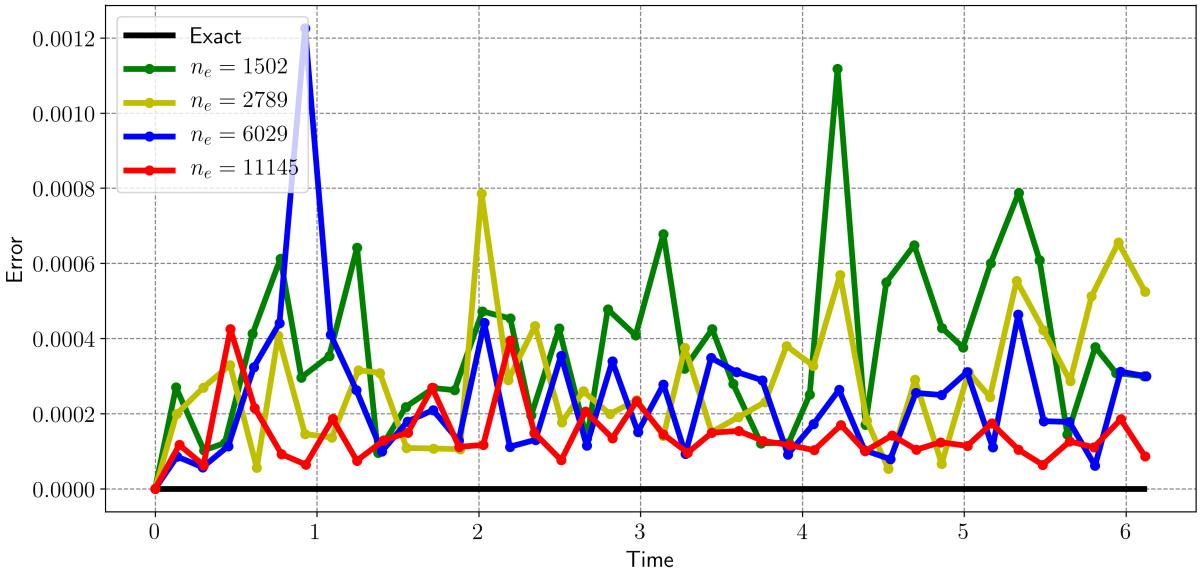


Figure 7.18: Mean error $\overline{E_\Gamma}$ for ball-8 in rotating flow for different mesh sizes.

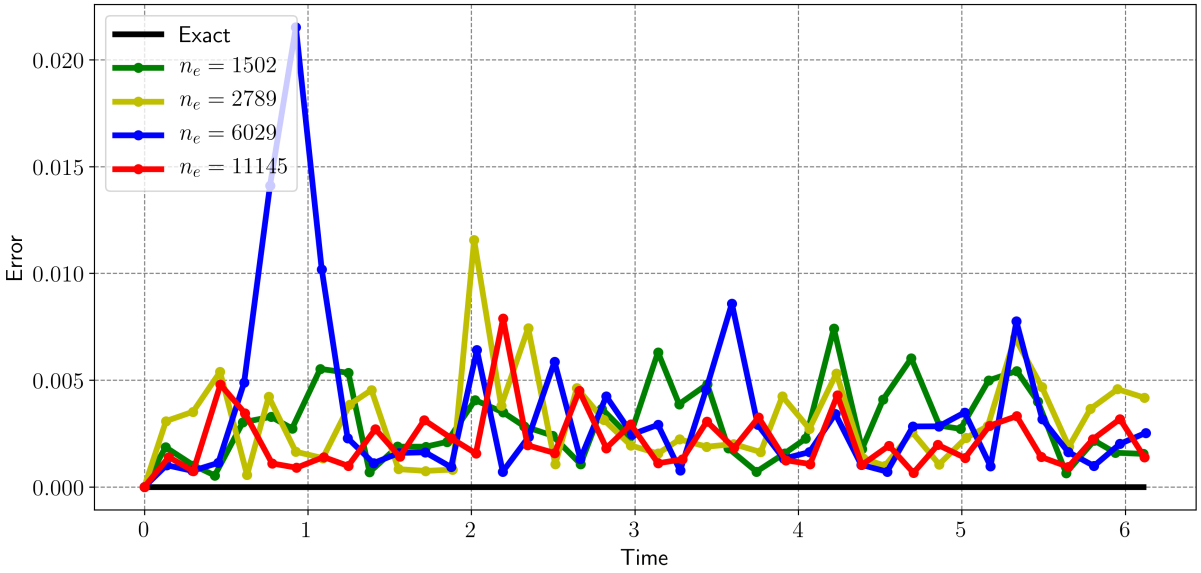


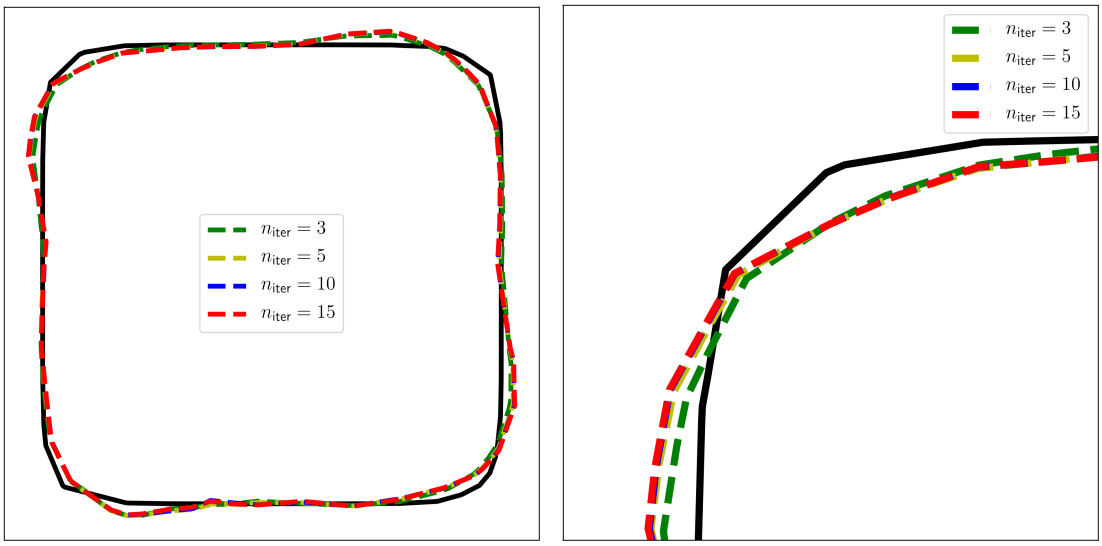
Figure 7.19: Max error $E_\Gamma^{\max}$ for ball-8 in rotating flow for different mesh sizes.

	$\sum^{n_{\text{steps}}} \overline{E_\Gamma} / n_{\text{steps}}$	$\sum^{n_{\text{steps}}} E_\Gamma^{\max} / n_{\text{steps}}$
$n_e = 1502$	4.107×10^{-4}	3.384×10^{-3}
$n_e = 2789$	3.129×10^{-4}	3.047×10^{-3}
$n_e = 6029$	2.464×10^{-4}	3.502×10^{-3}
$n_e = 11145$	1.573×10^{-4}	2.429×10^{-3}

Table 7.10: Average volume errors for different mesh sizes.

7.2.2. Number of iterations

Now we will discuss the effect of the maximum number of iterations on the performance of the developed method in this thesis. The simulation is similar to the simulation described in the previous section, but this time we choose $n_e = 2789$ and vary the maximum number of iterations; $n_{\text{iter}} = 3, 5, 10$ and 15.



(a) Result (b) Zoomed-in result

Figure 7.20: Result of droplet ball-8 in rotating flow at $t = 2\pi$ for number of iterations.

