Development of an unstructured Finite Volume Level Set Method in OpenFOAM

Entwicklung einer unstrukturierten Finite Volumen Level Set Methode in OpenFOAM
Master thesis in the field of study “Computational Engineering” by Julian Reitzel
Date of submission: 31.05.2023

1. Review: Prof. Dr. Dieter Bothe
2. Review: Dr.-Ing. Tomislav Maric
Darmstadt

Development of an unstructured Finite Volume Level Set Method in OpenFOAM
Entwicklung einer unstrukturierten Finite Volumen Level Set Methode in OpenFOAM

Source code:

	Archive:	`https://doi.org/10.5281/zenodo.10721725`
	Repository:	`https://github.com/leia-openfoam/leia`
	Documentation:	`https://leia-openfoam.github.io/leia/`

Document:	Master thesis
Author:	Julian Reitzel
University:	Technical University of Darmstadt
Course of study:	M.Sc. Computational Engineering
Institute:	Mathematical Modeling and Analysis
Director of the institute:	Prof. Dr. Dieter Bothe
Supervisor:	Dr.-Ing. Tomislav Maric
Issued on:	1 December 2022
Submitted on:	31 May 2023

Copyright © 2023 Julian Reitzel

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License (CC BY-NC-SA 4.0, 🅭🅯🄏🄎).

Abstract

This master thesis presents the development and application of a Finite Volume Level Set Method for simulating two-phase flows on unstructured meshes within the OpenFOAM Computational Fluid Dynamics (CFD) framework. The proposed method focuses on incompressible, immiscible, non-reactive, isothermal, two-phase Newtonian fluid flows, considering surface tension forces and gravity. A main objective is to implement and evaluate the Signed Distance Preserving Level Set (SDPLS) method, as proposed by Fricke et al. [17]. Furthermore, the developed Level Set (LS) method is tested with various advection schemes on various mesh types, including hexahedral, perturbed hexahedral, and polyhedral meshes. Lastly, the LS method is coupled to the Navier-Stokes (NS) equations with the Continuum Surface Force (CSF) model proposed by Brackbill et al. [4] and the use of the geometrical phase indicator proposed by us [40], which does not rely on a signed distance property of the LS field. A segregated solver is implemented and validated through numerical studies of the 3D stationary droplet test case.

Zusammenfassung

Diese Master Thesis präsentiert die Entwicklung und Anwendung einer Finite Volumen Level Set Methode zur Simulation von Zweiphasenströmungen auf unstrukturierten Gittern innerhalb des OpenFOAM Computational Fluid Dynamics (CFD)-Frameworks. Die vorgeschlagene Methode konzentriert sich auf inkompressible, nicht mischbare, nicht reaktive, isotherme Zweiphasenströmungen mit Newtonscher Fluiden, unter Wirkung von Oberflächenspannungskräften und Schwerkraft. Ein Hauptziel besteht darin, die Signed Distance Preserving Level Set (SDPLS)-Methode von von Fricke et al. [17] zu implementieren und zu untersuchen. Darüber hinaus wird die entwickelte Level Set (LS) Methode mit verschiedenen Advektionsschemata auf verschiedene Gittertypen getestet, einschließlich hexahedrischer, gestörter hexahedrischer und polyedrischer Gitter. Schließlich wird die LS Methode mit den Navier-Stokes (NS) Gleichungen gekoppelt unter Verwendung des Continuum Surface Force (CSF) Modells von Brackbill et al. [4] und des von uns vorgeschlagenen geometrischen Phasenindikators [40], der nicht auf einer vorzeichenbehafteten Distanzeigenschaft des LS Feldes beruht. Ein segregierter Solver wird implementiert und durch numerische Studien des Testfalls stationärer 3D-Tropfen validiert.

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Dr.-Ing. Tomislav Maric, for his valuable guidance and support throughout my master's thesis. His expertise and encouragement have been instrumental in shaping my master thesis.

I am also thankful to Prof. Dr. Dieter Bothe for his valuable insights and contributions, which played a significant role in deepening my understanding of the subject.

I would like to extend my appreciation to the faculty members and staff of the Mathematical Modeling and Analysis institute for creating this welcoming and conducive academic environment.

Furthermore, I gratefully acknowledge the computing time provided to me on the high-performance computer Lichtenberg at the NHR Centers NHR4CES at TU Darmstadt. This is funded by the Federal Ministry of Education and Research, and the state governments participating on the basis of the resolutions of the GWK for national high performance computing at universities (www.nhr-verein.de/unsere-partner).

Lastly, I am grateful to my friends and family for their unwavering support and understanding during my Master's thesis.

Contents

List of Figures

Acronyms

ACLS	Accurate Conservative Level Set.
AMR	Adaptive Mesh Refinement.
BVP	Boundary Value Problem.
CAD	Computer Aided Design.
CDS	Central Differencing Scheme.
CFD	Computational Fluid Dynamics.
CLI	Command Line Interface.
CLS	Conservative Level Set.
CSF	Continuum Surface Force.
ENO	Essentially Non-Oscillatory.
FDM	Finite Difference Method.
FEM	Finite Element Method.
FMM	Fast Marching Method.
FSM	Fast Sweeping Method.
FT	Front Tracking.
FVM	Finite Volume Method.
HPC	High Performance Computing.

IBVP	Initial-Boundary Value Problem.
IVP	Initial Value Problem.
LS	Level Set.
NS	Navier-Stokes.
NVD	Normalised Variable Diagram.
ODE	Ordinary Differential Equation.
PDE	Partial Differential Equation.
RTS	Runtime Type Selection.
SDPLS	Signed Distance Preserving Level Set.
TVD	Total Variation Diminishing.
VoF	Volume of Fluid.
WENO	Weighted Essentially Non-Oscillatory.

1. Introduction

1.1. Motivation

Fluid flows occur in many scenarios in nature and in technical applications. In particular, understanding and predicting multiphase flows in engineering systems is of great interest because many engineering flows have more than one phase. The field of multiphase flows includes

- free surface flows, for example in the marine industry,
- spray and jet flows, for example in combustion engines
- porous media flows, for example in fracking applications in the oil industry,
- surface tension driven flows such as flows in capillary structures for example in lap-on-a-chip applications,
- wetting flows, for example droplets sliding on a wall,
- dispersion flows, for example in chemical reactors, and
- phase-change flows, for example in flash boilers in thermal process applications.

Numerical methods for the investigation and prediction of flow problems have been in development since the 1930s. While the simulation of single-phase flows has become state of the art, the simulation of multi- or two-phase flows is still very challenging. At present, there is no best approach and the choice depends on the problem, the relevant physics and the available computing resources. Typically, trade-offs are made between volume conservation, numerical stability, numerical consistency, order of convergence, handling of topological interface changes, computational efficiency and ability to handle complex geometries.

This work addresses the LS method in the context of the unstructured Finite Volume Method (FVM). Two-phase flows in this work are restricted to be immiscible, incompressible, isothermal, laminar, non-reactive, without phase change, with surface tension forces, under gravity influence and consisting of isotropic Newtonian fluids. Although there are other methods capable of simulating such a two-phase flow problem. The LS has the advantages of being computationally very efficient, naturally handling topological changes such as merging or splitting, and easily calculating interface properties such as interface normal and curvature. The LS method is categorised in comparison to other methods in section 1.2.

Typically, the FVM is the numerical discretization method of choice in CFD because of its high volume conservation. Here the unstructured FVM is chosen because it uses unstructured meshes for domain discretization, which are crucial for engineering applications where complex geometries are often encountered. The discretization of complex domains is often not possible or too difficult for structured meshes. In this case mesh generation algorithms are used, which derive meshes with arbitrarily shaped polyhedral cells from Computer Aided Design (CAD) models. OpenFOAM is the chosen CFD software framework for unstructured FVM and is introduced in section 1.3. However, the unstructured Finite Volume Level Set Method is not without its problems.

- During the advection of the LS field, the LS profile deteriorates. Current techniques to reinitialize or preserve the LS profile harm the order of convergence, move the interface, are not mass conservative or are computational inefficient.
- Oscillations arising during the advection of the LS field are typically circumvented by the use of higher order Weighted Essentially Non-Oscillatory (WENO) schemes. For unstructured meshes and the use of Adaptive Mesh Refinement (AMR) in outlook, WENO schemes are computationally inefficient.
- On signed distance relying or inaccurate phase indication leads to parasitic currents in the flow field near the interface.

Most implementations of the LS method are developed for Cartesian meshes, as the typically used WENO schemes are complicated for unstructured meshes [44]. This work provides an implementation of the LS method with the unstructured FVM in OpenFOAM and tests various advection schemes on different mesh types. Additionally, as Trujillo et al. [76] states, the effectiveness of the reinitialization depends on the resolution. If the ratio between the local curvature and the grid spacing is too high, reinitializing the LS profile is not only compromised, but reinitializing even harm the accuracy of the method [76]. With the problems listed above, this motivates methods which preserve the LS profile and do not need reinitialization. One technique for preserving the LS profile is to use expansion velocities. Unfortunately, computing these at each time step is computationally expensive. Another preservation method, the SDPLS method developed by Fricke et al. [17], shows promising results and is implemented and tested in this work. Furthermore, the LS method is coupled to the NS equations and a segregated solver for simulating two-phase flows is implemented.

In chapter 2 the foundations for the mathematical model of two-phase flows and the unstructured FVM are given. In chapter 3 the LS method is presented and a literature review is given. Chapter 4 describes how two-phase flows are modeled with the LS method and derives the discrete equations. In chapter 5, the most important developments and implementations are described. Chapter 6 presents the results for interface advection with the developed LS method, the implemented SDPLS method and results for the 3D stationary droplet two-phase flow test case with the developed solver. Chapter 7 summarizes and discusses the main results of this thesis.

1.2. Two-phase flow numerical methods

In the field of continuum physics, the modelling of two-phase flow systems involves the description of the different phases and their positions in space. There are a variety of approaches to consider, each with their own advantages and disadvantages depending on the specific physical phenomena, time and domain scales, and available computational resources. A comprehensive review of different numerical two-phase flow methods is given in [57, 79, 21]. Here, the categories of numerical two-phase flow methods and how the LS method relates to them are described.

Numerical two-phase flow methods can be classified in many ways. These include the governing equations (Eulerian, Lagrangian and Lagrangian-Eulerian methods); the discretization of the governing equations (Finite Difference Method (FDM), FVM, Finite Element Method (FEM) and spectral methods); and the nature of the interface (sharp interface and diffuse interface methods). The LS method belongs to the Eulerian methods, which implicitly describe the interface as a field defined over the entire solution domain. In this work, the LS method is combined with the unstructured FVM, as the FVM is commonly used in CFD due to its high volume conservation. However, the LS method can also be combined with the FEM, as was originally done by Dervieux and Thomasset [12].

The LS method is a sharp interface method, which means that the interface between two phases is infinitesimally thin and carries no mass. This is particularly true for immiscible phases and is a valid assumption in many cases. However, for small scale events such as small droplet collisions, the boundary layer between the phases (approximately $100\,\mathrm{nm}$ thick) may be taken into account. This is done by diffuse interface methods, such as the phase field method [21]. There are other sharp interface methods, such as Front Tracking (FT) methods or Volume of Fluid (VoF) methods. FT methods explicitly represent the interface by line segments in two dimensions and triangular surface segments in three dimensions, and track them in a Lagrangian manner. Despite the high accuracy of these techniques, difficulties arise with topological changes and re-meshing of interface regions where segments become too dense or sparse. VoF methods, on the other hand, implicitly represent the interface through a volume fraction field, have volume conservation to machine tolerance, and can naturally handle topological changes. However, the computation of interface properties such as curvature or surface normal is difficult. LS methods represent the interface implicitly, through a LS of a higher dimensional function, and have low computational cost. They can also easily compute the curvature and interface normals, and handle topological changes naturally. The main disadvantage of the LS method is its lack of volume conservation and accuracy. See section 3 for a detailed description of the LS method.

1.3. OpenFOAM

OpenFOAM is an open source software for CFD, developed by OpenCFD Ltd and distributed under the GPL licence. It is an object-oriented and highly efficient C++ framework which is suitable for the development and testing of numerical methods without the need to implement the unstructured FVM from scratch. OpenFOAM provides an extensive library of solvers and models for various physical phenomena such as multiphase flows and turbulence. It also includes advanced CFD features such as AMR, support for unstructured meshes and MPI parallelism for running on High Performance Computing (HPC) clusters. In addition, its text file-based simulation cases allow scripting and automation, such as parameter study generation with `pyFoam`. In this work, OpenFOAM-v2212 is chosen as the implementation framework due to these several advantageous features. [46, 74]

2. Foundations

2.1. Mathematical model

This section provides a description of the investigated problem and a brief overview of the mathematical model that is used. For a detailed derivation of the equations, the interested reader is referred to [20, 21, 78, 79].

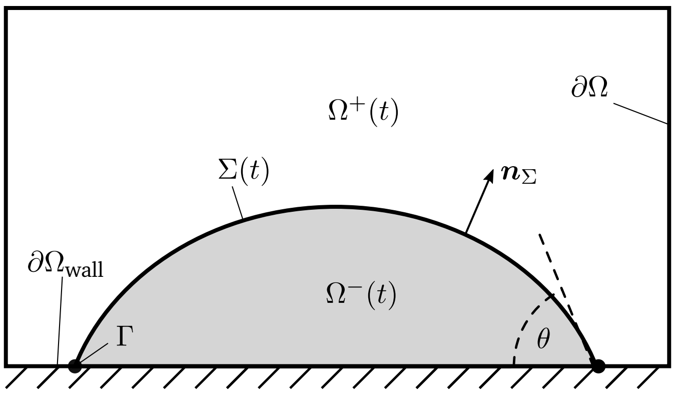


Figure 2.1.: Mathematical model [37]

A two or three dimensional spatial domain $\Omega \subset \mathbb{R}^2$ or $\mathbb{R}^3$ is given. Within the domain, a laminar two-phase flow system is considered where the two phases are immiscible, incompressible, viscous, isotropic Newtonian with constant material properties. Both phases move in time and are subject to surface tension and gravity forces. The system is isothermal and there are no phase transitions or reactions between the phases. The two phases in the domain occupy the subdomains $\Omega^+(t)$ and $\Omega^-(t)$ which are functions of time t such that $\Omega = \Omega^+(t) \cup \Omega^-(t)$ and $\Omega^+(t) \cap \Omega^-(t) = \emptyset$. The domain boundaries are denoted as $\partial\Omega$. The phases are separated by a sharp interface $\Sigma(t)$, which is infinitesimally thin and contains no mass. The interface normal vector $\boldsymbol{n}_\Sigma$ points from $\Omega^-(t)$ to $\Omega^+(t)$. See the sketch in figure 2.1. Within the domain the velocity field $\boldsymbol{v}(\mathbf{x}, t)$ and the pressure field $p(\mathbf{x}, t)$ are defined as functions of position $\mathbf{x} \in \Omega$ and time $t \in [t^0, t^{end}]$. The density and viscosity of the phases are denoted as $\rho^\pm$, $\mu^\pm$.