	E_1	E_2
$n_{\text{iter}} = 3$	8.509×10^{-2}	1.659×10^{-1}
$n_{\text{iter}} = 5$	9.240×10^{-3}	1.802×10^{-1}
$n_{\text{iter}} = 10$	9.326×10^{-3}	1.818×10^{-1}
$n_{\text{iter}} = 15$	9.159×10^{-3}	1.786×10^{-1}

Table 7.11: Position errors for droplet ball-8 in rotating flow for different number of iterations.

From Figure 7.20 and Table 7.11 we see a significant difference between $n_{\text{iter}} = 3$ and $n_{\text{iter}} >= 5$. However, there is almost no difference between 5, 10 and 15 iterations. So increasing the number of iteration for $n_{\text{iter}} > 5$ will have very little to no effect on the accuracy of the interface location, and will only increase the computation time. In order to investigate this further, we also consider the volume errors, which can be found in Figures 7.21-7.23 and Table 7.12. These results show that increasing the number of iterations has a positive effect on the local and global volume conservation.

	$\sum^{n_{\text{steps}}} \overline{E_\Gamma} / n_{\text{steps}}$	$\sum^{n_{\text{steps}}} E_\Gamma^{\max} / n_{\text{steps}}$
$n_{\text{iter}} = 3$	8.244×10^{-4}	6.735×10^{-3}
$n_{\text{iter}} = 5$	4.654×10^{-4}	4.677×10^{-3}
$n_{\text{iter}} = 10$	3.129×10^{-4}	3.047×10^{-3}
$n_{\text{iter}} = 15$	2.756×10^{-4}	2.621×10^{-3}

Table 7.12: Average volume errors for different number of iterations.

7.2.3. Conclusions from error analysis

From the error analysis we can conclude that VOF-LICLS is capable of correcting volume quite accurately. In each time step volume loss/gain is locally restored with an average maximum error around 4×10^{-3}. From this observation we would expect an accurate interface location during and at the end of the simulation. However, this is not always the case, since deformations are almost always present, especially near corners.

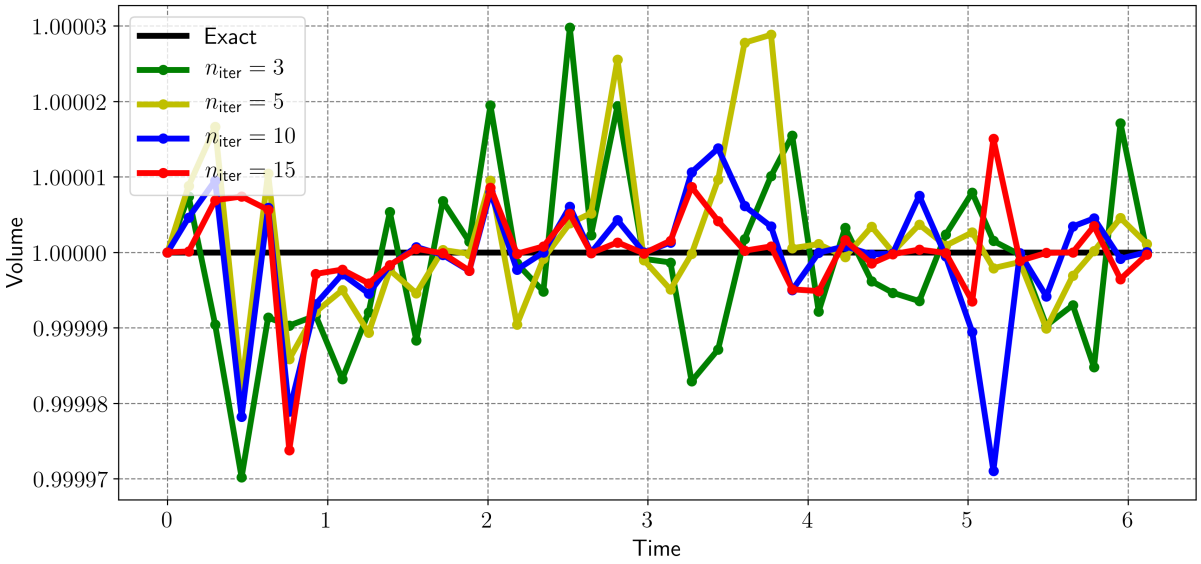


Figure 7.21: Volume for ball-8 in rotating flow for different mesh sizes.

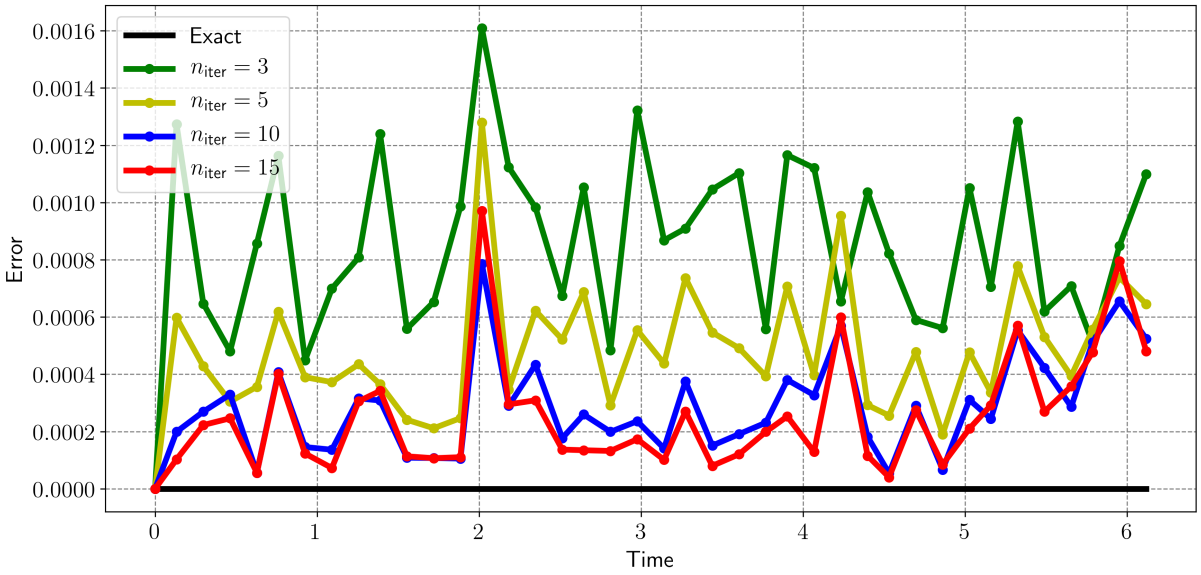


Figure 7.22: Mean error $\overline{E_\Gamma}$ for ball-8 in rotating flow for different mesh sizes.

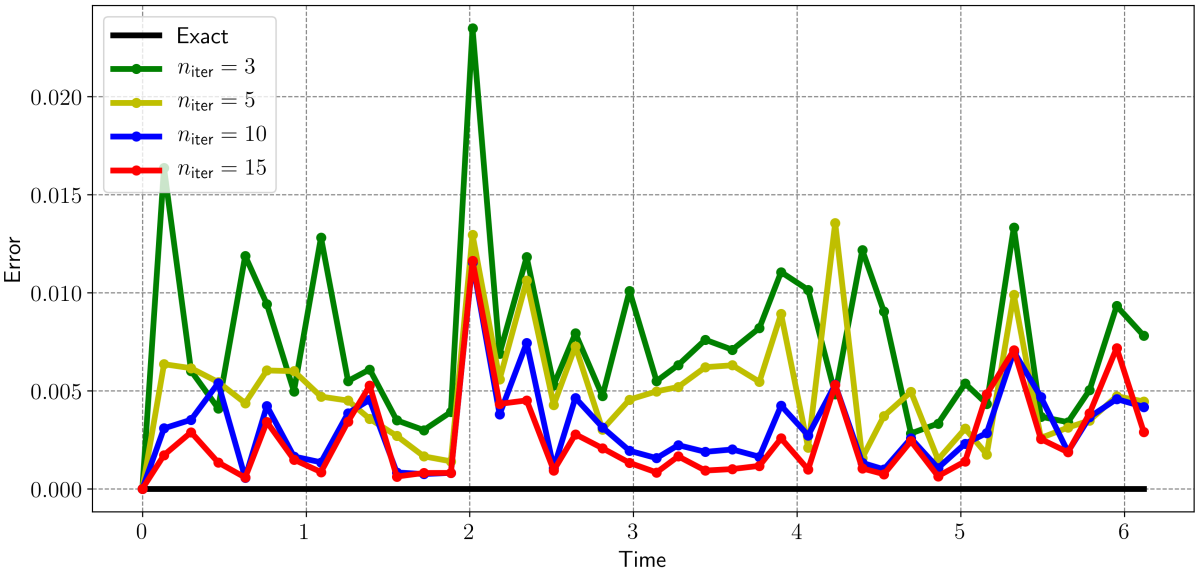


Figure 7.23: Max error $E_{\Gamma}^{\max}$ for ball-8 in rotating flow for different mesh sizes.

We can conclude that the interface is not corrected in a very precise manner, which causes the interface to slowly diverge from the exact solution.

7.3. Observations

7.3.1. Emergence of peaks

One of the disadvantages of the VOF-LICLS method is that peak tends to occur. Corrections are made in the direction of the flow velocity, and while the magnitude of the velocity is analytically determined, the direction is the same as the gradient of the level-set field. This may not always be the best choice, which can result in an incorrect interface position. This phenomena happens especially near corners. In Figure 7.24(a) an example of this situation is given. The emergence of peaks can happen when either the level-set is inaccurate or when the node is near a region with high curvature. Since the gradient of the level-set field is calculated locally, the gradient will be almost perpendicular to the interface. However, this is not always the ideal direction for correcting the volume. We also observed that the emergence of peaks tends to occur at nodes with an outward-pointing velocity surrounded by inward-pointing velocity nodes.

In order to overcome this problem, we propose an averaging procedure. The gradient of ϕ, $\nabla\phi$, is averaged in both x and y-direction, see Figure 7.24(b). The averaging of for an arbitrary node K is executed as follows

$$\overline{\nabla\phi_x}_K = \frac{1}{|\Omega_t|}\int_{\Omega_t} \nabla\phi_x \,\mathrm{d}\Omega, \quad \overline{\nabla\phi_y}_K = \frac{1}{|\Omega_t|}\int_{\Omega_t} \nabla\phi_y \,\mathrm{d}\Omega, \tag{7.3}$$

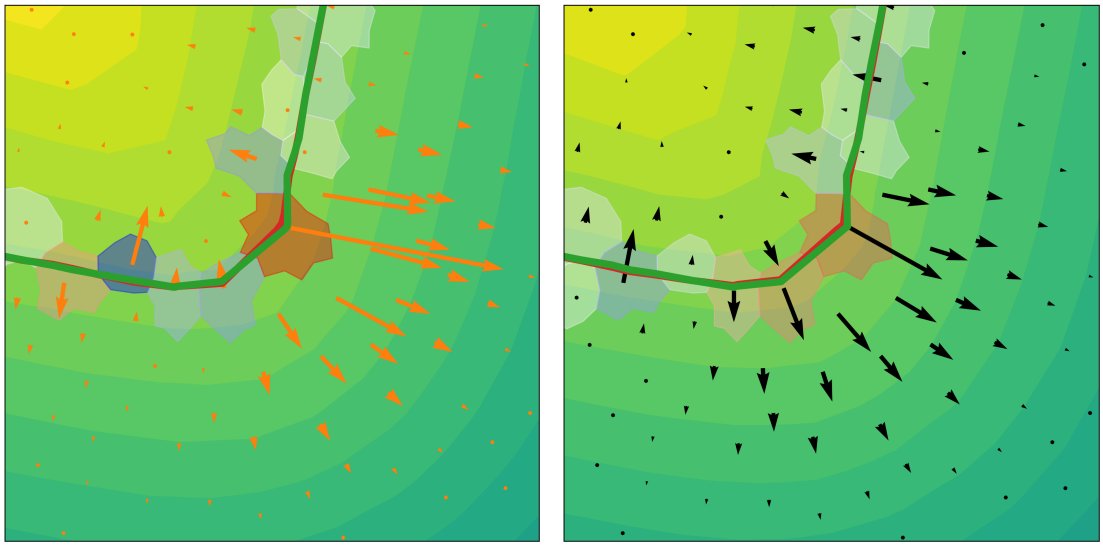
with T the corresponding dual element and Ω_t its control volume, see Figure 7.25.

The result of the simulation with averaging can be seen in Figure 7.26. The interface positions are nearly the same, and the errors indicate that the performance is almost identical. Therefore we choose not to use this in the implementation.

7.3.2. Computational time

The current implementation of VOF-LICLS used in this thesis is not straightforward in contrast to the simplicity of the ICLS method. Construction of the correction-velocity has multiple steps which require a lot of computational power, such as iteratively evaluating a line integral across the interface. In Table 7.13 the computation time of one iteration is shown for the standard, ICLS and VOF-LICLS method for $n_e = 2789, 6029$ and 11145. VOF-LICLS needs a lot more time for one iteration than ICLS. This is because VOF-LICLS uses more than one iteration in each time iteration step. This is a disadvantage of the developed method in this thesis.

There is still room for improvement. For example, construction of the VoF field from the LS field can be executed in parallel, since small computations are executed for cells independently from other cells. Similarly, a lot of subtasks for constructing the correction-velocity field can be executed in parallel. Furthermore, it is



(a) The correction-velocity is pointed in a suboptimal direction, resulting in a peak in the interface.

(b) Correction-velocity where the averaging procedure is applied.

Figure 7.24: Example of the occurrence of a peak in the interface.

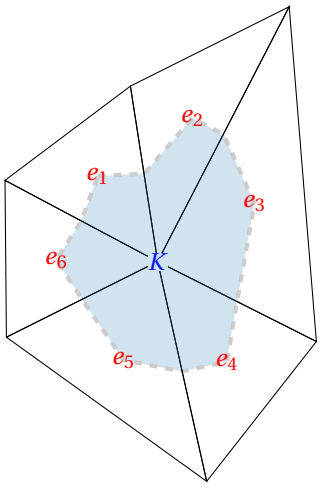
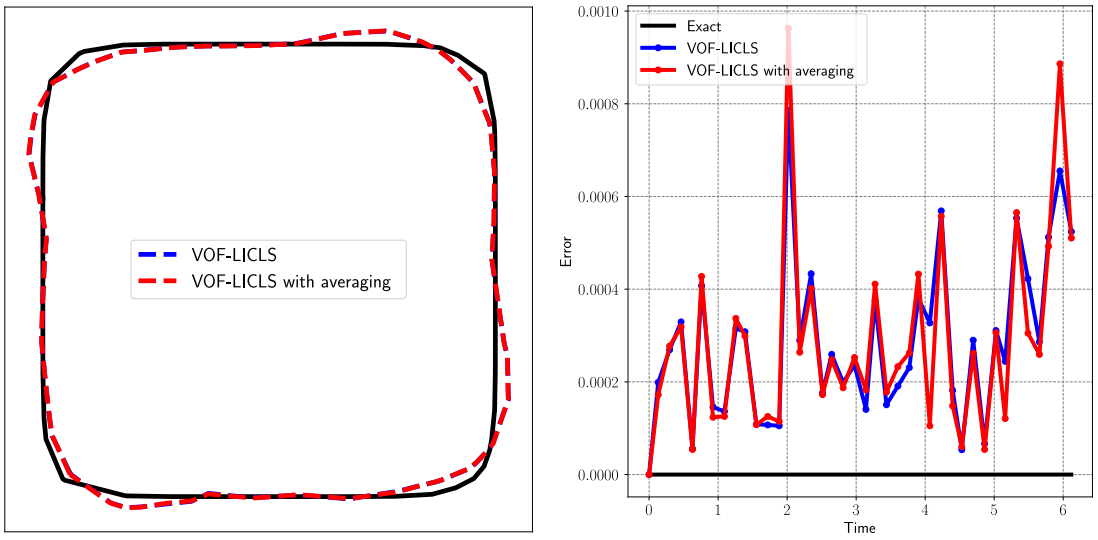


Figure 7.25: Region (depicted in blue) over which the field is integrated for node K.