2.1.1. Standard model for two-phase flows

The conservation of mass and momentum holds within each subdomain, $\Omega^-(t)$ and $\Omega^+(t)$, and is governed by the NS equations (2.2) with the gravity vector $\boldsymbol{g}$ and the stress tensor

$$\mathbf{S}^i = -p\mathbf{I} + \mu^i(\nabla \boldsymbol{v} + \nabla(\boldsymbol{v})^T). \tag{2.1}$$

$$\left.\begin{aligned} \nabla \cdot \boldsymbol{v} &= 0 \\ \partial_t(\rho^i \boldsymbol{v}) + \nabla \cdot (\rho^i \boldsymbol{v}\boldsymbol{v}) &= \nabla \cdot \mathbf{S}^i + \rho^i \boldsymbol{g} \end{aligned}\right\} \text{ in } \Omega^i,\ i = +, - \tag{2.2}$$

$$\left.\begin{aligned} [\![\boldsymbol{v}]\!] &= 0 \\ [\![\mathbf{S}]\!] \cdot \boldsymbol{n}_\Sigma &= \sigma \kappa_\Sigma \boldsymbol{n}_\Sigma \end{aligned}\right\} \text{ on } \Sigma \tag{2.3}$$

Boundary conditions, equation (2.3), with $[\![\phi]\!] = \phi^+ - \phi^-$ model the interface $\Sigma(t)$. For the velocity no-slip is considered, hence the velocity is continuous. Furthermore, the surface tension force at an interface balances the discontinuity of the normal stress along the fluid interface [5], with the constant surface tension coefficient σ, the interface mean curvature κ_Σ as defined in equation (2.4) and the surface normal unit vector $\boldsymbol{n}_\Sigma$ pointing into subdomain $\Omega^+(t)$. For a detailed derivation see [21].

$$\kappa_\Sigma = -\nabla_\Sigma \cdot \boldsymbol{n}_\Sigma \tag{2.4}$$

Equations (2.2) and (2.3) are known as the standard model for two-phase flows [21]. However, since the interface moves with time, the subdomain boundaries are not stationary, which causes difficulties. Therefore, the next formulation is often used with the LS method.

2.1.2. Single-field Navies-Stokes formulation

Instead of having two sets of NS equations, one for each subdomain $\Omega^\pm(t)$, and interface conditions, an equivalent Partial Differential Equation (PDE) system for the whole domain was derived by Brackbill et al. [4]. The phase indicator function χ, also known as characteristic or marker function [12, 78, 4, 64], is introduced as

$$\chi(\mathbf{x}, t) = \begin{cases} 1, & \mathbf{x} \in \Omega^- \\ 0, & \mathbf{x} \in \Omega^+ \end{cases}, \tag{2.5}$$

and is used to ensure that discontinuities across the interface are integrated into the material properties. With χ the global density ρ and viscosity μ are expressed as

$$\rho(\mathbf{x}, t) = \chi(\mathbf{x}, t)\rho^- + (1 - \chi(\mathbf{x}, t))\rho^+, \tag{2.6}$$

$$\mu(\mathbf{x}, t) = \chi(\mathbf{x}, t)\mu^- + (1 - \chi(\mathbf{x}, t))\mu^+. \tag{2.7}$$

With these global representations of the material properties the set of equations is rewritten as

$$\left.\begin{aligned} \nabla \cdot \boldsymbol{v} &= 0 \\ \partial_t(\rho \boldsymbol{v}) + \nabla \cdot (\rho \boldsymbol{v}\boldsymbol{v}) &= \nabla \cdot \mathbf{S} + \rho \boldsymbol{g} + \boldsymbol{f}_\Sigma \end{aligned}\right\} \text{ in } \Omega, \tag{2.8}$$

to the so-called single-field or one-fluid NS formulation [5, 78], which is valid in the whole domain. It utilises the surface tension force not as a boundary condition anymore but rather as a volumetric force $\boldsymbol{f}_\Sigma$. Brackbill et al. [4] introduced the CSF model to calculate $\boldsymbol{f}_\Sigma$ as

$$\boldsymbol{f}_\Sigma = \sigma \kappa_\Sigma \delta_\Sigma \boldsymbol{n}_\Sigma, \tag{2.9}$$

where $\delta_\Sigma = \delta(\mathbf{x} - \mathbf{x}_\Sigma)$ is a Dirac δ-distribution that results in $\boldsymbol{f}_\Sigma$ acting as a singular volumetric force, see [20, 21] for a derivation. Rewriting the momentum equation and inserting the dynamic pressure p' defined as

$$\begin{aligned} p &= p' + \rho \boldsymbol{g} \cdot \mathbf{x} \\ -\nabla p &= -\nabla p' - (\boldsymbol{g} \cdot \mathbf{x})\nabla \rho - \rho \boldsymbol{g} \end{aligned} \tag{2.10}$$

yields the set of single-field NS equations that will be used throughout this work

$$\left.\begin{aligned} \nabla \cdot \boldsymbol{v} &= 0 \\ \partial_t(\rho \boldsymbol{v}) + \nabla \cdot (\rho \boldsymbol{v}\boldsymbol{v}) - \nabla \cdot (\mu \nabla \boldsymbol{v}) &= -\nabla p' - (\boldsymbol{g} \cdot \mathbf{x})\nabla \rho + \boldsymbol{f}_\Sigma + \nabla \cdot (\mu (\nabla \boldsymbol{v})^T) \end{aligned}\right\} \text{ in } \Omega. \tag{2.11}$$

2.2. Unstructured Finite Volume Method

This section provides a description of the unstructured Finite Volume Method (FVM). For a comprehensive introduction it is referred to [39, 80, 42].

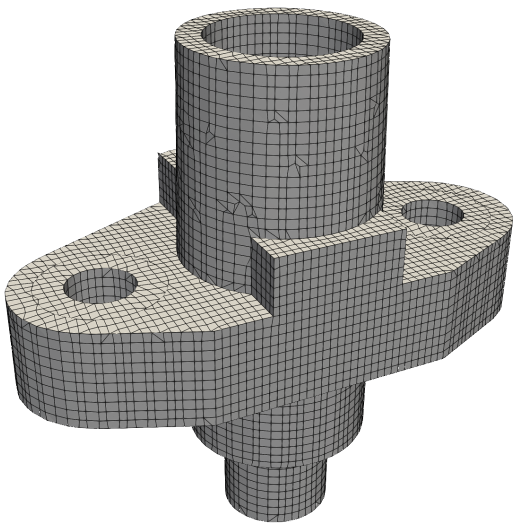


Figure 2.2.: Unstructured mesh for a flange from the cfMesh tutorials

In fluid dynamics and other disciplines, the physical system is often modelled as a system of Ordinary Differential Equations (ODEs) or PDEs with initial and/or boundary conditions. The evaluation of an analytical solution is only possible in rare cases. Therefore numerical approximation methods have been developed, the unstructured FVM is one of them. Usually numerical methods discretize the PDEs into an algebraic linear system, which then is solved, at least approximately. With discretization, the continuous solution space is transformed into a discrete space. Additionally, all derivatives are discretized, by replacing them with representations connected to the discrete solution like finite differences. The idea of the FVM is to divide the domain into a finite number of small volumes, also called cells, see figure 2.2 for an example of a discretized domain. The key step of the FVM is to integrate the governing equations over each cell yielding a system of coupled equations. Discretizing these equation with so-called schemes and solving the resulting algebraic linear system yields an approximated discrete solution of the problem. Overall the FVM can be structured in the three steps of

1. domain discretization,
2. PDE discretization,
3. solving the algebraic linear system.

For demonstration purpose an Initial-Boundary Value Problem (IBVP) with a scalar transport PDE is considered. The three dimensional spatial domain is denoted by $\Omega \subset \mathbb{R}^3$. The domain can has any shape, for example see figure 2.2. The independent variable $\mathbf{x} \in \Omega$ denotes position in the domain and the independent variable $t \in [t_0, t_{end}]$ time between t^0 the initial time and t^{end} the end time. $\phi(\mathbf{x}, t)$ is the solution variable which depends on $\mathbf{x}$ and t. Given is a velocity field $\boldsymbol{v}(\mathbf{x}, t)$, D the diffusion coefficient, the possible non-linear autonomous source term $\hat{S}(\phi(\mathbf{x}, t))$, the function $g(x)$ describing fixed Diriclet-values at the Direclet domain boundary $\partial\Omega_D$, a homogeneous Neumann boundary condition at $\partial\Omega_N$ and a initial condition $\phi_0(\mathbf{x})$ defined at t^0. Then, the IBVP is formulated as

$$\begin{aligned} \frac{\partial\phi(\mathbf{x},t)}{\partial t} + \nabla\cdot(\boldsymbol{v}\phi(\mathbf{x},t)) - \nabla\cdot(D\nabla\phi(\mathbf{x},t)) &= \hat{S}(\phi(\mathbf{x},t)), && \mathbf{x} \in \Omega,\ t \in [t^0, t^{end}], \\ \phi(\mathbf{x}) &= g(\mathbf{x}), && \mathbf{x} \in \partial\Omega_D, \\ \nabla\phi(\mathbf{x}) \cdot \mathbf{n}_{\partial\Omega} &= 0, && \mathbf{x} \in \partial\Omega_N, \\ \phi(\mathbf{x}, t_0) &= \phi_0(\mathbf{x}), && \mathbf{x} \in \Omega. \end{aligned} \tag{2.12}$$

2.2.1. Domain discretization

Spatial discretization

The first step is to approximate the solution domain Ω by a discretized solution domain Ω_h. With that the domain is divided into a set C of disjoint finite volumes or cells which form the solution domain as an union

$$\Omega_h = \bigcup_{c \in C} \Omega_c. \tag{2.13}$$

This set of cells is called mesh or grid. Unlike the structured FVM, cells in the unstructured FVM can have any shape and are constructed from at least four possible non-planar faces. Typically, hexahedral cells are desired, but any polyhedral cell shape is possible. See figure 2.2 for an unstructured mesh example. In contrast to the structured FVM, this does not allow for directional connectivity in the mesh. Each cell in the mesh is only connected to its immediate neighbours. Essential for the FVM is the centroid $\mathbf{x}_c$ of a cell Ω_c. Let

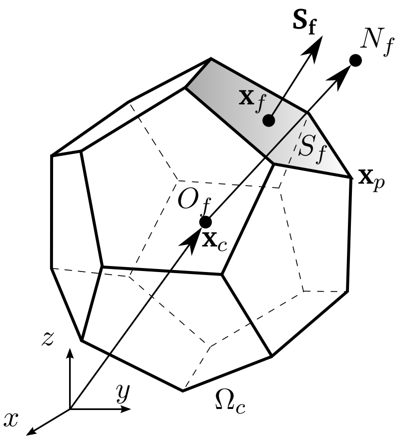


Figure 2.3.: Polyhedral cell [38, figure 3]

$P = \{\mathbf{x}_c : c \in C\}$ be the set of centroids of all cells C. Figure 2.3 shows a polyhedral cell Ω_c with its cell centroid $\mathbf{x}_c$, a shaded face f, its face centroid $\mathbf{x}_f$ and its surface normal vector $\mathbf{S}_f$ with length $|S_f|$. From the view of the face f, it is shared between two cells O_f and N_f, where the latter is not shown. For several reasons, ownership is declared for each face. The face is owned by cell O_f and cell N_f is its neighbour. The surface normal vector $\mathbf{S}_f$ always points from owner to neighbour.

Temporal discretization

The temporal domain $T = [t^0, t^{end}]$ is discretized into T_h, a finite set of points in time written as

$$T_h = \left\{t^n : t^0 < t^1 < \ldots < t^n < \ldots < t^{end}\right\} \tag{2.14}$$

Solution discretization

Goal of the FVM is to find a approximate solution to the IBVP, equation (2.12). With the spatial and temporal domain being discretized as Ω_h and T_h we can define the approximate solution $\phi_h \approx \phi(\mathbf{x}, t)$ as

$$\phi_h = \{\phi_h(\mathbf{x}_c, t^n) : \mathbf{x}_c \in P,\ t^n \in T_h\} \tag{2.15}$$

The approximate solution ϕ_h is the set of solution values $\phi_h(\mathbf{x}_c, t^n)$ defined at the cell centroids $\mathbf{x}_c$ at time t^n, representing mean values for each cell. For the rest of the work the abbreviation $\phi_h(\mathbf{x}_c, t^n) = \phi_c^n$ is used.

2.2.2. Equation discretization

As the equation depends on spatial dimensions $\mathbf{x}$ and the temporal dimension t, the discretization is done separately.

Spatial discretization

Having the spatial domain discretized, the scalar transport PDE, equation (2.12), is not considered globally anymore, instead the equation is considered in every cell. Integrating the equation over every cell, with index $c \in C$ denoting any cell in the set C, yields

$$\int_{\Omega_c} \frac{\partial \phi(\mathbf{x}, t)}{\partial t} \, dV + \int_{\Omega_c} \nabla \cdot (\boldsymbol{v} \phi(\mathbf{x}, t)) \, dV - \int_{\Omega_c} \nabla \cdot (D \nabla \phi(\mathbf{x}, t)) \, dV = \int_{\Omega_c} \hat{S}(\phi(\mathbf{x}, t)) \, dV. \tag{2.16}$$

With the use of the divergence theorem, the second and third volume integral of the left hand site of the equation are rewritten as surface integrals, which yields

$$\int_{\Omega_c} \frac{\partial \phi(\mathbf{x}, t)}{\partial t} \, dV + \int_{\partial \Omega_c} (\boldsymbol{v} \phi(\mathbf{x}, t)) \cdot \mathbf{n} \, dS - \int_{\partial \Omega_c} (D \nabla \phi(\mathbf{x}, t)) \cdot \mathbf{n} \, dS = \int_{\Omega_c} \hat{S}(\phi(\mathbf{x}, t)) \, dV. \tag{2.17}$$

Then, the surface integrals are replaced by the sum of surface integrals of each face of the cell, yielding

$$\int_{\Omega_c} \frac{\partial \phi(\mathbf{x}, t)}{\partial t} \, dV + \sum_{f \in F_c} \int_{S_f} \phi(\mathbf{x}, t) \boldsymbol{v} \cdot \mathbf{n} \, dS - \sum_{f \in F_c} \int_{S_f} (D \nabla \phi(\mathbf{x}, t)) \cdot \mathbf{n} \, dS = \int_{\Omega_c} \hat{S}(\phi(\mathbf{x}, t)) \, dV. \tag{2.18}$$

Until this point, the equation is still exact for cell Ω_c.

Discretizing spatial derivatives

Now, first approximations and the discretized solution variable ϕ_h are introduced with the discretization of derivatives. Starting with the discretization of spatial derivatives, the temporal derivatives are ignored for the moment. In the spatial dimension, the problem is a Boundary Value Problem (BVP), which is discretized with the unstructured FVM leading to a semi-discretized PDE, semi-discretized because the temporal derivatives will still be there. First, in equation (2.18) the volume integrals are replaced by the centroid value ϕ_c times the cell volume $|\Omega_c|$ and the surface integrals are replaced by the surface centroid value ϕ_f times the surface area $|S_f|$, see equations (2.19) and (2.20). This is second order accurate and is further explained in [39].

$$\int_{\Omega_c} \phi(\mathbf{x})\, dV \approx \phi_c |\Omega_c| \tag{2.19}$$

$$\int_{S_f} \phi(\mathbf{x})\, dS \approx \phi_f |S_f| \tag{2.20}$$

For the temporal term, the Leibniz integral rule is applied to switch between volume integral and partial temporal derivative. Then the volume integral is replaced by the centroid value times the cell volume, see equation (2.21).

$$\int_{\Omega_c} \frac{\partial \phi(\mathbf{x},t)}{\partial t}\, dV = \frac{\partial}{\partial t} \int_{\Omega_c} \phi(\mathbf{x},t)\, dV \approx \frac{\partial \phi_c(t)}{\partial t} |\Omega_c| \tag{2.21}$$

The source term can generally be any non-linear function, that the final linear solver can not handle. Linearization with $\hat{S}_c$ being a constant and $\hat{S}_p$ the proportionality constant to the linear part yields

$$\hat{S}(\phi(\mathbf{x},t)) \approx \hat{S}_c + \hat{S}_p \phi(\mathbf{x},t). \tag{2.22}$$

Inserting equations (2.19)-(2.22) into the PDE, equation (2.18), yields

$$\frac{\partial \phi_c}{\partial t} |\Omega_c| + \sum_{f \in F_c} \phi_f \boldsymbol{v}_f \cdot \mathbf{S}_f - \sum_{f \in F_c} D_f (\nabla \phi)_f \cdot \mathbf{S}_f = S_c |\Omega_c| + S_p \phi_c |\Omega_c|, \tag{2.23}$$

where $\mathbf{S}_f = \mathbf{n}_f |S_f|$ and F_c being the set of all faces belonging to cell c. This introduces face values ϕ_f in the advection term and surface normal gradients $(\nabla\phi)_f \cdot \mathbf{S}_f$ in the Laplace term. Since the FVM only solves for the centroid values ϕ_c, these quantities are approximated by so-called schemes, similar to finite differences. In the following, the standard schemes are presented and numerical errors typical for unstructured meshes are mentioned. Details of these errors are given in the section 2.2.4.