	$n_e = 2789$	$n_e = 6029$	$n_e = 11145$
Standard	0.006	0.013	0.022
ICLS	0.085	0.232	0.412
VOF-LICLS	0.287	1.108	2.328

Table 7.13: Computation time in minutes of one iteration for different mesh sizes.

sufficient to advect the correction-velocity field only near the interface like the narrow band level-set method [1], since the correction-velocity is almost zero away from the interface. When the correction advection is restricted to the region near the interface, the method will be sped up. However, it will never be as fast as the ICLS by the simple fact that VOF-LICLS requires multiple pseudo time steps to yield accurate results.



(a) Result at $t = 2\pi$

(b) Mean error $\overline{E_\Gamma}$ at interface ($\overline{E_\Gamma}$)

Figure 7.26: Result of simulation with and without averaging for droplet ball-8 in rotating flow.

8

Conclusion and discussion

In this thesis a volume-conserving level-set method for unstructured triangular meshes is proposed. A continuous Finite Element approach is used for discretization. Our focus was to conserve volume and obtain an accurate continuous description of the interface. The developed method is a coupling between VoF and LS, where ideas from MCLS [17] and ICLS [6] are used. A correction-velocity field is constructed based on an analytic equation for advecting the level-set field in order to restore the volume loss.

The standard Galerkin method applied to the level-set equation does not give a stable method, which was shown in [10]. The SUPG method was used to provide stabilization. This was achieved by adding a diffusion term in the streamline direction, which also smoothed out the interface, resulting in a less accurate method. Furthermore, we imposed Dirichlet boundary conditions at the boundary of the domain to prevent instabilities at the inlet boundary. Besides not being inherently volume-conserving, the level-set function also suffers from diverging from a signed-distance function. Reinitialization can be applied to maintain this property. An experiment was carried out to test the performance with reinitialization, but this did not yield a significant improvement.

The ICLS method utilizes an addition advection with an artificial velocity field for achieving global volume conservation. Our aim was to use this approach and couple this with the VoF method in order to obtain a local volume-conserving method. Since ICLS gives us the choice to use an arbitrary speed function, we chose a VoF-dependent speed function. Some extra work was needed to successfully implement this, because the original formulation only allows non-negative functions. The VOF-ICLS method was compared with ICLS in numerical experiments. Both methods yielded similar results; ICLS was slightly better at preserving global volume and VOF-ICLS at conserving volume locally.

An improvement to VOF-ICLS was introduced by proposing a new correction-velocity field, such that the analytic equation, Eq. (4.9), is satisfied for each element. Weights are used to scale the gradient of the smoothed Heaviside function, in order to achieve local volume conservation. In order to prevent the weights from blowing up, an alternative scaling with respect to the corresponding total inward and outward flux was introduced. The developed method in this thesis, VOF-LICLS, uses the structure of MCLS. Volume fractions are constructed from the advected LS field and compared with the advected VoF field. In contrast to MCLS, a dual mesh for VoF advection is used, such that the primal nodes are located inside the dual elements.

The method was tested with numerical experiments to investigate and compare the performance. We focused on the volume fraction errors during the simulation and the position errors at the end of the simulation. The VOF-LICLS method yielded the best result. Both ICLS and VOF-LICLS are able to maintain global volume conservation, but VOF-LICLS outperforms ICLS when it comes to restoring volume locally. Visualizations of the interfaces showed that result of the proposed method was the most accurate. We observed that the edges are smoothed out by all three methods, but the least for VOF-LICLS, however, the interface position in the slot of Zalesak's disk was shifted to the left. From the positions errors for the reverse-vortex case we found that VOF-LICLS was best able to handle deformations.

The method was also tested for different mesh sizes. From the results we observed that the solution converges to the exact solution, however, even for larger mesh sizes, deformations in interfaces were still present. Simulation for different number of iterations were also carried out. From this we found that there was little improvement for choosing more than 5 iterations.

A drawback of the developed method in this thesis is its implementation; it is not simple and computationally expensive. Furthermore, while it manages to restore volume globally and locally quite accurately, the LS field is not always corrected in the right manner, resulting in peaks and inaccurate interface positions, especially near corners.

8.1. Suggestions for further work

For further development of the method, we put forward the following suggestions for further work:

- The level-set method is able to handle merging and splitting of the interface automatically. However, this has not yet been tested in combination with the proposed volume correction procedure. Therefore it needs to be investigated how well the method restores volume for droplet interactions.
- The VOF-LICLS method can be extended to 3D, since both MCLS and ICLS are able to be extended to 3D. Similar procedures were used for the current implementation, so we expect this should be possible.

Development of a more efficient and more accurate algorithm for the VOF-LICLS method is needed. In order to successfully use the proposed we suggest to consider the following things:

- The computational cost needs to be reduced, since the current implementation is computationally expensive. In this thesis we did not focus on providing a fast working algorithm, therefore we believe that the speed of the method can be enhanced by improving the implementation. Furthermore, simple subtasks can be executed parallel. And for the volume correction procedure the narrow-band level-set method can be utilized to speed up the correction-advection.
- An extension needs to be added to prevent the emergence of peaks, for example by modifying the correction-velocity field. In Section 7.3.1 we tried to solve this by introducing an averaging procedure. However, this did not give the desired result.

Bibliography

[1] David Adalsteinsson and James A. Sethian. A fast level set method for propagating interfaces. *Journal of Computational Physics*, 118(2):269 – 277, 1995. ISSN 0021-9991. doi: https://doi.org/10.1006/jcph.1995.1098. URL http://www.sciencedirect.com/science/article/pii/S0021999185710984.

[2] Erik Burman. Consistent supg-method for transient transport problems: Stability and convergence. *Computer Methods in Applied Mechanics and Engineering*, 199(17):1114 – 1123, 2010. ISSN 0045-7825. doi: https://doi.org/10.1016/j.cma.2009.11.023. URL http://www.sciencedirect.com/science/article/pii/S0045782509003983.

[3] Zhizhu Cao, Dongliang Sun, Bo Yu, and Jinjia Wei. A coupled volume-of-fluid and level set (voset) method based on remapping algorithm for unstructured triangular grids. *International Journal of Heat and Mass Transfer*, 111:232 – 245, 2017. ISSN 0017-9310. doi: https://doi.org/10.1016/j.ijheatmasstransfer.2017.03.096. URL http://www.sciencedirect.com/science/article/pii/S0017931016331672.

[4] Dennis den Ouden-van der Horst and Matthias Möller. *Volume-Preserving Continuous Galerkin Level-Set Approach for Linear Finite Elements*. Delft University of Technology, Netherlands, 2 2020.

[5] Vít Dolejsí and Miloslav Feistauer. *Discontinuous Galerkin Method: Analysis and Applications to Compressible Flow*. Springer Publishing Company, Incorporated, 1st edition, 2015. ISBN 3319192663, 9783319192666.

[6] Zhouyang Ge, Jean-Christophe Loiseau, Outi Tammisola, and Luca Brandt. An efficient mass-preserving interface-correction level set/ghost fluid method for droplet suspensions under depletion forces. *Journal of Computational Physics*, 353:435 – 459, 2018. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2017.10.046. URL http://www.sciencedirect.com/science/article/pii/S0021999117308136.

[7] Sven Gross and Arnold Reusken. *Numerical Methods for Two-phase Incompressible Flows*. Springer, Berlin, Heidelberg, 1st edition, 2011. ISBN 978-3-642-19686-7.

[8] Thomas J.R. Hughes, Leopoldo P. Franca, and Gregory M. Hulbert. A new finite element formulation for computational fluid dynamics: Viii. the galerkin/least-squares method for advective-diffusive equations. *Computer Methods in Applied Mechanics and Engineering*, 73(2):173 – 189, 1989. ISSN 0045-7825. doi: https://doi.org/10.1016/0045-7825(89)90111-4. URL http://www.sciencedirect.com/science/article/pii/0045782589901114.

[9] Deniz Kennedy. Level set methods for two-phase flows with fem. Master's thesis, Uppsala University, Department of Information Technology, 2014.

[10] Arthur Kerst. A review of level-set methods for two-phase flows. *Repository TU Delft*, 2020.

[11] Kong Ling, Shuai Zhang, Peng-Zhan Wu, Si-Yuan Yang, and Wen-Quan Tao. A coupled volume-of-fluid and level-set method (voset) for capturing interface of two-phase flows in arbitrary polygon grid. *International Journal of Heat and Mass Transfer*, 143:118565, 2019. ISSN 0017-9310. doi: https://doi.org/10.1016/j.ijheatmasstransfer.2019.118565. URL http://www.sciencedirect.com/science/article/pii/S0017931019327796.

[12] Eva Loch and Arnold Reusken. The level set method for capturing interfaces with applications in two-phase flow problems. In *PhD Thesis, RWTH Aachen*, 2013.

[13] Frank Losasso, Ronald Fedkiw, and Stanley Osher. Spatially adaptive techniques for level set methods and incompressible flow. *Computers & Fluids*, 35(10):995 – 1010, 2006. ISSN 0045-7930. doi: https://doi.org/10.1016/j.compfluid.2005.01.006. URL http://www.sciencedirect.com/science/article/pii/S004579300500174X.

[14] Emilie Marchandise, Jean-François Remacle, and Nicolas Chevaugeon. A quadrature-free discontinuous galerkin method for the level set equation. *Journal of Computational Physics*, 212(1):338 – 357, 2006. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2005.07.006. URL http://www.sciencedirect.com/science/article/pii/S0021999105003244.

[15] Hashem M. Mourad, John Dolbow, and Krishna Garikipati. An assumed-gradient finite element method for the level set equation. *International Journal for Numerical Methods in Engineering*, 64(8):1009–1032, 2005. doi: 10.1002/nme.1395. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/nme.1395.

[16] Daniele A. Di Pietro, Stefania Lo Forte, and Nicola Parolini. Mass preserving finite element implementations of the level set method. *Applied Numerical Mathematics*, 56(9):1179 – 1195, 2006. ISSN 0168-9274. doi: https://doi.org/10.1016/j.apnum.2006.03.003. URL http://www.sciencedirect.com/science/article/pii/S0168927406000432. Numerical Methods for Viscosity Solutions and Applications.

[17] F. Raees, D. R. van der Heul, and C. Vuik. A mass-conserving level-set method for simulation of multiphase flow in geometrically complicated domains. *International Journal for Numerical Methods in Fluids*, 81(7):399–425, 2016. doi: 10.1002/fld.4188. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/fld.4188.

[18] Khosro Shahbazi, Marius Paraschivoiu, and Javad Mostaghimi. Second order accurate volume tracking based on remapping for triangular meshes. *Journal of Computational Physics*, 188(1):100 – 122, 2003. ISSN 0021-9991. doi: https://doi.org/10.1016/S0021-9991(03)00156-6. URL http://www.sciencedirect.com/science/article/pii/S0021999103001566.

[19] Mark Sussman and Elbridge Gerry Puckett. A coupled level set and volume-of-fluid method for computing 3d and axisymmetric incompressible two-phase flows. *Journal of Computational Physics*, 162(2):301 – 337, 2000. ISSN 0021-9991. doi: https://doi.org/10.1006/jcph.2000.6537. URL http://www.sciencedirect.com/science/article/pii/S0021999100965379.

[20] Mamadou Kabirou Touré and Azzeddine Soulaimani. Stabilized finite element methods for solving the level set equation without reinitialization. *Computers & Mathematics with Applications*, 71(8):1602 – 1623, 2016. ISSN 0898-1221. doi: https://doi.org/10.1016/j.camwa.2016.02.028. URL http://www.sciencedirect.com/science/article/pii/S0898122116300827.

[21] S. P. van der Pijl, A. Segal, C. Vuik, and P. Wesseling. A mass-conserving level-set method for modelling of multi-phase flows. *International Journal for Numerical Methods in Fluids*, 47(4):339–361, 2005. doi: 10.1002/fld.817. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/fld.817.

[22] Xiaofeng Yang, Ashley J. James, John Lowengrub, Xiaoming Zheng, and Vittorio Cristini. An adaptive coupled level-set/volume-of-fluid interface capturing method for unstructured triangular grids. *Journal of Computational Physics*, 217(2):364 – 394, 2006. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2006.01.007. URL http://www.sciencedirect.com/science/article/pii/S0021999106000143.

[23] C.H. Yu, Z.T. Ye, Tony W.H. Sheu, Y.T. Lin, and X.Z. Zhao. An improved interface preserving level set method for simulating three dimensional rising bubble. *International Journal of Heat and Mass Transfer*, 103:753 – 772, 2016. ISSN 0017-9310. doi: https://doi.org/10.1016/j.ijheatmasstransfer.2016.07.096. URL http://www.sciencedirect.com/science/article/pii/S001793101630833X.

[24] Lanhao Zhao, Jia Mao, Xin Bai, Xiaoqing Liu, Tongchun Li, and J.J.R. Williams. Finite element implementation of an improved conservative level set method for two-phase flow. *Computers & Fluids*, 100:138 – 154, 2014. ISSN 0045-7930. doi: https://doi.org/10.1016/j.compfluid.2014.04.027. URL http://www.sciencedirect.com/science/article/pii/S0045793014001777.

A

Level-set method

A.1. Derivation of level-set method

In this part we will give the derivation of the level-set equation, Eq. (2.2). Consider a point $\mathbf{x}(t)$ on the interface $\Gamma(t)$ where $\phi = 0$. Now assume we have $\phi = \phi(\mathbf{x}(t), t)$ and consider the equation

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

Differentiating this equation with respect to t yields

$$\frac{\partial \phi}{\partial t} + \nabla\phi \cdot \frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = 0.$$

Under the assumption that $\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{u}$, we get

$$\frac{\partial \phi}{\partial t} + \nabla\phi \cdot \mathbf{u} = 0 \quad \text{on } \Gamma(t).$$

This can be generalized to the entire domain Ω resulting in Eq. (2.2).

A.2. Spatial discretization

A.2.1. Spatial discretization with standard Galerkin

Consider the level-set equation, Eq. (2.2), the weak form is obtained by multiplying with a test function $v \in V_h^k$. This gives

$$\int_\Omega \frac{\partial \phi}{\partial t} v \,\mathrm{d}\Omega + \int_\Omega \left(\mathbf{u} \cdot \nabla\phi\right) v \,\mathrm{d}v = 0, \quad \forall v \in V_h^k. \tag{A.1}$$

With Galerkin's method the following approximation is introduced

$$\phi(\mathbf{x}, t) \approx \phi^n(\mathbf{x}, t) = \sum_{j=1}^{n} c_j(t) v_j(\mathbf{x}), \tag{A.2}$$

where $v_j(\mathbf{x})$ for $j = 1, \ldots, n$ are piecewise polynomials with property $v_j(\mathbf{x}_i) = \delta_{ij}$ and $\sum_j v_j(\mathbf{x}) = 1$.