For the advection term, the *Gauß linear* scheme is typically chosen because of its second order accuracy. For Cartesian grids it is known as CDS. The general second order convergence is compromised if the mesh is skewed. It is also not bounded, so oscillations in the

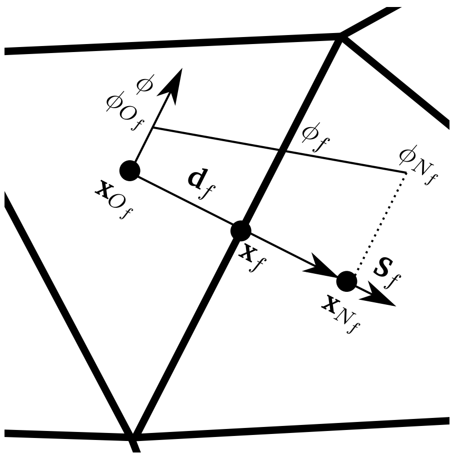


Figure 2.4.: CDS applied on an orthogonal mesh [39, Figure 1.8]

solution can occur. [42]

$$\phi_f \approx w_f \phi_{N_f} + (1 - w_f)\phi_{O_f} \quad \text{with} \quad w_f = \frac{\|\mathbf{x}_f - \mathbf{x}_{O_f}\|_2}{\|\mathbf{d}_f\|_2}, \quad \mathbf{d}_f = \mathbf{x}_{N_f} - \mathbf{x}_{O_f} \tag{2.24}$$

For the Laplace term, the *Gauß linear orthogonal* scheme is typically chosen. The Laplace term is smooth and bounded. However, the second order convergence of the scheme is harmed by a non-equidistant and skewed mesh. In addition, a non-orthogonal mesh harms the convergence order, which can be compensated by explicit cross-diffusion correction. [80, 42, 27]

$$(\nabla\phi)_f \cdot \mathbf{S}_f = (\nabla\phi)_f \cdot \mathbf{n}_f |S_f| \approx \frac{\phi_{N_f} - \phi_{O_f}}{\|\mathbf{d}_f\|_2} |S_f| \tag{2.25}$$

Inserting the interpolation schemes, equation (2.24) and (2.25), into equation (2.23) yields equation (2.26), the semi-discretized scalar transport PDE, where N_c is the set of all neighbouring cells to cell c and a_c, a_N are coefficients summed up from all known quantities.

$$\frac{\partial \phi_c}{\partial t} = \hat{a}_c \phi_c + \sum_{n \in N_c} \hat{a}_N \phi_N + \hat{S}_c \tag{2.26}$$

Temporal discretization

Now the semi-discretized PDE, equation (2.26), of each cell forms a coupled system of scalar Initial Value Problems (IVPs), because only the temporal derivatives and the initial conditions are left. There are several numerical methods for solving IVPs, including one-step and multi-step methods. The focus here is on one-step methods. Given a general scalar IVP

$$\begin{aligned} \frac{\partial \phi(t)}{\partial t} &= f(t, \phi(t)), \qquad t \in [t^0, t^{end}], \\ \phi(t_0) &= \phi_0. \end{aligned} \tag{2.27}$$

Temporally integrating the problem from t^n to t^{n+1}, where $\delta t = t^{n+1} - t^n$, and using the right quadrature rule to approximate the integral yields the implicit Euler method

$$\frac{\phi^{n+1} - \phi^n}{\delta t} = f(t^{n+1}, \phi^{n+1}), \tag{2.28}$$

which is the default in OpenFOAM and is first order accurate. Using the trapezoidal quadrature rule yields the second order accurate Crank-Nicolson method

$$\frac{\phi^{n+1} - \phi^n}{\delta t} = \frac{1}{2} f(t^n, \phi^n) + \frac{1}{2} f(t^{n+1}, \phi^{n+1}). \tag{2.29}$$

By applying the chosen method to the IVP given in equation (2.26), collecting all the coefficients and inserting the solution values from the previous time step ϕ^n into the source term, yields the final discretized equation

$$a_c \phi_c^{n+1} + \sum_{n \in N_c} a_N \phi_N^{n+1} = S_c. \tag{2.30}$$

This equation couples the solution ϕ_c^{n+1} of cell c to its immediate neighbours and can not be solved without them. Assembling the equations of all cells yields a large algebraic linear system of the form

$$\mathbf{A} \phi^{n+1} = \mathbf{b}, \tag{2.31}$$

where **A** is the system matrix and **b** the force vector. For the incorporation of the boundary condition it is referred to [39, 42, 80].

2.2.3. Solving the algebraic linear system

This linear algebraic system must be constructed and solved at each time step. It is a large system of size $N \times N$, where N is the number of cells. As only the neighbouring cells affect the solution for a given cell, this system has a band structure and is sparse. With these characteristics, iterative linear solver are appropriate for solving, like Gauß-Seidel or Conjugate-Gradient. For further details on solving algebraic linear systems, the reader is referred to [63].

2.2.4. Discretization errors with unstructured meshes

The section lists errors related to unstructured meshes. Unstructured meshes can be characterised as non-equidistant, skewed and non-orthogonal, all of these characteristics are suboptimal and lead to further discretization errors that can negatively affect the convergence rate.

Non-equidistant error

With a non-equidistant mesh, the distance between the face centroid and the owner cell centroid is generally not equal to the distance between the face centroid and the neighbour cell centroid $\|\mathbf{x}_f - \mathbf{x}_{O_f}\|_2 \neq \|\mathbf{x}_f - \mathbf{x}_{N_f}\|_2$.. As a result, a face centroid and the centroid between $\mathbf{x}_{O_f}$ and $\mathbf{x}_{N_f}$ do not coincide. This error affects Laplacian schemes, since the gradient approximation is only second order accurate at the centroid of the vector $\mathbf{x}_{N_f} - \mathbf{x}_{O_f}$.

Skewness error

Mesh skewness occurs when the vectors $\mathbf{x}_f - \mathbf{x}_{O_f}$ and $\mathbf{x}_f - \mathbf{x}_{N_f}$ are generally not collinear. Again, the result is that a surface centroid and the centroid between $\mathbf{x}_{O_f}$ and $\mathbf{x}_{N_f}$ do not coincide. Hence Laplacian schemes are affected. But furthermore, advection schemes are affected, as the face interpolation do not cross the face center.

Non-orthogonality error

The mesh is not orthogonal if the vectors $\mathbf{x}_{N_f} - \mathbf{x}_{O_f}$ and $\mathbf{S}_f$ are not parallel. This error affects Laplacian schemes as the gradient in the face normal direction $\mathbf{S}_f$ can not be expressed with ϕ_{N_f} and ϕ_{O_f}.

3. Level Set Method

This chapter provides an overview of the Level Set (LS) method. A literature survey is given with emphasis on developments regarding LS reinitialization and preserving. The plain LS method is a pure interface advection method with the advecting velocity field being considered given at any time. This will be the case for this chapter. The full method for simulating two-phase flow fields with the LS method coupled to the single-field NS equations (2.11) is explained in chapter 4.

3.1. Method overview

In 1979 the LS method was developed by Dervieux and Thomasset [12] and was first used in the CFD context for the advection of two-phase flow. But beyond that, the LS method is a general concept for describing and advecting surfaces. With several contributions by Osher and Sethian, the LS method has been popularised and applied in the fields of image processing, computer vision, computer graphics, computational geometry and optimization [50, 68, 48, 47, 49, 81].

With LS methods, an interface is implicitly represented as a LS of a higher dimensional function defined in the domain Ω. Let $\psi(\mathbf{x}, t)$ define a scalar function, here called LS function, as a function of position $\mathbf{x} \in \Omega$ and time $t \in [t^0, t^{end}]$. Then an interface $\Sigma(t)$ can be defined as the λ-LS, see equation (3.1). Typically, the zero LS denotes the interface, but other LS are possible, as in [45, 3].

$$\Sigma(t) = \{\mathbf{x} : \psi(\mathbf{x}, t) = \lambda\}, \quad \lambda \in \mathbb{R} \tag{3.1}$$

Figure 3.1 sketches a interface $\Sigma(t)$ and its enclosed phase (grey) for a two dimensional case. The interface $\Sigma(t)$ can be pictured as the intersection of the LS function $\psi : [\mathbb{R}^2, \mathbb{R}] \rightarrow \mathbb{R}$ (red) with the plane (blue) representing the λ-level. Hence, the interface $\Sigma(t)$ consists of the set of points where ψ has the value λ, the so-called λ-LS.

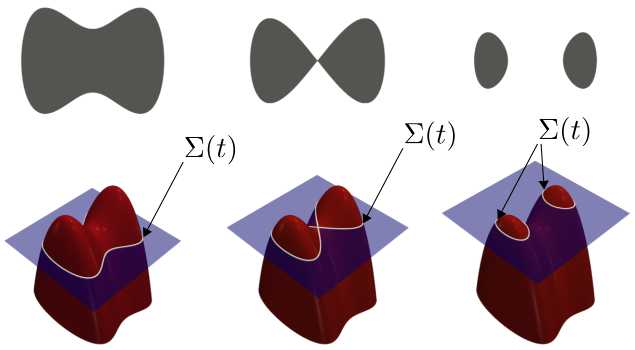


Figure 3.1.: Adapted from Nicoguaro (2018). Level-set method. Wikimedia. Licensed under CC-BY-4.0.

Advected is the interface implicitly by advecting the LS field with the so-called LS advection equation (3.5), which is derived from the definition that a fluid particle of the interface $\mathbf{X}(t) \in \Sigma(t)$ keeps its λ-LS

$$\frac{d\psi(\mathbf{X}(t), t)}{dt} = 0, \tag{3.2}$$

evaluating the total derivative

$$\frac{\partial\psi(\mathbf{x}, t)}{\partial t} + \frac{d\mathbf{X}(t)}{dt} \nabla \cdot \psi(\mathbf{x}, t) = 0, \tag{3.3}$$

inserting the pathline ODE

$$\frac{d\mathbf{X}(t)}{dt} = \boldsymbol{v}(\mathbf{x}, t) \tag{3.4}$$

and the incompressibility condition, equation (2.11).

$$\frac{\partial\psi(\mathbf{x}, t)}{\partial t} + \nabla \cdot (\boldsymbol{v}(\mathbf{x}, t)\psi(\mathbf{x}, t)) = 0 \tag{3.5}$$

The level-set function should be smooth in order to compute derivatives and should fulfill

$$\begin{aligned} \psi(\mathbf{x}, t) > \lambda, \quad & \mathbf{x} \in \Omega^{+}(t), \\ \psi(\mathbf{x}, t) < \lambda, \quad & \mathbf{x} \in \Omega^{-}(t). \end{aligned} \tag{3.6}$$

Having a LS function, the interface normal vector $\boldsymbol{n}_\Sigma$, the interface mean curvature κ_Σ and the phase indicator χ can be computed as

$$\mathbf{n}_\Sigma(\mathbf{x}, t) = \frac{\nabla\psi(\mathbf{x}, t)}{\|\nabla\psi(\mathbf{x}, t)\|_2}, \qquad \mathbf{x} \in \Sigma(t), \tag{3.7}$$

$$\kappa_\Sigma(\mathbf{x}, t) = -\nabla \cdot \boldsymbol{n}_\Sigma, \qquad \mathbf{x} \in \Sigma(t), \tag{3.8}$$

$$\chi(\mathbf{x}, t) = \begin{cases} 1, & \psi(\mathbf{x}, t) \leq \lambda, \\ 0, & \psi(\mathbf{x}, t) > \lambda, \end{cases} \qquad \mathbf{x} \in \Omega. \tag{3.9}$$

Typically, the LS function is initialised to be the signed distance function (equation (3.10)), where the interface is at the zero LS and has the signed distance property (equation (3.11)). This function has many advantages, such as avoiding steep and flat gradients, being monotone across the interface, being very smooth, and avoiding rapidly changing features [47]. These characteristics make it a good choice for the LS function.

$$\psi(\mathbf{x}) = \begin{cases} +\min_{\mathbf{x}_\Sigma \in \Sigma} \|\mathbf{x} - \mathbf{x}_\Sigma\|_2, & \mathbf{x} \in \Omega^+, \\ -\min_{\mathbf{x}_\Sigma \in \Sigma} \|\mathbf{x} - \mathbf{x}_\Sigma\|_2, & \mathbf{x} \in \Omega^-, \\ 0, & \mathbf{x} \in \Sigma \end{cases} \tag{3.10}$$

$$\|\nabla\psi(\mathbf{x})\|_2 = 1, \quad \mathbf{x} \in \Omega \tag{3.11}$$

The LS method has a number of advantages, such as the ability to easily extend from two to three dimensions, the natural handling of topological changes such as coalescence or breakups as sketched in figure 3.1, the natural handling of singularities and self-intersections in the evolving interface, the easy computation of curvature and the interface normal which are required to evaluate the surface tension force, and the computational efficiency due to the lack of explicit computation of the interface. However, the level set method is not without its drawbacks. Discrete solutions of the LS advection equation (3.5) are prone to numerical errors that lead to mass loss or gain [5]. In particular, with grid-scale interface structures, LS methods lose (or gain) mass [61, 60]. In addition, as the interface is advected, the signed distance property can be lost and gradients can diminish or become very steep when advected over a long time [8, 72]. This leads to errors in the numerical evaluation of geometric properties of the interface such as interface normal and curvature. Also, the desired second order convergent CDS scheme for the advection term produces oscillations around the interface, as can be seen in [45].

The interface properties, mean curvature and surface normal, are needed for the evaluation of the surface tension force in the single field NS equations, when simulating a two-phase flow field. Since these quantities are derived from the LS field, it is crucial to have an accurate advected LS field. Any noise in the LS field will lead to even more noise in the normal field and mean curvature, since they are computed with derivatives of the LS field [10]. Furthermore, wrong interface properties lead to high artificial velocities near the interface, so-called parasitic currents [12, 72, 47]. As mentioned, the full two-phase flow simulation with the LS method is described in chapter 4. In this chapter, for the pure interface advection, the interface properties are not needed, but their accurate computation should be mentioned as an important goal at this point.

For the advection type LS equation (3.5) different advection schemes have been used and developed for accurate advection and numerical oscillation prevention. An overview is given in the section 3.2. Several developments have been proposed to avoid the loss of the LS profile. Reinitialization attempts to restore the signed distance LS of a distorted signed distance field. This is reviewed in section 3.3. A similar procedure is attempted by Conservative Level Set (CLS) methods. After some advection steps, a supporting PDE is solved in pseudo time to restore the chosen LS profile. This is discussed in the section 3.5. With extension velocities, the LS field is tried to be preserved by using an alternative velocity field. This approach is reviewed in section 3.4. Similarly, the novel SDPLS method, which attempts to preserve the signed distance as the LS field is advected, is presented in section 3.6.

3.2. Advection schemes

This section discussed the important role of the advection scheme for discretization of the LS advection equation (3.5) and briefly presents schemes that has been used in the literature.

As noted in chapter 2.2, the standard advection scheme is *Gauß linear*, also called CDS, which is second order accurate but not bounded/ monotone. However, despite the fact that the LS field is uniformly smooth across the interface, a number of studies have shown that the CDS can generate numerical oscillations close to the interface [45, 3, 83]. These numerical oscillations occur in regions of small gradients due to partial floating-point cancellation errors [51], in regions of steep gradients close to discontinuities [3, 56] or in regions of thin films, break-ups or merging of interfaces where discontinuous derivatives occur [54, 58]. Unfortunately, the LS advection equation (3.5) lacks a diffusion term, which would naturally compensate for numerical oscillations. Therefore, a more bounded/ monotone scheme for the advection of the LS field is needed. The *upwind* scheme is the simplest choice for a bounded scheme, but is only first order accurate. Since the FVM is generally second order accurate and a high accuracy of the LS field is desired for the purposes of mass conservation and accurate interface quantities, other high accuracy schemes are required. But according to the Godunov theorem, higher order linear schemes for discretizing PDEs can not be monotone. This led to the use of higher order monotone non-linear Total Variation Diminishing (TVD), Normalised Variable Diagram (NVD), Essentially Non-Oscillatory (ENO), WENO and Semi-Lagrangian schemes for LS methods.

The ENO and WENO schemes are commonly used for LS methods due to their high order of accuracy. They involve a non-oscillatory piecewise polynomial reconstruction of the solution from its cell averages and use a larger adaptive stencil of grid points, see [23, 35]. Despite their high accuracy for unstructured meshes, these schemes are difficult to implement due to their large stencils. Existing implementations for unstructured meshes make reuse of assembled stencils to be computationally efficient, but for AMR in outlook the computational cost of assembling and communicating across multiple MPI boundaries makes them inefficient for the goal of this work. There is no ENO or WENO scheme available in OpenFOAM-v2212.

Similar to blended schemes, TVD and NVD schemes blend upwind biased and CDS schemes. However, their blending factor, called flux limiter, is evaluated on a local basis. TVD and NVD schemes are also popular in the LS context because they are computationally efficient and do not require large stencils, making them suitable for unstructured meshes. The Total Variation TV, for one dimension defined in equation (3.12) where $f \in F$ and F is the set of all internal faces, with the boundedness criterion, equation (3.13), have been used to derive a TVD condition for adaptive bounded schemes that blend between a higher order scheme and a lower order upwind scheme [22, 73]. OpenFOAM provides several TVD schemes such as *van Leer* and *SuperBee*.

$$TV(\psi^n) = \sum_f |\psi_N^n - \psi_O^n| \tag{3.12}$$

$$TV(\psi^{n+1}) \leq TV(\psi^n) \tag{3.13}$$

With the concept of NVD Leonard [31] invented a first NVD scheme and a less restrictive and more general approach to boundedness [26]. In the following, the explanation is given for Cartesian meshes, while the adaptation to unstructured meshes can be found in [26]. To avoid oscillations, it is required that each cell value ψ_c and the values of its upwind and downwind neighbours ψ_U, ψ_D satisfy

$$\psi_U \leq \psi_c \leq \psi_D \quad \text{or} \quad \psi_U \geq \psi_c \geq \psi_D. \tag{3.14}$$

Let ψ_f be the face value between the cell Ω_c and its downwind neighbour Ω_D. For Cartesian meshes the normalized face value $\tilde{\psi}_f$ is evaluated as a non-linear function of the normalized cell value $\tilde{\psi}_c$ as

$$\tilde{\psi}_f = f(\tilde{\psi}_c) \quad \text{with} \quad \tilde{\psi}_c = \frac{\psi_c - \psi_U}{\psi_D - \psi_U} \quad \text{and} \quad \tilde{\psi}_f = \frac{\psi_f - \psi_U}{\psi_D - \psi_U} \tag{3.15}$$

In order for the scheme to ensure ψ_f stays bounded between ψ_c and ψ_D, following must be fulfilled

$$\begin{aligned} \tilde{\psi}_c \leq \tilde{\psi}_f \leq 1, \qquad & \text{if} \quad 0 \leq \tilde{\psi}_c \leq 1, \\ \tilde{\psi}_f = \tilde{\psi}_c, \qquad & \text{else}. \end{aligned} \tag{3.16}$$

The second case states that whenever an oscillation appears in the scale of the grid size, a switch to *upwind* is made. Consequently, any flux limiter satisfying the equation (3.16) yields a monotone scheme. OpenFOAM provides NVD schemes such as *MUSCL*, *UMIST*.

Note that all high accuracy monotone schemes reviewed so far are non-linear, meaning they have weights or flux limiters that depend on the solution itself and are explicitly evaluated, introducing a CFL-like condition [73]. Furthermore, these schemes are derived considering one dimension without general statements to more dimensions. This makes convergence proofs only available in rare cases. [26]

To come to another class of schemes, the semi-Lagrangian schemes. So far, all the advection schemes reviewed have been of Eulerian type. This means that they discretize the Eulerian type LS advection equation (3.5). The Semi-Lagrangian schemes or Semi-Lagrangian methods also use an Eulerian grid, but discretize the Lagrangian advection equation (3.2), where $\mathbf{X}(t)$ is a fluid particle in the domain Ω. For example, using the implicit Euler method to discretize the ODE yields

$$\psi(\mathbf{X}(t^{n+1}), t^{n+1}) = \psi(\mathbf{X}(t^n), t^n). \tag{3.17}$$

In other words, a fluid particle keeps its LS. The idea of Semi-Lagrangian schemes is to calculate the point of origin of a fluid particle. If the position of the particle $\mathbf{X}(t^{n+1})$ at time t^{n+1} is known, the LS value can be interpolated with the LS field of the previous time step. The position of the origin is calculated with the pathline ODE, equation (3.4), using the same ODE approximation method as before. Semi-Lagrangian schemes update all centroids of an Eulerian mesh with this approach, are unconditionally stable in time and bounded if a bounded interpolation is used, such as linear interpolation for structured meshes or inverse distance weighted interpolation for unstructured meshes [70]. In general, Lagrangian schemes are more accurate in advecting small phase filaments than their Eulerian counterparts [61, 60]. However, Semi-Lagrangian schemes are not strictly mass conservative in the LS context, since the rate of change does not correspond to the flux over faces as in the standard FVM. [18]. Nevertheless, Semi-Lagrangian schemes show good mass conservation as they preserve material characteristics [14]. OpenFOAM-v2212 does not provide Semi-Lagrangian schemes.

3.3. Reinitialization

One of the earliest developments regarding the LS method is the reinitialization or also called redistancing. This technique is specific to the signed distance LS function. When the signed distance LS field is advected, it loses its signed distance property. Thus, reinitialization techniques must be applied periodically to restore the signed distance. Given a distorted signed distance field $\tilde{\psi}(\mathbf{x})$, the goal of reinitialization is to restore a signed distance field $\psi(\mathbf{x})$ without moving the initial zero LS. The reinitialization problem can be formulated as

$$\begin{aligned} \|\nabla\psi(\mathbf{x})\|_2 &= 1, \quad \mathbf{x} \in \Omega, \\ \psi(\mathbf{x}) &= 0, \quad \mathbf{x} \in \{\mathbf{x} : \tilde{\psi}(\mathbf{x}) = 0\}. \end{aligned} \tag{3.18}$$

There are different approaches, which can be divided into PDE based reinitialization an direct reinitialization.

3.3.1. Partial Differential Equation based reinitialization

The initial idea of a reinitialization PDE came from Rouy and Tourin [62] and was further developed by Sussman et al. [72]. Instead of solving the problem equation directly (3.18), the reinitialization PDE (3.19) is suggested.

A distorted signed distance field $\tilde{\psi}(\mathbf{x}, t)$ is given at time t. For the corrected signed distance field $\psi(\mathbf{x}, \tau)$ the reinitializing PDE is solved in the pseudo time τ. As equation (3.19) restores the signed distance from the interface to the boundaries, there is even no need to solve for the steady state.

$$\begin{aligned} \frac{\partial\psi(\mathbf{x},\tau)}{\partial\tau} &= \text{sgn}(\tilde{\psi}(\mathbf{x},t))(1 - \|\nabla\psi(\mathbf{x},\tau)\|_2), \\ \psi(\mathbf{x},0) &= \tilde{\psi}(\mathbf{x},t) \end{aligned} \tag{3.19}$$

with

$$\operatorname{sgn}(\psi) = \begin{cases} -1, & \psi < 0, \\ 0, & \psi = 0, \\ +1, & \psi > 0 \end{cases} \tag{3.20}$$

or to avoid discontinuity the sgn function is often smoothed by using

$$\operatorname{sgn}_{\text{smoothed}}(\psi) = \frac{\psi}{\sqrt{\psi^2 + \varepsilon^2}} \tag{3.21}$$

where ε is proportional to the mesh spacing [47]. This method is effective if the level set function is initially close to a signed distance function, but if the level set function is flat near the interface, it may take a long time to complete the process. Furthermore, analytically the reinitialization equation (3.19) does not move the zero LS, but as reported by several authors the discretized equation does, and can even move the interface across multiple cells due to numerical errors. [9, 47, 58, 54]

Sussman and Fatemi [71] proposed a reinitialization method with improved mass conservation. They extended the equation (3.19) with a so-called constraint and used higher order ENO schemes to improve the accuracy. But according to Saye and Sethian [64], the proposed corrections are not satisfactory and direct reinitialization is the more modern and preferable approach as they do not move the interface across cells.

3.3.2. Direct reinitialization

The direct reinitialization method, first introduced by Chopp [8], is the most intuitive approach to reinitialization. This technique explicitly constructs the interface and computes the signed distances to it. While some effective algorithms for direct redistancing on Cartesian meshes have become available, methods for unstructured meshes are more rare. Methods developed for unstructured meshes were mainly for the FEM, but the idea is transferable to the FVM. They all locate the position of the interface by linear interpolation between two mesh nodes with a change of sign. The linear interpolations are connected to form the approximated interface as a union of linear segments. The main differences between the methods developed by [15, 6, 44] lie in the way they compute the distances from far field nodes to the approximated interface.

The main advantage of these methods is that they do not move the zero LS across cells [44]. However, these methods are computationally expensive and slow, especially if reinitialization has to be done at each time step. They also have some unpleasant side effects, such as oscillations in curvature [54]. All reported methods for unstructured meshes are only first order accurate.

Conservative direct reinitialization

Mut et al. [43] proposed a mass conservative direct reinitialization technique for unstructured meshes also in FEM context. After reinitialization the nodes of cells containing the interface, a local mass conservation correction step is proposed. There the LS of the nodes is corrected to have the same amount of characteristic phase as before. Let $H(\psi)$ be the Heaviside step function, then for all cells containing the interface $\Omega_\Sigma \in \{\Omega_c : \Omega_c \cap \Sigma \neq \emptyset\} \subseteq \Omega_h$, nodal update LS values are calculated iteratively to fulfil the following conditions

$$\int_{\Omega_\Sigma} H(\psi(\mathbf{x}))d\mathbf{x} = \int_{\Omega_\Sigma} H(\tilde{\psi}(\mathbf{x}))d\mathbf{x}, \quad \Omega_\Sigma \in \{\Omega_c : \Omega_c \cap \Sigma \neq \emptyset\} \tag{3.22}$$

Since a node can be shared by several cells containing the interface in FEM, the update values of the nodes are averaged and scaled to fulfil a global mass constraint similar to equation (3.22). Nodes in the far field are updated using an algorithm similar to the Fast Marching Method (FMM), see section 3.3.3. Further improvements have been suggested [2, 41] but without significant improvements in the order of convergence.

3.3.3. Fast Marching Method

Fast Marching Method (FMM), a development by Sethian [67, 65, 68], is a Dijkstra-like algorithm for approximating the solution to non-linear eikonal equations of the form

$$\begin{aligned} \|\nabla\psi(\mathbf{x})\|_2 &= F(\mathbf{x}), \quad F(\mathbf{x}) > 0, \quad && \mathbf{x} \in \Omega, \\ \psi(\mathbf{x}) &= g(\mathbf{x}), && \mathbf{x} \in \Sigma. \end{aligned} \tag{3.23}$$

The FMM can be used to directly solve the reinitialization problem (3.18) efficiently with $O(N\log(N))$ complexity, where N is the number of cells. For this the FMM method is divided into two steps: the initialization and the actual iteration of the method. Goal of the initialization is to have a narrow band of cells around the interface with values that approximately fulfill equation (3.23) which can then be used to start the iteration cycle. This narrow band initialization can be done by direct reinitialization techniques, see section 3.3.2. Suppose a initialised narrow band is present, the actual iteration starts by finding the cell with the smallest value which is not fixed, followed by updating and fixing its value using a discretized version of equation (3.23). This process is then repeated for all cells starting from the initialised cells at the interface to the boundaries.

In its first form [67], it was designed for Cartesian meshes and was only first order accurate as it used the *upwind* scheme for gradient computations. Kimmel and Sethian [29] proposed a variation of this first order FMM for two dimensional unstructured meshes, in particular triangulated meshes. Later Sethain proposed a second order accurate FMM for Cartesian meshes [65]. Sethian and Vladimirsky [66] extended the second order FMM technique to general unstructured meshes. For Cartesian meshes, Chopp [9] proposed a new interpolation method for initializing high order FMMs using higher order polynomials and root finding with Newton's method. But the resulting set of equations is difficult to solve and the iterative Newtonian root finding has a high computational cost and may not converge [47]. According to Saye and Sethian [64], using the FMM to periodically reinitialise the signed distance field is computationally expensive in higher dimensions as it rely on the geometrically complicated reconstruction of the interface with direct reinitialization techniques [64]. The similar Fast Sweeping Method (FSM) by Zhao [82] has the same initialization problem .

3.4. Extension velocities

The extension velocities approach, inspired by other areas where LS methods are used, also addresses the challenge of preserving the signed distance property. In applications such as dendritic solidification, the velocity is defined only at the interface, so it is necessary to extend the interface velocity into the entire domain to solve equation (3.5). It is preferable to extend the interface velocity normal to the interface into the domain. This means that for a point $\mathbf{x}$ not at the interface, its extension velocity $\boldsymbol{v}_{ext}(\mathbf{x})$ is the velocity of the nearest point at the interface. Mathematically, this is expressed by the equation (3.24).

$$\begin{aligned} \nabla \boldsymbol{v}_{ext}(\mathbf{x}) \cdot \nabla \psi(\mathbf{x}) &= 0, \qquad && \mathbf{x} \in \Omega, \\ \boldsymbol{v}_{ext}(\mathbf{x}) &= \boldsymbol{v}(\mathbf{x}), && \mathbf{x} \in \Sigma \end{aligned} \tag{3.24}$$

Using extension velocities for level-set advection, equation (3.5), preserves the signed distance property [1, 69, 7].

Malladi et al. [36] computed extension velocities in the domain by extracting the velocity at the closest point to the front for all grid points. This approach assumes that the position of the interface is known, requiring an interface approximation. The calculation of extension velocities with the FMM has been proposed by Adalsteinsson and Sethian [1]. However, the problem of how to initialise the narrow band around the interface with high accuracy for unstructured meshes arises as well.

$$\frac{\partial \boldsymbol{v}_{ext}(\mathbf{x},\tau)}{\partial \tau} + \text{sgn}(\psi(\mathbf{x},t))\frac{\nabla\psi(\mathbf{x},t)}{\|\nabla\psi(\mathbf{x},t)\|_2} \cdot \nabla \boldsymbol{v}_{ext}(\mathbf{x},\tau) = 0 \tag{3.25}$$

Zhao et al. [83] proposed to achieve a solution to equation (3.24) by solving the support PDE, equation (3.25), in pseudo time τ, where no boundary conditions are needed as the characteristics flow out of the interface. Hence there is no need to approximate the interface.