Substituting this into Eq. (3.2) yields

$$\frac{\mathrm{d}}{\mathrm{d}t} \sum_{j=1}^{n} c_j(t) \int_\Omega v_j v_i \,\mathrm{d}\Omega = \sum_{j=1}^{n} -c_j(t) \int_\Omega \left(\mathbf{u} \cdot \nabla v_j\right) v_i \,\mathrm{d}\Omega, \quad i = 1, \ldots, n, \tag{A.3}$$

$$\frac{\mathrm{d}}{\mathrm{d}t} \sum_{j=1}^{n} M_{ij} c_j(t) = \sum_{j=1}^{n} S_{ij} c_j(t), \quad i = 1, \ldots, n, \tag{A.4}$$

with

$$\begin{aligned} M_{ij} &= \int_\Omega v_j v_i \,\mathrm{d}\Omega \\ &= \sum_{p=1}^{n_T} \int_{e_p} v_j v_i \,\mathrm{d}\Omega \\ &= \sum_{p=1}^{n_T} M_{ij}^{e_p}, \end{aligned} \tag{A.5}$$

and

$$\begin{aligned} S_{ij} &= -\int_\Omega \left(\mathbf{u}\cdot\nabla v_j\right) v_i \,\mathrm{d}\Omega \\ &= \sum_{p=1}^{n_T} -\int_{e_p} \left(\mathbf{u}\cdot\nabla v_j\right) v_i \,\mathrm{d}\Omega \\ &= \sum_{p=1}^{n_T} S_{ij}^{e_p}, \end{aligned} \tag{A.6}$$

where e_p denotes a triangular element with vertices $\mathbf{x}_{p_1}$, $\mathbf{x}_{p_2}$ and $\mathbf{x}_{p_3}$, and n_T the number of elements.

For the next part the formula of Holand and Bell is used. Let e be a triangle with vertices $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}_3$ and v_1, v_2 and v_3 linear on e, and let $m_1, m_2, m_3 \in \mathbb{N}$, then

$$\int_e v_1^{m_1} v_2^{m_2} v_3^{m_3} \,\mathrm{d}\Omega = \frac{|\Delta e| m_1! m_2! m_3!}{(2+m_1+m_2+m_3)!}, \tag{A.7}$$

where the area of e is equal to $|\Delta e|/2$ with

$$|\Delta e| = \left\| \begin{matrix} x_3 - x_1 & x_2 - x_1 \\ y_3 - y_1 & y_2 - y_1 \end{matrix} \right\| = \left\| \begin{matrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{matrix} \right\|.$$

Then with the Holand and Bell formula we find

$$\begin{aligned} M_{ij}^{e_p} = \int_{e_p} v_j v_i \,\mathrm{d}\Omega &\overset{\text{H-B}}{=} \begin{cases} \dfrac{|\Delta e_p|}{24} & i \neq j \\ \dfrac{|\Delta e_p|}{12} & i = j \end{cases} \\ &= \frac{1}{24}(1+\delta_{ij})|\Delta e_p|, \quad i,j \in \{p_1,p_2,p_3\}. \end{aligned} \tag{A.8}$$

So the element mass matrix of e_p is given by

$$M^{e_p} = \frac{|\Delta e_p|}{24} \begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix}.$$

For the piecewise polynomial $v_j(\mathbf{x})$ we can write $\phi_j(\mathbf{x}) = \alpha_j + \beta_j x + \gamma_j y$. Using $\mathbf{u} = \begin{bmatrix} u_1 & u_2 \end{bmatrix}^\top$, we can write

$$\begin{aligned} S_{ij}^{e_p} &= -\int_{e_p} \left(\mathbf{u}\cdot\nabla v_j\right) v_i \,\mathrm{d}\Omega \\ &= -\left(\beta_j \int_{e_p} u_1 v_i \,\mathrm{d}\Omega + \gamma_j \int_{e_p} u_2 v_i \,\mathrm{d}\Omega\right). \end{aligned} \tag{A.9}$$

We evaluate the first integral using the Newton-Cotes quadrature rule

$$\begin{aligned} \int_{e_p} u_1 v_i \,\mathrm{d}\Omega &\overset{\text{N-C}}{=} \frac{|\Delta e_p|}{6} \sum_{l\in\{p_1,p_2,p_3\}} u_1(\mathbf{x}_l) v_i(\mathbf{x}_l) \\ &= \frac{|\Delta e_p|}{6} \sum_{l\in\{p_1,p_2,p_3\}} u_1(\mathbf{x}_l) \delta_{il} \\ &= \frac{|\Delta e_p|}{6} u_1(\mathbf{x}_i), \quad i \in \{p_1,p_2,p_3\}. \end{aligned}$$

Hence, for $S_{ij}^{e_p}$ we have

$$S_{ij}^{e_p} = -\frac{|\Delta e_p|}{6}\Big(\beta_j u_1(\mathbf{x}_i) + \gamma_j u_2(\mathbf{x}_i)\Big), \quad i,j \in \{p_1,p_2,p_3\}. \tag{A.10}$$

So the element stiffness matrix of e_p is given by

$$\begin{aligned} S^{e_p} &= -\frac{|\Delta e_p|}{6}\begin{bmatrix} [\beta_1\ \gamma_1]^\top\cdot\mathbf{u}(\mathbf{x}_1) & [\beta_2\ \gamma_2]^\top\cdot\mathbf{u}(\mathbf{x}_1) & [\beta_3\ \gamma_3]^\top\cdot\mathbf{u}(\mathbf{x}_1)\\ [\beta_1\ \gamma_1]^\top\cdot\mathbf{u}(\mathbf{x}_2) & [\beta_2\ \gamma_2]^\top\cdot\mathbf{u}(\mathbf{x}_2) & [\beta_3\ \gamma_3]^\top\cdot\mathbf{u}(\mathbf{x}_2)\\ [\beta_1\ \gamma_1]^\top\cdot\mathbf{u}(\mathbf{x}_3) & [\beta_2\ \gamma_2]^\top\cdot\mathbf{u}(\mathbf{x}_3) & [\beta_3\ \gamma_3]^\top\cdot\mathbf{u}(\mathbf{x}_3)\end{bmatrix}\\ &= -\frac{|\Delta e_p|}{6}\left(\begin{bmatrix} u_1(\mathbf{x}_1)\\ u_1(\mathbf{x}_2)\\ u_1(\mathbf{x}_3)\end{bmatrix}\begin{bmatrix}\beta_1 & \beta_2 & \beta_3\end{bmatrix} + \begin{bmatrix} u_2(\mathbf{x}_1)\\ u_2(\mathbf{x}_2)\\ u_2(\mathbf{x}_3)\end{bmatrix}\begin{bmatrix}\gamma_1 & \gamma_2 & \gamma_3\end{bmatrix}\right). \end{aligned}$$

Finally, the semi-discrete system is obtained

$$M\frac{\mathrm{d}\mathbf{c}}{\mathrm{d}t} = S\mathbf{c}. \tag{A.11}$$

A.2.2. Spatial discretization with standard Galerkin and Dirichlet boundary conditions

Now we will also incorporate Dirichlet boundary conditions at the inlet boundaries

$$\begin{cases} \dfrac{\partial\phi}{\partial t} + \mathbf{u}(\mathbf{x})\cdot\nabla\phi = 0 & \mathbf{x}\in\Omega,\\ \phi = g(\mathbf{x}) & \mathbf{x}\in\partial\Omega_{\text{inlet}}. \end{cases} \tag{A.12}$$

With Galerkin's method we use the following approximation

$$\phi(\mathbf{x},t) \approx \phi^n(\mathbf{x},t) = \sum_{j\in\text{Ind}} c_j(t)\,v_j(\mathbf{x}) + \sum_{j\in\text{Dir}} g(\mathbf{x}_j)\,v_j(\mathbf{x}), \tag{A.13}$$

where $v_j(\mathbf{x})$ for $j = 1,\ldots,n$ are piecewise polynomials with property $v_j(\mathbf{x}_i) = \delta_{ij}$. The independent nodes are denoted by Ind and the Dirichlet nodes by Dir.