3.5. Conservative Level Set Method

Olsson and Kreiss [45] developed the so-called Conservative Level Set (CLS) method with the goals of being conservative, not introducing spurious oscillations, and keeping the profile of the LS function constant. They suggested using a smeared Heaviside LS function

$$\tilde{\psi}(\mathbf{x},t) = \begin{cases} 0, & \psi < \varepsilon, \\ \frac{1}{2} + \frac{\psi}{2\varepsilon} + \frac{1}{2\pi}\sin\left(\frac{\pi\psi}{\varepsilon}\right), & -\varepsilon \leq \psi \leq \varepsilon, \\ 1, & \psi > \varepsilon, \end{cases} \tag{3.26}$$

where ψ is the signed distance (equation (3.10)). The sharp interface is defined at the LS of $\tilde{\psi}(\mathbf{x},t) = 0.5$ and at the same time a diffusive interface representation is defined for $0 < \tilde{\psi}(\mathbf{x},t) < 1$. However, this LS function is subject to the loss of the initial profile during advection as the standard signed-distance profile is. Similarly, Olsson and Kreiss introduced a reinitialization PDE which they solved in pseudo time τ after each time step.

$$\frac{\partial\tilde{\psi}}{\partial\tau} + \nabla\cdot\tilde{\psi}(1-\tilde{\psi})\boldsymbol{n}_\Sigma = \varepsilon\Delta\tilde{\psi} \tag{3.27}$$

Here the convection term acts as a compression flux to the diffuse interface $0 < \tilde{\psi}(\mathbf{x},t) < 1$. To avoid discontinuity, the compression flux is balanced by a diffusion term which depends on the diffusive interface thickness ε. To initialize the LS field at time $t = 0$, equation (3.27) is solved to steady state.

Using the FDM with uniform grid and CDS or TVD schemes for the convection term, the two-dimensional rotating circle test case was investigated. Without reinitialization, the

TVD scheme *SuperBee* performed best and CDS performed worst. On the other hand, with four reinitialization steps the cases have no visual difference and the *SuperBee* scheme shows ~ 1.3 order of convergence for the geometrical error where the CDS shows ~ 1.9 order of convergence.

Balcázar et al. [3] investigated a two-dimensional shear test case using the CLS method coupled with the unstructured FVM on a triangular mesh. They developed a TVD scheme which is a mixture of CDS, upwind and *SuperBee*. Their tests showing good accuracy without providing further error properties.

Desjardins et al. [13] have further improved the solving of two-phase flows with the CLS method. Their Accurate Conservative Level Set (ACLS) method uses the hyperbolic tangent LS function and the redistancing PDE from the CLS method, which shows good conservation of the interface during redistancing. However, according to them, the computed normal field still has spurious oscillations which distort the curvature and thus the surface tension calculation. They proposed to include a FMM to compute a temporary signed distance field, which is used to compute the normal field. Implementations were made using the FDM and higher order TVD schemes for the convection term. Their method indeed shows better performance than the CLS method, but at the cost of evaluating the FMM at each time step. [13] Still, the CLS approach introduces interface distortions and is under research [52].

3.6. Signed Distance Preserving Level Set Method

This section introduces the Signed Distance Preserving Level Set (SDPLS) method developed by Fricke et al. [17]. The idea is similar to the CLS method. Instead of solving a redistancing PDE after each time step, the original LS advection equation (3.5) is extended to preserve the signed distance during advection. In particular, Fricke et al. propose a non-linear source term, see the r.h.s. of the extended LS advection equation (3.28). Following their derivation, the source term compensates for local interface stretching or compression at the zero level set. It is referred to the original paper for details. [17]

$$\frac{\partial \psi}{\partial t} + \nabla \cdot (\boldsymbol{v}\psi) = \psi \left\langle (\nabla \boldsymbol{v}) \frac{\nabla \psi}{\|\nabla \psi\|_2}, \frac{\nabla \psi}{\|\nabla \psi\|_2} \right\rangle \tag{3.28}$$

Fricke et al. tested advection with the FVM on Cartesian meshes with a time explicit discretization, upwind discretization for the convection term and a second order CDS for the gradients in the source term. Among others, a two-dimensional test case with a droplet on a wall exposed to a time- periodic velocity field was investigated. Compared to a setup with the conventional LS advection equation (3.5), the SDPLS equation (3.28) preserves $\|\nabla\psi(\mathbf{x},t)\|_2 = 1$ and shows first order convergence for the signed distance, curvature and contact angle.

4. Two-phase flows

This chapter describes the coupling of the LS equation with the NS equations for the simulation of two-phase flows with the unstructured Finite Volume Level Set method. Starting with an overview of the governing equations for the description of two-phase flows with the LS method, the discretization of these equations with the unstructured FVM is derived. The difficulties encountered are discussed and the final discrete equations for the iterative segregated solution algorithm are derived. In the last section the importance of the surface tension force is presented.

4.1. Governing equations

To describe two-phase flows with the LS method, the single-field NS equations (2.11) as derived in the chapter 2.1 are required. They introduce velocity $\boldsymbol{v}$ and pressure $\mathbf{p}$ as solution variables and the surface tension force $\boldsymbol{f}_\Sigma$ on the r.h.s.. The interface motion between the two phases is prescribed by the LS equation (3.5), where the LS field ψ is introduced as a solution variable. The NS equations are coupled to the LS equation by providing the velocity that moves the LS field. The LS equation is coupled to the NS equation by the material properties ρ and μ, which depend on the phase indicator χ, and the surface tension force $\boldsymbol{f}_\Sigma(\kappa_\Sigma, \boldsymbol{n}_\Sigma)$, which includes interface properties. These PDEs together with the material properties, equations (2.6) and (2.7), and the phase indicator, equation (3.9), form the governing equations and are summarised in the following box, where the dynamic pressure is renamed as $p \leftarrow p'$.

Single-field NS equations

$$\nabla \cdot \boldsymbol{v} = 0 \tag{4.1}$$

$$\partial_t(\rho \boldsymbol{v}) + \nabla \cdot (\rho \boldsymbol{v}\boldsymbol{v}) - \nabla \cdot (\mu \nabla \boldsymbol{v}) = -\nabla p - (\boldsymbol{g} \cdot \mathbf{x})\nabla \rho + \boldsymbol{f}_\Sigma + \nabla \cdot (\mu (\nabla \boldsymbol{v})^T) \tag{4.2}$$

LS equation

$$\partial_t \psi + \nabla \cdot (\boldsymbol{v}\psi) = 0 \tag{4.3}$$

Material properties

$$\rho(\mathbf{x}, t) = \chi(\mathbf{x}, t)\rho^- + (1 - \chi(\mathbf{x}, t))\rho^+ \tag{4.4}$$

$$\mu(\mathbf{x}, t) = \chi(\mathbf{x}, t)\mu^- + (1 - \chi(\mathbf{x}, t))\mu^+ \tag{4.5}$$

Phase indicator

$$\chi(\mathbf{x}, t) = \begin{cases} 1, & \psi(\mathbf{x}, t) \le \lambda, \\ 0, & \psi(\mathbf{x}, t) > \lambda \end{cases} \tag{4.6}$$

4.2. Discretization

In this section, the discretization of the governing equations (4.1)-(4.5) for the iterative segregated solution algorithm is derived.

4.2.1. Discretizing the Partial Differential Equations

The three PDEs, the continuity equation (4.1), the momentum equation (4.2) and the LS equation (4.3), are discretized with the unstructured FVM and the implicit Euler method, as explained in chapter 2.2.

The continuity equation (4.1) discretizes as

$$\sum_f \boldsymbol{v}_f^{n+1} \cdot \mathbf{S}_f = 0. \tag{4.7}$$

The momentum conservation equation (4.2) discretizes as

$$\frac{\rho_c^{n+1}\boldsymbol{v}_c^{n+1}-\rho_c^n\boldsymbol{v}_c^n}{\delta t}|\Omega_c| + \sum_f \rho_f^{n+1}F_f^{n+1}\boldsymbol{v}_f^{n+1} - \sum_f \mu_f^{n+1}(\nabla\boldsymbol{v})_f^{n+1}\cdot\mathbf{S}_f =$$
$$-(\nabla p)_c^{n+1}|\Omega_c| - (\boldsymbol{g}\cdot\mathbf{x})_c(\nabla\rho)_c^{n+1}|\Omega_c| + (\boldsymbol{f}_\Sigma)_c^{n+1}|\Omega_c| + \sum_f \mu_f^{n+1}((\nabla\boldsymbol{v})^T)_f^{n+1}\cdot\mathbf{S}_f. \quad (4.8)$$

And the LS advection PDE, equation (4.3), discretizes as

$$\frac{\psi_c^{n+1}-\psi_c^n}{\delta t}|\Omega_c| + \sum_f \boldsymbol{v}_f^{n+1}\psi_f^{n+1}\cdot\mathbf{S}_f = 0. \quad (4.9)$$

4.2.2. Discrete phase indicator

This section introduces the discrete phase indicator α_h, also called the volume averaged phase indicator. The discrete phase indicator is needed because the discretization with the FVM introduces cell averaged material properties ρ_c and μ_c into the discretized momentum equation (4.8). The FVM typical cell-wise integration of the continuous material property equations (4.4) and (4.5) leads to the definition of the discrete phase indicator as

$$\alpha_c(t) = \frac{1}{|\Omega_c|}\int_{\Omega_c}\chi(\mathbf{x},t)\,dV. \quad (4.10)$$

Since the continuous phase indicator $\chi(\mathbf{x},t)$ is unknown, this definition can't be used for the computation. Instead, the computation is performed with a selected phase indicator model A as

$$\alpha_h = A[\psi_h]. \quad (4.11)$$

With the calculated discrete phase indicator α_c inside a cell Ω_c, the material properties for this cell are calculated as

$$\rho_c = \alpha_c\rho^- + (1-\alpha_c)\rho^+ \quad (4.12)$$
$$\mu_c = \alpha_c\mu^- + (1-\alpha_c)\mu^+. \quad (4.13)$$

Sharp phase indicator model

The first publications of the LS method used the most intuitive way to calculate the phase indicator α_c. According to the definition of the phase indicator χ, equation (4.6), the use of the sharp Heaviside function yields the sharp phase indicator model

$$\alpha_{\text{sharp},\,c} = \begin{cases} 1, & \psi_c \leq \lambda, \\ 0, & \psi_c > \lambda. \end{cases} \tag{4.14}$$

As accurate as this phase indication approach may be, it introduces discontinuities. This makes the numerical method prone to discretization errors. Therefore, the standard LS method often uses the following smoothed version.

Smoothed Heaviside phase indicator model

The smoothed Heaviside phase indicator model introduces a slightly diffuse interface representation over multiple cells with a phase indicator value between $(0, 1)$. It is defined as

$$\alpha_{\text{smooth},\,c} = \begin{cases} 0, & \psi_c < -\varepsilon, \\ \frac{1}{2}\left(1 + \frac{\psi_c}{\varepsilon} + \frac{1}{\pi}\sin(\frac{\pi\psi_c}{\varepsilon})\right), & |\psi_c| \leq \varepsilon, \\ 1, & \psi_c > \varepsilon, \end{cases} \tag{4.15}$$

where ε is the size of the cell distance $\varepsilon = O(\Delta x)$. The problem here is the dependence on the signed distance property, or more generally on the initial LS profile, which can deteriorate with advection causing the interface cell band to widen or narrow. Furthermore, this model is only first order accurate [47].

Geometrical phase indicator model

We [40] propose a second order accurate geometrical phase indicator that does not rely on the LS field maintaining its signed distance or initial profile. For the computation, a piecewise linear interface is reconstructed by least squares fitting. The discrete phase indicator α_c for arbitrarily shaped polyhedrons is then computed with the intersection algorithm of Kromer and Bothe [30] using the divergence theorem twice.

4.2.3. Main difficulties

This section discusses the main difficulties of the discretized equations (4.7)-(4.9) and how they are treated in this work.

As $\boldsymbol{v}^{n+1}$, p^{n+1} and ψ^{n+1} are the solution fields, the temporal, convection and diffusion terms of the momentum equation (4.9) and the convection term of the LS equation (4.9) are non-linear. Since the FVM method assembles an algebraic linear system, these terms must be linearized. The non-linear dependence is typically resolved by an iterative solution approach where the linearized quantities are updated with each iteration until convergence is reached.

Also, the transpose of the velocity gradient $(\nabla \boldsymbol{v})^T$ can't be evaluated implicitly and must be treated explicitly or updated in each iteration.

The literature has shown that high density ratios $\left|\frac{\rho^-}{\rho^+}\right| \gg 1$ or high viscosity ratios $\left|\frac{\mu^-}{\mu^+}\right| \gg 1$ are difficult for single-field NS approaches. These high ratios, combined with the abruptly changing discrete phase indicator α_h, lead to large momentum steps, which cause numerical instabilities with the *Gauss linear* or other unbounded discretizations.

Furthermore, the continuity equation (4.1) and the momentum equation (4.2) alone form a saddle-point problem, which can not be solved by any iterative method that relies on a definite system matrix [42]. Therefore, the system is reformulated as shown in the next section.

Today, productive simulations with the FVM easily scale to cell numbers of $\sim 10^6$ and more. With multiple solution variables per cell, the resulting algebraic linear system may be too large for most computer memories. This is why segregated solution algorithms have been developed in the past. Segregated methods solve one equation at a time and vector equations component-wise in a specific designed sequence. The unknowns are replaced by estimates, which are then updated throughout the iterative solution process. Block-coupled solution algorithms that solve the system at once exist, but are not currently state of the art and are therefore not used in this work. Specific for segregated solution algorithms is that there must be at least one equation to compute a solution field. Here, an explicit equation for the calculation of the pressure field is missing. In the next section, an equation for evaluating the pressure is derived. The continuity equation (4.1) does not contain pressure and is unfortunately more of a constraint.

4.2.4. Equations for the segregated solution algorithm

With the motivation for the choice of the iterative segregated solution approach in the previous section, the discrete equations for this algorithm are derived in this section.

The derived discrete equations must still be linear in order to be solved with a linear algebraic solver. This is done by linearizing the non-linear terms, replacing the unknowns $(\cdot)^{n+1}$ by estimates $(\cdot)^*$ which are updated iteratively and converge to $(\cdot)^* \to (\cdot)^{n+1}$ with iterations.

Linearizing the LS advection equation (4.9) and using the definition of flux $F_f = v_f \cdot \mathbf{S}_f$ at face f yields

$$\frac{\psi_c^{n+1} - \psi_c^n}{\delta t}|\Omega_c| + \sum_f F_f^* \psi_f^{n+1} = 0. \tag{4.16}$$

Linearizing the momentum conservation equation (4.8), and replacing all non-velocity unknowns by estimates $(\cdot)^*$ except the pressure p and the surface tension force $\boldsymbol{f}_\Sigma$ yields

$$\begin{aligned}\frac{\rho_c^* \boldsymbol{v}_c^{n+1} - \rho_c^n \boldsymbol{v}_c^n}{\delta t}|\Omega_c| + \sum_f \rho_f^* F_f^* \boldsymbol{v}_f^{n+1} - \sum_f \mu_f^* (\nabla \boldsymbol{v})_f^{n+1} \cdot \mathbf{S}_f = \\ - (\nabla p)_c^{n+1}|\Omega_c| - (\boldsymbol{g} \cdot \mathbf{x})_c (\nabla \rho)_c^* |\Omega_c| + (\boldsymbol{f}_\Sigma)_c^{n+1}|\Omega_c| + \sum_f \mu_f^* ((\nabla \boldsymbol{v})^T)_f^* \cdot \mathbf{S}_f.\end{aligned} \tag{4.17}$$

Then, inserting schemes for face values and derivatives and rewriting the momentum equation yields

$$\boldsymbol{v}_c^{n+1} + \mathbf{H}_c[\boldsymbol{v}^{n+1}] = -\mathbf{D}_c^v (\nabla p)_c^{n+1} + \mathbf{D}_c^v (\boldsymbol{f}_\Sigma)_c^{n+1} + \mathbf{B}_c^v \tag{4.18}$$

with

$$\mathbf{H}_c[v] = \frac{1}{a_c^v} \sum_{N(c)} a_N^v \boldsymbol{v}_N, \tag{4.19}$$

$$\mathbf{D}_c^v = \frac{|\Omega_c|}{a_c^v}, \tag{4.20}$$

$$\mathbf{B}_c^v = \frac{1}{a_c^v} \left(-(\boldsymbol{g} \cdot \mathbf{x})_c (\nabla \rho)_c^* |\Omega_c| + \sum_f \mu_f^* ((\nabla \boldsymbol{v})^T)_f^* \cdot \mathbf{S}_f \right). \tag{4.21}$$

Similar to staggered grids, where the velocity is defined at faces rather than cell centroids, the momentum equation (4.18) is interpolated to faces. Using the interpolation notation

$$\overline{\square}_f = w_f \square_{N_f} + (1 - w_f) \square_{O_f}, \tag{4.22}$$

and $\overline{\mathbf{D}_f^v}\,\overline{(\cdot)_f^{n+1}} = \overline{\mathbf{D}_f^v(\cdot)_f^{n+1}} + O((\Delta x)^2)$, see [42, p. 586], the face interpolated momentum equation is written as

$$\overline{\boldsymbol{v}_f^{n+1}} + \overline{\mathbf{H}_f}[\boldsymbol{v}^{n+1}] = -\overline{\mathbf{D}_f^v}\,\overline{(\nabla p)_f^{n+1}} + \overline{\mathbf{D}_c^v}\,\overline{(\boldsymbol{f}_\Sigma)_f^{n+1}} + \overline{\mathbf{B}_f^v}. \tag{4.23}$$

Insertion of the Rhie-Chow interpolation [59], defined by

$$\boldsymbol{v}_f = \overline{\boldsymbol{v}_f} - \overline{\mathbf{D}_f^v}\left((\nabla p)_f - \overline{(\nabla p)_f}\right), \tag{4.24}$$

which suppresses checkerboard oscillation, yields

$$\boldsymbol{v}_f^{n+1} + \overline{\mathbf{H}_f}[\boldsymbol{v}^{n+1}] = -\overline{\mathbf{D}_f^v}(\nabla p)_f^{n+1} + \overline{\mathbf{D}_c^v}\,\overline{(\boldsymbol{f}_\Sigma)_f^{n+1}} + \overline{\mathbf{B}_f^v} \tag{4.25}$$

This equation is then inserted into the continuity equation (4.7), yielding the pressure equation

$$-\sum_f \overline{\mathbf{D}_f^v}(\nabla p)_f^{n+1} \cdot \mathbf{S}_f = \sum_f \overline{\mathbf{H}_f}[\boldsymbol{v}^{n+1}] \cdot \mathbf{S}_f - \sum_f \overline{\mathbf{D}_c^v}\,\overline{(\boldsymbol{f}_\Sigma)_f^{n+1}} \cdot \mathbf{S}_f - \sum_f \overline{\mathbf{B}_f^v} \cdot \mathbf{S}_f. \tag{4.26}$$

Resulting equations

For the iterative segregated solution approach chosen in this work, an update equation must exist for each solution field, here $\boldsymbol{v}$, $\mathbf{p}$ and ψ. These update equations are solved only for one solution field. Other implicit fields are replaced by estimations $(\cdot)^*$, which are known from the previous iteration. The implicit field that is solved for is rewritten as $(\cdot)^{**}$. After solving the equation, the estimation $(\cdot)^*$ is updated with the solution $(\cdot)^{**}$. This iterative and segregated solution approach will converge the quantities $(\cdot)^*, (\cdot)^{**} \to (\cdot)^{n+1}$. Note that the $\mathbf{H}_c$ matrix and the $\mathbf{B}_c^v$ vector also contain estimates. Using the momentum equation (4.18) and estimating all implicit non-velocity quantities yields the so-called momentum predictor equation (4.27) to calculate the velocity $\boldsymbol{v}$. As this equation is computationally expensive to evaluate, it is usually evaluated only once or only in outer loop iterations. Therefore there is another equation for the velocity $\boldsymbol{v}$. By using estimates for all non-diagonal entries of the momentum predictor equation, the explicit velocity update equation (4.29) is formulated, which is easy to compute. For the pressure p, all implicit non-pressure quantities are replaced by estimations in the equation (4.26). This yields the final pressure equation (4.28), a discrete Poisson equation, which holds the discrete maxima principle and converges if the r.h.s. is negative [19]. It is used to calculate the pressure field for the velocity to satisfy continuity. Finally, the linearized LS advection equation (4.16) is used to update the LS field ψ with the equation (4.30).

Momentum predictor

$$\boldsymbol{v}^{**}_c + \mathbf{H}_c[\boldsymbol{v}^{**}] = -\mathbf{D}_c(\nabla p)^*_c + \mathbf{D}^v_c(\boldsymbol{f}_\Sigma)^*_c + \mathbf{B}^v_c \tag{4.27}$$

Pressure equation

$$-\sum_f \overline{\mathbf{D}^v_f}(\nabla p)^{**}_f \cdot \mathbf{S}_f = \sum_f \overline{\mathbf{H}_f}[\boldsymbol{v}^*] \cdot \mathbf{S}_f - \sum_f \overline{\mathbf{D}^v_f}\,\overline{(\boldsymbol{f}_\Sigma)^*_f} \cdot \mathbf{S}_f - \sum_f \overline{\mathbf{B}^v_f} \cdot \mathbf{S}_f \tag{4.28}$$

Velocity update

$$\boldsymbol{v}^{**}_c = -\mathbf{H}_c[\boldsymbol{v}^*] - \mathbf{D}_c(\nabla p)^*_c + \mathbf{D}^v_c(\boldsymbol{f}_\Sigma)^*_c + \mathbf{B}^v_c \tag{4.29}$$

Level Set equation

$$\frac{\psi^{**}_c - \psi^n_c}{\delta t} + \sum_f F^*_f \psi^{**}_f = 0 \tag{4.30}$$

These are the essential equations for the PISO-like segregated solution algorithm developed in this work. The concrete algorithm is described in chapter 5.2.

4.3. Surface tension force

In this section the importance of the surface tension force in the solution algorithm is discussed. Starting with a short overview of existing surface tension force models, concept of a force balanced discretization is explained and the evaluation of the interface mean curvature is discussed.

4.3.1. Surface tension force models

CSF model

As mentioned in section 2.1, the initial model for the surface tension force $\boldsymbol{f}_\Sigma$ in the momentum equation is the CSF model proposed by Brackbill et al. [4] and is defined as

$$\boldsymbol{f}_\Sigma = \sigma \kappa_\Sigma \delta_\Sigma \boldsymbol{n}_\Sigma \tag{4.31}$$

To compute $\boldsymbol{f}_\Sigma$, Brackbill et al. approximated $\delta_\Sigma \boldsymbol{n}_\Sigma \approx -\nabla\alpha$. This regularizes the singular source term $\boldsymbol{f}_\Sigma$ by spreading the force over a small band of cells.

Extended CSF model

The extended semi implicit CSF model by Hysing [24] calculates $\boldsymbol{f}_\Sigma$ as

$$\boldsymbol{f}_\Sigma^{n+1} = \sigma(\kappa_\Sigma(-\nabla\alpha))^n + \delta t\, \sigma(\Delta_\Sigma \boldsymbol{v})^{n+1}\nabla\alpha \tag{4.32}$$

with the Laplace-Beltrami operator (surface Laplacian) Δ_Σ, an explicit interface operator which is naturally not applicable for LS methods.

Integral model

The integral or tensile model was proposed by Tryggvason et al. [77] for FT methods and requires an explicit interface representation. It avoids the need to calculate curvature, which involves higher order derivatives and is generally not as accurate. For LS methods this model is not straightforward, since no explicit interface representation naturally exists. Assuming that an explicit representation for $\Sigma(t)$ is available, then the integral model calculates the surface tension of a cell Ω_c as

$$\int_{\Omega_c} \boldsymbol{f}_\Sigma \, dV = \oint_{\partial\Omega_c \cap \Sigma(t)} \sigma \mathbf{t} \times \boldsymbol{n}_\Sigma ds, \tag{4.33}$$

where $\mathbf{t}$ is a tangential vector of the intersection and s is the arc length of the intersection curve $\partial\Omega_c \cap \Sigma(t)$.

4.3.2. Concept of force balance

To reduce parasitic currents, Francois et al. [16] introduced the concept of force-balanced discretization. Considering the case of a stationary droplet without impact of gravity, the mean interface curvature κ_Σ becomes constant and the continuous momentum equation (4.2) with the CSF model (equation (4.31)) for the surface tension force $\boldsymbol{f}_\Sigma$ reduces to

$$\nabla p = \sigma \kappa_\Sigma \delta_\Sigma \boldsymbol{n}_\Sigma. \tag{4.34}$$

It states that the pressure force is equal to the surface tension force for a stationary case. If the discretized system inherits this property, the discretization and the schemes are considered to be force balanced. The discrete pressure equation (4.28) in the case of the stationary droplet reduces to

$$\sum_f (\nabla p)_f \cdot \mathbf{n}_f = \sum_f \overline{(\boldsymbol{f}_\Sigma)_f} \cdot \mathbf{n}_f. \tag{4.35}$$

With the CSF model and the approximation $\delta_\Sigma \boldsymbol{n}_\Sigma \approx -\nabla\alpha$ by Brackbill et al. [4], the equation is written as

$$\sum_f (\nabla p)_f \cdot \mathbf{n}_f = \sum_f \sigma\kappa_\Sigma \overline{(-\nabla\alpha)_f} \cdot \mathbf{n}_f. \tag{4.36}$$

For a force balanced discretization, the approximation $\overline{(-\nabla\alpha)_f} \cdot \mathbf{n}_f \approx (-\nabla\alpha)_f \cdot \mathbf{n}_f$ is used and the momentum equation can be written as

$$\sum_f (\nabla p)_f \cdot \mathbf{n}_f - \sigma\kappa_\Sigma(-\nabla\alpha)_f \cdot \mathbf{n}_f = 0. \tag{4.37}$$

If the pressure and phase indicator field uses the same surface normal gradient scheme $(\nabla(\cdot))_f \cdot \mathbf{n}_f$, error cancellation can partially occur and the discrete momentum can be written as

$$\sum_f \left((\nabla p)_f - \sigma\kappa_\Sigma(-\nabla\alpha)_f\right) \cdot \mathbf{n}_f = 0 \tag{4.38}$$

or

$$\nabla_c \cdot (\nabla p - \sigma\kappa_\Sigma(-\nabla\alpha)) = 0. \tag{4.39}$$

It states that the pressure force minus the surface tension force is cell-wise divergence free and no artificial force is generated. With that a force balanced discretization is given and parasitic currents are reduced to machine tolerance. [10, 55]

However, it is still problematic if the assumption of the CSF model, that $-\nabla\alpha$ and $\boldsymbol{n}_\Sigma$ are aligned, does not hold. In such cases, forces are not balanced, and parasitic currents occur.

4.3.3. Interface mean curvature

Using the CSF model for the surfaced tension, evaluating the interface mean curvature κ_Σ has several difficulties, which are described in the following.

Computing mean curvature

By definition, equation (3.8), the interface mean curvature is defined only at the interface, but the FVM generally works with cell-centred values, and mean curvature is needed at points not on the interface. So how to calculate the mean curvature in cell centroids that do not coincide with the interface?

The interface mean curvature of the nearest point on the interface mapped to the current position is required. At least for interface cells where $\nabla\alpha \neq 0$ holds. Assuming that the LS field satisfies the signed distance property near the interface, equation (3.11), then the LS normal field

$$\mathbf{n}(\mathbf{x}, t) = \frac{\nabla\psi(\mathbf{x}, t)}{\|\nabla\psi(\mathbf{x}, t)\|_2}, \quad \mathbf{x} \in \Omega, \tag{4.40}$$

at a point $\mathbf{x}_c$ is equal to the interface normal $\boldsymbol{n}_\Sigma$ of the nearest point on the interface. Thus the surface normal $\boldsymbol{n}_\Sigma$ can be projected in the normal direction into the domain. This defines a global mean curvature as

$$\kappa = -\nabla \cdot \mathbf{n}, \quad \mathbf{x} \in \Omega. \tag{4.41}$$

With that, cell centred mean curvature κ_c can be computed as

$$\kappa_c = \frac{1}{|\Omega_c|} \int_{\Omega_c} \kappa \, dV = -\frac{1}{|\Omega_c|} \int_{\Omega_c} \nabla \cdot \mathbf{n} \, dV = -\frac{1}{|\Omega_c|} \sum_f \mathbf{n}_f \cdot \mathbf{S}_f \tag{4.42}$$

Constant mean curvature

The problem with this global mean curvature field is that it is not constant in the $\boldsymbol{n}_\Sigma$ direction, as assumed for a force-balanced discretization without parasitic currents. As reported by Denner and van Wachem [11], parasitic currents are quadratically proportional to the mean curvature, making this a crucial problem. For demonstration, a LS field of a simple sphere is considered. The mean curvature of a sphere is given by $\kappa_{\text{sphere}} = \frac{2}{R}$. So the mean curvature at $\mathbf{x} \notin \Sigma$ is not equal to the mean curvature at $\mathbf{x} \in \Sigma$. Thus the mean curvature is not generally constant in the normal direction and Laplace equilibrium is not guaranteed [55]. Correction formulas have been proposed to avoid incorrect evaluation of the mean curvature. Kang et al. [28] proposed a distance-weighted curvature interpolation

$$\kappa_f = \frac{\kappa_{O_f}|\psi_{N_f}| + \kappa_{N_f}|\psi_{O_f}|}{|\psi_{N_f}| + |\psi_{O_f}|}, \tag{4.43}$$

with a near constant curvature approximation in the case of a spherical interface [10]. Tolle et al. [75] proposed a mean curvature correction based on the assumption of local spherical curvature

$$\kappa_{c,\,corrected} = 2\left(\frac{2}{\kappa_c} + \psi_c\right)^{-1}. \tag{4.44}$$

Accuracy

Furthermore, mean curvature evaluation becomes even more problematic if the LS field and n are deteriorated. Since the mean curvature is evaluated with the normal field n, which is evaluated with the LS field, the mean curvature is very sensitive to this. Coquerelle and Glockner [10] showed that with each derivative the order of convergence decreases by one. Thus, if the normal field $\mathbf{n}$ is of order $O(h^p)$, the cell-centred curvature κ_c will be of convergence order $O(h^{p-1})$. Therefore, high accuracy requirements are placed on the normal field n and the LS field ψ.

5. Developments and implementations

This chapter describes the main developments of this work and their implementations. The numerical method is implemented in OpenFOAM. The source code repository can be found at `https://github.com/leia-openfoam/leia` and is also archived at `https://doi.org/10.5281/zenodo.10721725`. This chapter covers the architecture of the most important classes, for details please refer to the source code documentation at `https://leia-openfoam.github.io/leia/`.

5.1. Signed Distance Preserving Level Set Method

This section describes the implementation of the SDPLS method in OpenFOAM. Throughout this work, different variants and discretizations of the source term have been tested. To cope with all the different variants and combinations, OpenFOAM offers its Runtime Type Selection (RTS) mechanism. It allows the user to specify his model selection in configuration files, here called dictionary files. The correct model is then selected at runtime, allowing subroutines being switched without recompilation. The design pattern used is OpenFOAM specific, but comes close to the factory method pattern. The model selection method, here called selector, is implemented in the base class of the model hierarchy. Thus, all hierarchy base classes have the special role of providing the RTS infrastructure and other shared functionality, while the inherited classes mostly just overload specific member functions.