Substituting this into Eq. (3.2) yields

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}\sum_{j\in\text{Ind}} c_j(t)\int_\Omega v_j v_i\,\mathrm{d}\Omega = &\sum_{j\in\text{Ind}} -c_j(t)\int_\Omega\Big(\mathbf{u}\cdot\nabla v_j\Big)v_i\,\mathrm{d}\Omega\\ &-\sum_{j\in\text{Dir}} g(\mathbf{x}_j)\left(\int_\Omega v_j v_i\,\mathrm{d}\Omega + \int_\Omega\Big(\mathbf{u}\cdot\nabla v_j\Big)v_i\,\mathrm{d}\Omega\right), \quad \forall i\in\text{Ind}. \end{aligned} \tag{A.14}$$

This can be written as

$$\frac{\mathrm{d}}{\mathrm{d}t}\sum_{j\in\text{Ind}} M_{ij}c_j(t) = \sum_{j\in\text{Ind}} S_{ij}c_j(t) + \sum_{j\in\text{Dir}}(S_{ij}-M_{ij})g(\mathbf{x}_j), \quad \forall i\in\text{Ind}. \tag{A.15}$$

With the same derivation as in Section A.2.1 we obtain the semi-discrete system for Dirichlet boundary conditions at the inlet boundary

$$M\frac{\mathrm{d}\mathbf{c}}{\mathrm{d}t} = S\mathbf{c} + \mathbf{f}. \tag{A.16}$$

A.2.3. Spatial discretization with SUPG

The discretization is similar to the discretization from Section A.2.2, but instead the test function from Eq. (3.7) is used. This gives

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\sum_{j\in\mathrm{Ind}} c_j(t)\int_\Omega v_j(v_i+\delta_K\mathbf{u}\cdot\nabla v_i)\,\mathrm{d}\Omega = \\ \sum_{j\in\mathrm{Ind}} -c_j(t)\int_\Omega \Big(\mathbf{u}\cdot\nabla v_j\Big)(v_i+\delta_K\mathbf{u}\cdot\nabla v_i)\,\mathrm{d}\Omega \\ -\sum_{j\in\mathrm{Dir}} g(\mathbf{x}_j)\int_\Omega v_j(v_i+\delta_K\mathbf{u}\cdot\nabla v_i)\,\mathrm{d}\Omega \\ -\sum_{j\in\mathrm{Dir}} g(\mathbf{x}_j)\int_\Omega (\mathbf{u}\cdot\nabla v_j)(v_i+\delta_K\mathbf{u}\cdot\nabla v_i)\,\mathrm{d}\Omega,\quad \forall i\in\mathrm{Ind}.\end{aligned} \tag{A.17}$$

This can be written as

$$\frac{\mathrm{d}}{\mathrm{d}t}\sum_{j\in\mathrm{Ind}}\tilde{M}_{ij}c_j(t) = \sum_{j\in\mathrm{Ind}}\tilde{S}_{ij}c_j(t) + \sum_{j\in\mathrm{Dir}}(\tilde{S}_{ij}-\tilde{M}_{ij})g(\mathbf{x}_j),\quad \forall i\in\mathrm{Ind}. \tag{A.18}$$

with

$$\begin{aligned}\tilde{M}_{ij} &= \int_\Omega v_j(v_i+\delta_p\mathbf{u}\cdot\nabla v_i)\,\mathrm{d}\Omega \\ &= \frac{\Delta e_p}{24}\Big(1+\delta_{ij}+4\delta_p(\beta_i u_1(\mathbf{x}_j)+\gamma_i u_2(\mathbf{x}_j))\Big).\end{aligned} \tag{A.19}$$

and

$$\begin{aligned}\tilde{S}_{ij} &= -\int_\Omega \Big(\mathbf{u}\cdot\nabla v_j\Big)(v_i+\delta_K\mathbf{u}\cdot\nabla v_i)\,\mathrm{d}\Omega \\ &= -\frac{\Delta e_p}{6}\Big(\beta_j u_1(\mathbf{x}_i)+\gamma_j u_2(\mathbf{x}_i)+\delta_p((u_1^2)_p\beta_i\beta_j+(u_1u_2)_p\gamma_i\beta_j+(u_1u_2)_p\beta_i\gamma_j)+(u_2^2)_p\gamma_i\gamma_j\Big),\end{aligned} \tag{A.20}$$

with

$$u_p = \sum_{l\in\{p_1,p_2,p_3\}} u(\mathbf{x}_l).$$

The mass matrix element is computed as follows

$$\tilde{M}^{e_p} = \frac{|\Delta e_p|}{24}\begin{bmatrix} 2+4\delta_p[\beta_1\ \gamma_1]^\top\cdot\mathbf{u}(\mathbf{x}_1) & 1+4\delta_p[\beta_1\ \gamma_1]^\top\cdot\mathbf{u}(\mathbf{x}_2) & 1+4\delta_p[\beta_1\ \gamma_1]^\top\cdot\mathbf{u}(\mathbf{x}_3) \\ 1+4\delta_p[\beta_2\ \gamma_2]^\top\cdot\mathbf{u}(\mathbf{x}_1) & 2+4\delta_p[\beta_2\ \gamma_2]^\top\cdot\mathbf{u}(\mathbf{x}_2) & 1+4\delta_p[\beta_2\ \gamma_2]^\top\cdot\mathbf{u}(\mathbf{x}_3) \\ 1+4\delta_p[\beta_3\ \gamma_3]^\top\cdot\mathbf{u}(\mathbf{x}_1) & 1+4\delta_p[\beta_3\ \gamma_3]^\top\cdot\mathbf{u}(\mathbf{x}_2) & 2+4\delta_p[\beta_3\ \gamma_3]^\top\cdot\mathbf{u}(\mathbf{x}_3)\end{bmatrix}.$$

For the stiffness matrix we will only show how one entry is computed

$$\begin{aligned}\tilde{S}_{12} = -\frac{\Delta e_p}{6}\big(&\beta_2 u_1(\mathbf{x}_1)+\gamma_2 u_2(\mathbf{x}_1) \\ &+\delta_p((u_1^2)_p\beta_1\beta_2+(u_1u_2)_p\gamma_1\beta_2+(u_1u_2)_p\beta_1\gamma_2)+(u_2^2)_p\gamma_1\gamma_2\big).\end{aligned}$$

Then we obtain the semi-discrete SUPG system for Dirichlet boundary conditions at the inlet boundary

$$\tilde{M}\frac{\mathrm{d}\mathbf{c}}{\mathrm{d}t} = \tilde{S}\mathbf{c}+\tilde{\mathbf{f}}. \tag{A.21}$$

B

Discretization

B.1. Laplace equation

Consider the following problem

$$\begin{cases} \Delta f_s = 0, & \mathbf{x} \in \Omega, \\ \frac{\partial f_s}{\partial \mathbf{n}} = 0, & \mathbf{x} \in \partial\Omega, \\ f_s = \tilde{f}_s, & \mathbf{x} \in \Gamma. \end{cases} \tag{B.1}$$

Problem (B.1) is discretized with a standard Galerkin approach. The weak form is obtained by multiplying with a test function $v \in V_h^k$, which gives

$$\begin{aligned} \int_\Omega \nabla \cdot (\nabla f_s v) - \nabla f_s \cdot \nabla v \, \mathrm{d}\Omega = 0, \quad \forall v \in V_h^k \\ \int_{\partial\Omega} (\nabla f_s \cdot \mathbf{n}) v \, \mathrm{d}\Gamma - \int_\Omega \nabla f_s \cdot \nabla v \, \mathrm{d}\Omega = 0, \quad \forall v \in V_h^k \\ \int_\Omega \nabla f_s \cdot \nabla v \, \mathrm{d}\Omega = 0, \quad \forall v \in V_h^k \end{aligned} \tag{B.2}$$

where we used the divergence theorem and that $\frac{\partial f_s}{\partial \mathbf{n}} = 0$ at $\partial\Omega$.