5.1.1. sdplsSource

sdplsSource

The base class `sdplsSource` is the root for the model hierarchy of the SDPLS method by Fricke et al. [17]. It is the only class with which the user of the SDPLS method should interact. The SDPLS method extends the standard LS advection equation with the SDPLS source term. This is done by extending the r.h.s. of the LS equation in the solver application by calling

```
fvmsdplsSource(const volScalarField& psi, const volVectorField& U)
```

on the RTS selected model and passing a reference to the LS field `psi` and the velocity field `U`. It returns a `tmp<fvScalarMatrix>` object for integration into the LS equation. This class hierarchy is responsible for providing implementations for the formula of non-linear SDPLS source terms. It is assumed that all SDPLS source term implementation will have the following structure and evaluate the source in cell Ω_c as

$$S_{\text{SDPLS}}(\psi^{n+1}) = f_{nl}(\psi^n)\psi_c^{n+1} \tag{5.1}$$

with the known solution field ψ^n at time t^n, some non-linear scalar function $f_{nl}(\psi^n)$ and ψ_c^{n+1} the unknown solution value in cell Ω_c at time t^{n+1}. All inherited classes are responsible for implementing the explicit evaluation of the non-linear function by overloading the member function `nonLinearPart()`. This base class implements the case of an inactive source term $f_{nl} = 0$. The discretization is done separately by the class hierarchy `discretization`, which is related via the strategy pattern. Furthermore, this class has a strategy for a `Mollifier`, which restricts the active source term to a narrow band around the interface.

sdplsR

The class `sdplsR` implements the concrete SDPLS source term formula proposed by Fricke et al. [17]. As described in the base class `sdplsSource`, the explicit evaluation of the SDPLS source term instance is implemented in the member function `nonLinearPart()` as

$$f_{nl,\,R}(\psi^n) = \left\langle (\nabla \boldsymbol{v})_c \frac{(\nabla\psi)_c^n}{\|(\nabla\psi)_c^n\|_2}, \frac{(\nabla\psi)_c^n}{\|(\nabla\psi)_c^n\|_2} \right\rangle, \tag{5.2}$$

with the use of the gradient schemes specified in the `fvSchemes` dictionary.

5.1.2. discretization

discretization

The base class `discretization` forms the root for the discretization hierarchy of the SDPLS source term implemented in `sdplsSource`. For the discretization, the non-linear SDPLS source term S_{SDPLS} is linearized as

$$S_{\text{SDPLS}}(\psi^{n+1}) \approx S_c + S_p \psi_c^{n+1}, \tag{5.3}$$

where S_c is a constant and S_p is the proportionality constant for the linear part. Initializing the discretization and returning the `tmp<fvScalarMatrix>` is done by the public method

`discretize(nonLinearPart, psi)`,

which takes the `nonLinearPart` and the LS field `psi` from the source term implementing the class `sdplsSource`. Inheriting classes are responsible for implementing `Sc()` for S_c and `Sp()` for S_p, which are called by `discretize()`. This base class implements the case of no discretization with

$$S_{\text{noSource}}(\psi^{n+1}) = 0. \tag{5.4}$$

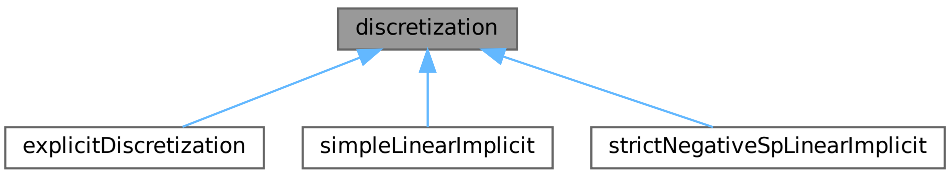


Figure 5.1.: Inheritance diagram for discretization

Three other discretization for source term are implemented, see figure 5.1.

explicitDiscretization

The class `explicitDiscretization` implements the explicit discretization of the source term. The linearization constants are calculated as

$$\begin{aligned} S_{\text{explicit},\,c} &= f_{nl}(\psi^n)\psi_c^n \\ S_{\text{explicit},\,p} &= 0 \end{aligned} \tag{5.5}$$

where the constant S_c explicitly evaluates the entire SDPLS source term. With this, the `explicit` source is written as

$$S_{\text{explicit}}(\psi^{n+1}) = f_{nl}(\psi^n)\psi_c^n \tag{5.6}$$

Without any implicit part, this discretized source term only contributes to the r.h.s. of the resulting algebraic linear system.

simpleLinearImplicit

The class `simpleLinearImplicitScheme` implements an implicit discretization of the source term in the most simplest way. The linearization constants are calculated as

$$\begin{aligned} S_{\text{simpleLinearImplicit},\,c} &= 0 \\ S_{\text{simpleLinearImplicit},\,p} &= f_{nl}(\psi^n), \end{aligned} \tag{5.7}$$

resulting in the `simpleLinearImplicit` source

$$S_{\text{simpleLinearImplicit}}(\psi^{n+1}) = f_{nl}(\psi^n)\psi_c^{n+1}. \tag{5.8}$$

This is the most accurate linearization as the approximation fits the tangent of the relation $S_{\text{SDPLS}}(\psi^{n+1}) \sim \psi^{n+1}$ [53], see the `sdplsSource` equation (5.1).

strictNegativeSpLinearImplicitScheme

The class `strictNegativeSpLinearImplicitScheme` implements an implicit discretization of the source term that ensures contributions to the diagonal dominance of the resulting algebraic linear system. The linearization constants are computed as

$$\begin{aligned} S_{\text{strictNegativeSpLinearImplicit},\,c} &= \max(f_{nl}(\psi^n), 0)\psi_c^n \\ S_{\text{strictNegativeSpLinearImplicit},\,p} &= \min(f_{nl}(\psi^n), 0), \end{aligned} \tag{5.9}$$

which ensures that the constant $S_p \leq 0$ for all cells [53]. The resulting source term, written as

$$S_{\text{strictNegativeSpLinearImplicit}}(\psi^{n+1}) = \max(f_{nl}(\psi^n), 0)\psi_c^n + \min(f_{nl}(\psi^n), 0)\psi_c^{n+1}, \quad (5.10)$$

thus switches between explicit and implicit discretization cell by cell depending on the sign of $f_{nl}(\psi^n)$. This addresses the problem that unconditional implicit treatment can cause the solution to diverge. The reason is that a positive S_p reduces the diagonal coefficient in the assembled linear system. This harms the diagonal dominance of the matrix, which then may lose its M-matrix and discrete stability property [19].

5.2. Segregated solution algorithm for two-phase flows

This section introduces the LS two-phase flow solver developed for this work. The goal of this solver is to compute a discrete converged solution for the governing non-linear PDE system, specifically the equations (4.1)-(4.6). The solution is obtained by iteratively solving the derived equations described in chapter 4.2.4 in a segregated manner similar to the PISO algorithm of Issa [25]. The pseudo code for the calculation of a time step is provided in the algorithm 1. The calculation of the solution values of the current time step t^{n+1} starts by setting the estimates $(\cdot)^*$ with the solutions of the previous time step. The outer loop (O) evolves the interface by solving the LS advection equation (4.30), computes the phase indicator α, the density field ρ and the viscosity field μ. It calculates the quantities $\mathbf{H}_c$, $\mathbf{D}_c^v$ and $\mathbf{B}_c^v$ and updates the velocity field by solving the momentum predictor equation (4.27), if specified. For consistency, all r.h.s. quantities are reconstructed from the face-centred fields of the pressure equation 4.28. Within the inner loop (I) the face centred quantities $\overline{\mathbf{H}_f}[\boldsymbol{v}^*]$, $\overline{\mathbf{D}_f^v}$, $\overline{(\boldsymbol{f}_\Sigma)_f^*}$, $\overline{\mathbf{B}_f^v}$ are calculated. Then, within the non-orthogonal correction loop (N), the pressure equation (4.28) is solved. At each iteration, the explicit cross-diffusion part of the Laplace term is updated. On the last non-orthogonal iteration the velocity field is updated with the equation (4.29).

Algorithm 1 Pseudo code of the segregated solution algorithm. The operator ':=' denotes assignment.

```
1: $\boldsymbol{v}^* := \boldsymbol{v}^n$
2: $p^* := p^n$
3: $\psi^* := \psi^n$
4: $F^* := F^n$
5: $O := 0$
6: while $O \leq O_{\text{max}}$ do
7:     Solve Level Set equation (4.30) and $\psi^* := \psi^{**}$
8:     Compute phase indicator $\alpha$
9:     Compute density $\rho$ (equation (4.12) and viscosity $\mu$ (equation 4.13)
10:    Compute $\mathbf{H}_c$, $\mathbf{D}_c^v$ and $\mathbf{B}_c^v$
11:    if predict-momentum == True then
12:        Solve momentum predictor equation 4.27 and $\boldsymbol{v}^* := \boldsymbol{v}^{**}$
13:    end if
14:    $I := 0$
15:    while $I \leq I_{\text{max}}$ do
16:        Compute $\overline{\mathbf{H}_f}[\boldsymbol{v}^*]$, $\overline{\mathbf{D}_f^v}$, $\overline{(\boldsymbol{f}_\Sigma)_f^*}$ and $\overline{\mathbf{B}_f^v}$
17:        $N := 0$
18:        while $N \leq N_{\text{max}}$ do
19:            Solve pressure equation (4.28) and $p^* := p^{**}$
20:            if $N == N_{\text{max}}$ then
21:                Solve explicit velocity update equation (4.29) and $\boldsymbol{v}^* := \boldsymbol{v}^{**}$
22:                Update volumetric fluxes $F_f^*$
23:            end if
24:            $N := N + 1$
25:        end while
26:        $I := I + 1$
27:    end while
28:    $O := O + 1$
29: end while
30: $\boldsymbol{v}^{n+1} := \boldsymbol{v}^*$
31: $p^{n+1} := p^*$
32: $\psi^{n+1} := \psi^*$
33: $F^{n+1} := F^*$
```

5.3. Surface tension force

This section describes the implemented surface tension force classes. All implementations are based on the CSF model of Brackbill et al. [4] and differ in their computing strategy.

surfaceTensionForce

The abstract base class `surfaceTensionForce` is the root for surface tension modelling. It provides the purely abstract method `surfaceTensionForce()`, which is responsible for calculating and returning $\overline{(\boldsymbol{f}_\Sigma)_f} \cdot \mathbf{n}_f$ of type `tmp<surfaceScalarField>`. This surface normal gradient is used in the pressure equation (4.28).

The inheriting classes calculate the surface tension force based on the CSF model, using the approximation $\delta_\Sigma \boldsymbol{n}_\Sigma \approx -\nabla\alpha$ by Brackbill et al. [4] and the approximation $\overline{(-\nabla\alpha)_f} \cdot \mathbf{n}_f \approx (-\nabla\alpha)_f \cdot \mathbf{n}_f$ for a force balanced discretization, which gives

$$\overline{(\boldsymbol{f}_\Sigma)_f} \cdot \mathbf{n}_f = \sigma\kappa_\Sigma(-\nabla\alpha)_f \cdot \mathbf{n}_f. \tag{5.11}$$

Since the mean curvature κ_Σ is not globally defined, it is replaced by the surface field $\overline{\kappa_f}$. This results in

$$\overline{(\boldsymbol{f}_\Sigma)_f} \cdot \mathbf{n}_f = \sigma\overline{\kappa_f}(-\nabla\alpha)_f \cdot \mathbf{n}_f. \tag{5.12}$$

The two concrete classes presented below differ in the calculation of $\overline{\kappa_f}$.

divGradPsiSnGradAlpha

The concrete class `divGradPsiSnGradAlpha` implements the calculation of the surface tension force, equation (5.12) with the definition of the global mean curvature field, equation (4.41) and is expressed as

$$\left(\overline{(\boldsymbol{f}_\Sigma)_f} \cdot \mathbf{n}_f\right)_{\text{divGradPsiSnGradAlpha}} = \sigma \overline{\left(\nabla_c \cdot \frac{(\nabla\psi)_c}{\|(\nabla\psi)_c\|_2}\right)_f} (\nabla\alpha)_f \cdot \mathbf{n}_f. \tag{5.13}$$

traceGradGradPsiSnGradAlpha

The concrete class `traceGradGradPsiSnGradAlpha` calculates the surface tension similarly to `divGradPsiSnGradAlpha`, but instead of using the discrete divergence operator $\nabla_c \cdot (\cdot)$, it uses the trace of the gradient $\mathrm{tr}(\nabla(\cdot))_c$ to calculate the mean curvature. The calculation is expressed as

$$\left(\overline{(\boldsymbol{f}_\Sigma)_f} \cdot \mathbf{n}_f\right)_{\text{traceGradGradPsiSnGradAlpha}} = \sigma\, \overline{\mathrm{tr}\left(\frac{(\nabla\nabla\psi)_c}{\|(\nabla\psi)_c\|_2}\right)_f} (\nabla\alpha)_f \cdot \mathbf{n}_f. \tag{5.14}$$

5.4. Workflow for running parameter studies

To produce the results of chapter 6, the simulation of parameter studies and the handling of post-processing data is done using the workflow and scripts presented in this section. This section can be skipped as it is not necessary for either the numerics or the understanding of the method. Nevertheless, it is not trivial to work effectively with large amounts of simulation cases and files in general. For this reason, and for reproducibility, it is worth documenting the workflow used.

The goal of the workflow is to generate a study database CSV file containing relevant postprocessing data for all cases in a study, mapping this data to their case name and study parameters. The workflow includes creating, editing, initialising, running, monitoring and postprocessing the study. The workflow is divided into the following steps of

1. creating a study directory with `study_0_create.py`,
2. editing the default files,
3. initialising the study with `study_1_init.py`,
4. running the study with `study_2_run_sbatch.py`,
5. monitoring the running study and
6. postprocessing the study with `study_3_postprocess.py`.

The Python scripts used are part of a `pip` installable Python package. It is a submodule within the source code repository at `https://github.com/leia-openfoam/leia`, but can also be found separately at `https://github.com/jnj-reitzel/pyFoamStudy`.

1. Creating a study directory with `study_0_create.py`

Having an OpenFOAM case template `CASE`, which corresponds to the standard `pyFoam` template format, and a `pyFoam` parameter file `PARAMFILE`, the script `study_0_create.py` can be called, passing them and a study name `STUDYNAME`. This script creates a directory and copies the template case, parameter file into it. This keeps the template and parameters with the studies and documents the settings. The scripts also create the file `<STUDYDIR>.info` to store information and meta data about the study. See the 5.1 listing for the Command Line Interface (CLI) of the script.

```
usage: study_0_create.py [-h] -c CASE -p PARAMFILE -s STUDYNAME

Creates the directory study_<STUDYNAME> for a pyFoam study, copies the
    template case, parameter files into it and
creates the study_<STUDYNAME>.info file with some meta data in it.

options:
  -h, --help            show this help message and exit
  -c CASE, --case CASE  Template case
  -p PARAMFILE, --parameter-file PARAMFILE
                        PyFoam parameter file
  -s STUDYNAME, --study-name STUDYNAME
                        Name of the parameter study
```

Listing 5.1: CLI of study_0_create_empty.py

2. Editing the default files

In the second step, the user can manually edit the template case and parameter file within the study directory. This leaves the original template case untouched and does not sneak in temporary settings between the default template settings that may be forgotten to revert.

3. Initialising the study with `study_1_init.py`

The third step is to instantiate the concrete simulation cases from the template using the script `study_1_init.py`. It is basically a wrapper around `FoamRunParameterVariation`, but also creates some additional files like `<STUDYDIR>.cases`, a file listing all concrete cases, and <STUDYDIR>.json, a Python readable dictionary file mapping the concrete cases to their concrete parameters. Only the path to the study directory needs to be passed, other relevant information for the script is read from the <STUDYDIR>.info file. The script ends by appending information to the <STUDYDIR>.info file. The listing 5.2 shows the CLI of the script.

```
usage: study_1_init.py [-h] STUDYDIR

Initialises a parameter study, basically using `pyFoamRunParameterVariation.
    py`.
The script takes the study directory STUDYDIR created by `study_0_create.py`
and generates concrete simulation cases and some meta files.
No meshes are generated, only the pyFoam instantiations with some extra meta
    data files.

positional arguments:
  STUDYDIR    Study directory with the template case, parameter file and the
    info file inside.

options:
  -h, --help  show this help message and exit
```

Listing 5.2: CLI of study_1_init.py

4. Running the study with `study_2_run_sbatch.py`

The fourth step is to run the study. More specifically, the study cases are submitted to the Slurm job scheduler using the `study_2_run_sbatch.py` script. Alternatively, the sibling script `study_2_run_foamJob.py` can be used to run the study locally with `foamJob`. The script is passed the study directory STUDYDIR and the relative path within a case to slurm or shell script specifying preprocessing, execution and postprocessing. The script reads the cases to run from the <STUDYDIR>.cases file, which is known from reading the <STUDYDIR>.info file. It is recommended to have the ALLRUN script inside the template case, and thus in every case instance, to have self-documenting cases without the need to call local scripts from outside. See the listing 5.3 for the CLI of the script `study_2_run_sbatch.py`.

```
usage: study_2_run_sbatch.py [-h] STUDYDIR ALLRUN

Submitts the simulations of the STUDYDIR study to Slurm using `sbatch`.
It submits the specified Slurm ALLRUN script for all cases listed in <
    STUDYDIR>.cases.
The ALLRUN script is in each case.

positional arguments:
  STUDYDIR    Study directory with the template case, parameter file and the
    info file inside.
  ALLRUN      Slurm script that will be submitted and contains preprocessing,
     mesh building and execution.

options:
  -h, --help  show this help message and exit
```

Listing 5.3: CLI of study_2_run_sbatch.py

5. Monitoring the running study

Monitoring the progress of the simulations, or whether the study has run successfully, is important for the effective management of many studies. Therefore, it is mentioned that the Python package provides the scripts `study_print-latestTime.py`, which lists the latest time directories of all cases, and `study_print-status.py`, which infers the status of all cases in the study. Otherwise, it would be cumbersome to verify the success of a study or to identify cases with errors.