With Galerkin's method we use the following approximation

$$f_s(\mathbf{x}) \approx f_s^n(\mathbf{x}) = \sum_{j \in \mathrm{Ind}} c_j v_j(\mathbf{x}) + \sum_{j \in \mathrm{Dir}} \tilde{f}_s(\mathbf{x}_j) v_j(\mathbf{x}). \tag{B.3}$$

Combining Eqs. (B.2) and (B.3) results in

$$\sum_{j \in \mathrm{Ind}} c_j \int_\Omega \nabla v_j \cdot \nabla v_i \, \mathrm{d}\Omega + \sum_{j \in \mathrm{Dir}} \tilde{f}_s(\mathbf{x}_j) \int_\Omega \nabla v_j \cdot \nabla v_i \, \mathrm{d}\Omega = 0, \quad \forall i \in \mathrm{Ind}. \tag{B.4}$$

By writing

$$M = \sum_{j \in \mathrm{Ind}} M_{ij} = \sum_{j \in \mathrm{Ind}} \int_\Omega \nabla v_j \cdot \nabla v_i \, \mathrm{d}\Omega,$$

and

$$\mathbf{b} = -\sum_{j \in \mathrm{Dir}} M_{ij} \tilde{f}_s(\mathbf{x}_j),$$

the weak form from Eq. (B.4) becomes

$$M\mathbf{c} = \mathbf{b}. \tag{B.5}$$

For the piecewise polynomial $v_j(\mathbf{x})$ we can write $\phi_j(\mathbf{x}) = \alpha_j + \beta_j x + \gamma_j y$. Hence

$$\begin{aligned} M_{ij} &= \int_\Omega \nabla v_j \cdot \nabla v_i \, \mathrm{d}\Omega \\ &= \int_\Omega [\beta_j \ \gamma_j]^\top \cdot [\beta_i \ \gamma_i]^\top \, \mathrm{d}\Omega \\ &= \frac{|\Delta_e|}{2}(\beta_j\beta_i + \gamma_i\gamma_j). \end{aligned}$$

B.2. Time-dependent Eikonal equation

Consider the time-dependent Eikonal equation, Eq. (2.5). The weak form is obtained by multiplying with a SUPG test function

$$z = v + \delta_p \mathbf{w} \cdot \nabla v.$$

This gives

$$\int_\Omega \frac{\partial \phi}{\partial \tau} v \, \mathrm{d}\Omega + \int_\Omega \left(\mathbf{w} \cdot \nabla \phi\right) v \, \mathrm{d}\Omega = \int_\Omega S(\phi_0) v \, \mathrm{d}\Omega, \quad \forall v \in V_h^{\mathrm{SUPG}}. \tag{B.6}$$

With Galerkin's method the following approximation is introduced

$$\phi(\mathbf{x}, \tau) \approx \phi^n(\mathbf{x}, \tau) = \sum_{j=1}^{n} c_j(\tau) v_j(\mathbf{x}), \tag{B.7}$$

where $v_j(\mathbf{x})$ for $j = 1, \ldots, n$ are piecewise polynomials with property $v_j(\mathbf{x}_i) = \delta_{ij}$ and $\sum_j v_j(\mathbf{x}) = 1$.

Substituting (B.7) into Eq. (B.6) yields

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t} \sum_{j \in \mathrm{Ind}} c_j(t) \int_\Omega v_j (v_i + \delta_p \mathbf{u} \cdot \nabla v_i) \, \mathrm{d}\Omega = \\ \sum_{j \in \mathrm{Ind}} -c_j(t) \int_\Omega \left(\mathbf{u} \cdot \nabla v_j\right) (v_i + \delta_p \mathbf{w} \cdot \nabla v_i) \, \mathrm{d}\Omega \\ + \int_\Omega S(\phi_0)(v_i + \delta_p \mathbf{w} \cdot \nabla v_i) \, \mathrm{d}\Omega, \quad \forall i \in \mathrm{Ind}, \end{aligned} \tag{B.8}$$

which can be written as

$$\frac{\mathrm{d}}{\mathrm{d}t} \sum_{j=1}^{n} \tilde{M}_{ij} c_j(\tau) = \sum_{j=1}^{n} \tilde{S}_{ij} c_j(\tau) + \tilde{b}_i, \quad i = 1, \quad \forall i \in \mathrm{Ind}. \tag{B.9}$$

Here, $\tilde{b}_i$ is given by

$$\begin{aligned} \tilde{b}_i &= \int_\Omega S(\phi_0)(v_i + \delta_p \mathbf{w} \cdot \nabla v_i) \, \mathrm{d}\Omega \\ &= \sum_{p=1}^{n_T} \int_{e_p} S(\phi_0)(v_i + \delta_p \mathbf{w} \cdot \nabla v_i) \, \mathrm{d}\Omega \\ &= \sum_{p=1}^{n_T} \tilde{b}_i^{e_p}, \quad \forall i \in \mathrm{Ind}, \end{aligned} \tag{B.10}$$

where e_p denotes a triangular element with vertices $\mathbf{x}_{p_1}$, $\mathbf{x}_{p_2}$ and $\mathbf{x}_{p_3}$, and n_T the number of elements.

The first part of the integral from Eq. (B.10) is evaluated with the Newton-Cotes quadrature rule. For the next part we write $S_{\phi_0} = S(\phi_0)$,

$$\begin{aligned} \int_{e_p} S_{\phi_0} v_i \, \mathrm{d}\Omega &\overset{\text{N-C}}{=} \frac{|\Delta e_p|}{6} \sum_{l \in \{p_1, p_2, p_3\}} S_{\phi_0}(\mathbf{x}_l) v_i(\mathbf{x}_l) \\ &= \frac{|\Delta e_p|}{6} \sum_{l \in \{p_1, p_2, p_3\}} S_{\phi_0}(\mathbf{x}_l) \delta_{il} \\ &= \frac{|\Delta e_p|}{6} S_{\phi_0}(\mathbf{x}_i), \quad i \in \{p_1, p_2, p_3\}. \end{aligned}$$

The second part is also evaluated with the Newton-Cotes quadrature rule, resulting in

$$b_i^{e_p} = \frac{|\Delta e_p|}{6}\left[S_{\phi_0}(\mathbf{x}_i) + \delta_p\left(\beta_i \sum_{l\in\{p_1,p_2,p_3\}}\left(S_{\phi_0} w_1\right)(\mathbf{x}_l) + \gamma_i \sum_{l\in\{p_1,p_2,p_3\}}\left(S_{\phi_0} w_2\right)(\mathbf{x}_l)\right)\right]. \tag{B.11}$$

Finally, the semi-discrete system is obtained

$$\tilde{M}\frac{\mathrm{d}\mathbf{c}}{\mathrm{d}\tau} = \tilde{S}\mathbf{c} + \tilde{\mathbf{b}}, \tag{B.12}$$

with $\tilde{M}$, $\tilde{S}$ and $\tilde{\mathbf{b}}$ defined by Eqs. (A.19), (A.20) and (B.11), respectively.