6. Postprocessing the study with `study_3_postprocess.py`

Once all study simulations have run successfully, post-processing is performed using the script `study_3_postprocess.py`. This script collects relevant postprocessing data from CSV files in the cases and metadata of the study and builds the study database CSV file. The study directory `STUDYDIR` is passed to the script and the remaining information is retrieved from the information file `<STUDYDIR>.info`. The CLI interface is shown in the listing 5.4.

```
usage: study_3_postprocess.py [-h] [--skip-convergence] [-n DATABASE]
    STUDYDIR

Script merges and concatenates case specific ['leiaLevelSetFoam.csv', '
    gradPsiError.csv', 'TVerror.csv'] CSV files into one large database CSV
    file,
lists the latest times of all cases and
if the study is investigating refinement, the convergence rates of all the
    error properties are calculated and added to the CSV database file.

Note:
Run this script from within the directory where the actual study cases reside
    .

positional arguments:
  STUDYDIR              Study directory with the template case, parameter
    file and the info file inside.

options:
  -h, --help            show this help message and exit
  --skip-convergence    Skip calculation of convergence rates
  -n DATABASE, --name DATABASE
                        Provide a different database CSV file name. Default:
    <STUDYDIR>_database.csv
```

Listing 5.4: CLI of study_3_postprocess.py

Miscellaneous scripts

The Python package contains other scripts that are not part of the workflow, but support working with studies and database CSV files. All plots in this work are created using the `database_plot.py` script, which takes a study database CSV file and has several options for manipulation. The script `database_concat.py` concatenates two or more study database CSV files if they have the same columns, otherwise it throws an error. If one wants to concatenate study database CSV files that do not have matching columns, the following two scripts are useful. The `database_add-column.py` script can add a column filled with a specified value to a study database CSV file. This allows unlisted study settings to be added to a study database CSV file to match the columns of another study database CSV file. The `database_filter.py` script can be used to filter out rows or columns in study database CSV files. The last important script, `study_rsync.py`, is a wrapper around `rsync` with settings to synchronise study directories without concrete cases. This is useful for copying the study directory with meta and postprocessing data from one machine to another, or for archiving purposes.

6. Numerical results

This chapter presents numerical results for various simulations, focusing on pure advection using the LS method on different mesh types, pure advection using the SDPLS method, and two-phase flow simulations using the developed solver for the unstructured Finite Volume Level Set method.

Section 6.1 provides details on the numerical settings, evaluated error quantities and test cases considered in this study. Section 6.2 investigates the performance of different advection schemes for pure LS advection over different test cases and mesh types. The results for the SDPLS method are presented in section 6.3. Finally, the last section 6.4 presents the results for the stationary droplet test case with the developed two-phase flow solver.

6.1. Setups

This section shows numerical setups, test cases and error quantities. All test cases share the same LS field initialisation. They all start with a spherical droplet, initialised with the following implicit equation

$$\psi(\mathbf{x}, 0) = \|\mathbf{x} - \mathbf{x}_0\|_2 - R. \tag{6.1}$$

Furthermore, all cases use a zero gradient boundary condition for the LS field, a time step size δt satisfying CFL $= 0.3$ and the abrupt geometrical phase indicator presented by us [40]. Most simulations are performed on unstructured orthogonal hexahedral meshes created with the `blockMesh` application from OpenFOAM. Perturbed hexahedrons and polyhedrons are used as non-optimal meshes. Perturbed hexahedral meshes have an average mesh non-orthogonality of $4.2\,°$, a maximum mesh non-orthogonality of $14.2\,°$ and a maximum skewness of 0.17. The used polyhedral meshes are created with the OpenFOAM sub-module `cfmesh`. For all pure LS advection test cases, the Crank-Nicolson method is used to discretize the temporal term. Advection schemes are varied.

6.1.1. Test cases

2D contact line

In the 2D contact line advection test case used by Fricke et al. [17], a droplet (part of a sphere) is placed at the wall, similar to the sketch in figure 2.1. The implicit sphere, equation (6.1), with radius 0.3 is placed with its centre at $(0.5 \ -0.15)$ inside a domain of 1×0.5. The velocity field is time-periodic and is given by

$$\boldsymbol{v}(t, \mathbf{x}) = \cos\left(\frac{\pi t}{\tau}\right)\begin{pmatrix} v_0 + c_1 x + c_2 y \\ -c_1 y \end{pmatrix} \tag{6.2}$$

with $v_0 = -0.2,\ c_1 = 0.1,\ c_2 = -2,\ \tau = 0.4$ and $t^{end} = 0.8$. The test case was used by Fricke et al. to verify their SDPLS method. As in the original setup, a time step size of δt is used to satisfy CFL $= 0.5$. The smooth Heaviside phase indication method is used in this case. Figure 6.1 shows the LS field with a blue—red colour map, the interface at the zero LS as the black line and the velocity field with the grey arrows at time $t = 0$.

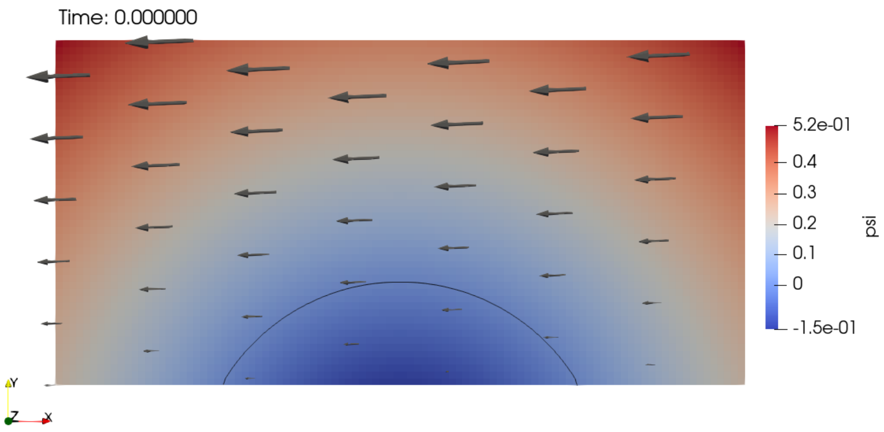


Figure 6.1.: LS (psi) and velocity field in the 2D contact line case. The black contour marks the interface. The grey arrows point in the direction of the velocity field where the size indicates the magnitude.

3D rotation

The 3D rotation advection test case is an extension of the canonical 2D rotation advection test case. Here a sphere with radius 0.15 is placed at $(0.5\ 0.5\ 0.75)$ with its centre inside a domain of $1 \times 1 \times 1$. A stationary velocity field is given by

$$\boldsymbol{v}(\mathbf{x}) = \boldsymbol{\omega} \times (\mathbf{x} - \mathbf{x}_0) \quad \text{with} \quad \boldsymbol{\omega} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \ \mathbf{x}_0 = \begin{pmatrix} 0.5 \\ 0.5 \\ 0.5 \end{pmatrix}. \tag{6.3}$$

This field rotates the eccentrically placed droplet around the domain centre $\mathbf{x}_0$ without any deformation. The simulation ends at $t^{end} = 2\pi$ when the droplet has reached its initial position. Figure 6.2 shows the domain of the text case with the droplet in blue at different times. The LS field at time $t = 0$ is visualised by the blue–red colour map and the stationary velocity field by the black arrows.

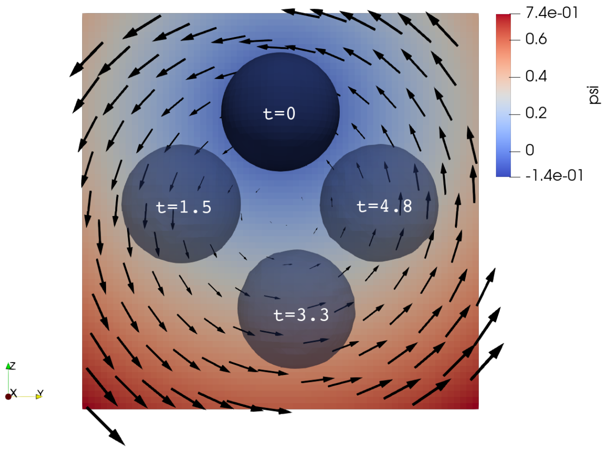


Figure 6.2.: 3D rotation LS field (psi) and velocity field (black arrows)

3D shear

The 3D shear advection test case introduced by Liovic et al. [33] is a three dimensional extension of the 2D shear test case originally introduced by Leveque [32]. A sphere of radius 0.15 is placed with its centre at $(0.35\ 0.75\ 0.25)$ within a domain of $1 \times 1 \times 2$. The velocity field consists of a vortex in the xy-plane with a parabolic channel flow in the z-dimension and is given by the following equation

$$\boldsymbol{v}(\mathbf{x}, t) = \cos\left(\frac{\pi t}{\tau}\right) \begin{pmatrix} \sin(2\pi y)\sin^2(\pi x) \\ \sin(2\pi x)\sin^2(\pi y) \\ U_{max}\left(1 - \frac{r}{R}\right)^2 \end{pmatrix}. \tag{6.4}$$

The velocity field, which oscillates with time t, has a maximum velocity of $U_{max} = 1$ and a period duration of $\tau = 3$, which is also the end time $t^{end} = 3$. Figure 6.3 shows the deformed droplet at different times for the 3D shear test case.

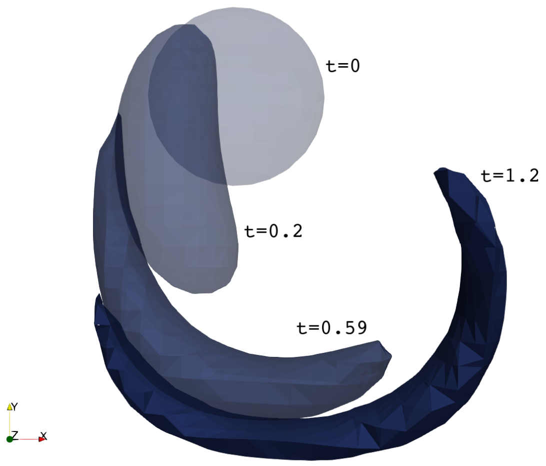


Figure 6.3.: 3D shear test case

3D deformation

The 3D deformation field advection test case proposed by LeVeque [32] combines a deformation in the xy plane with one in the xz plane. Within a domain of $1 \times 1 \times 1$ a sphere of radius 0.15 is placed with its centre at $(0.35\ 0.35\ 0.35)$. The velocity field is given by

$$\boldsymbol{v}(t, \mathbf{x}) = \cos\left(\frac{\pi t}{\tau}\right) \begin{pmatrix} 2\sin(2\pi y)\sin^2(\pi x)\sin(2\pi z) \\ -\sin(2\pi x)\sin^2(\pi y)\sin(2\pi z) \\ -\sin(2\pi x)\sin(2\pi y)\sin^2(\pi z) \end{pmatrix}, \tag{6.5}$$

with $\tau = 3$ and $t^{end} = 3$. The field entrains the sphere through two rotating vortices, creating a stretched surface. Figure 6.4 shows the deformed droplet at time $t = 0.9$ for the 3D deformation test case.

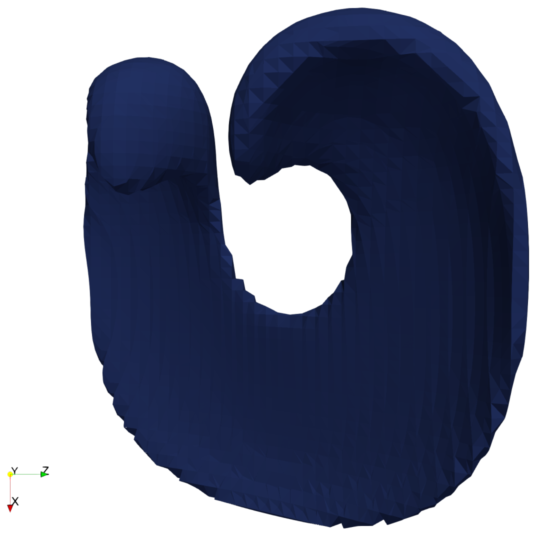


Figure 6.4.: Spherical droplet under 3D deformation velocity field at $t = 0.9$

3D stationary droplet

The 3D stationary droplet test case is a two-phase flow test case and is used to evaluate the performance of the LS NS coupling and surface tension force models. A spherical droplet of radius 0.001 is initialized at the centre of a $0.01 \times 0.01 \times 0.01$ domain. The velocity is initialized uniformly to zero within the domain and zero gradient boundary conditions are used. The dynamic pressure is uniformly initialized to zero and fixed at the boundaries. Although the solution to this two-phase flow case is stationary, this test case is very challenging. Violations of the discrete force balance, as presented in chapter 4.3.2, become visible as so-called parasitic currents. For quasi-stationary flows it is crucial to avoid parasitic currents as they can be large relative to the flow velocities and thus have a huge impact on the solution. Parasitic currents in the stationary droplet test case are measured by the magnitude of the maximum velocity $\max(|\mathbf{v}|)$ in the flow field.

6.1.2. Error quantities

The following error quantities are evaluated with the test cases. The geometrical error E_g, defined in equation (6.6), compares the predicted volume averaged phase indicator field at the end time $\alpha_c(t^{end})$ with the correct known field $\alpha_c(t^0)$. The correct field is known for simple test cases where the end state is equal to the initial state, such as the 3D rotation test case, or for cases with an oscillating velocity field that deforms a droplet back and forth into the initial state. The geometrical error can be interpreted as how much of the characteristic phase is misplaced.

$$E_g = \sum_{c \in C} |\Omega_c||\alpha_c(t^{end}) - \alpha_c(t^0)| \tag{6.6}$$

The relative volume conservation error E_v, defined in equation (6.7), is a measure of the relative volume loss compared to the initial state.

$$E_v(t) = \frac{\left|\sum_{c \in C} \alpha_c(t)|\Omega_c| - \sum_{c \in C} \alpha_c(t^0)|\Omega_c|\right|}{\left|\sum_{c \in C} \alpha_c(t^0)|\Omega_c|\right|} \tag{6.7}$$

The signed distance error $E_{\nabla\psi}$, defined in equation (6.8), measures how much the LS field at cells $c \in C_{narrow}$ within a narrow band C_{narrow} around the interface differs from the signed distance property, equation (3.11). The chosen narrow band has a width of three cells, consisting of the cell intersected by the interface and its immediate neighbours.

$$E_{\nabla\psi}(t) = \frac{1}{N_{C_{narrow}}} \sum_{c \in C_{narrow}} (|\|(\nabla\psi)_c(t)\|_2 - 1|) \tag{6.8}$$

The following sections show the convergence of these error quantities over different mesh resolutions. A mesh resolution is represented by the characteristic grid spacing h, which is the mean value of the distances between centroids and is defined as

$$h = \frac{1}{N_f} \sum_{f \in F} \|\mathbf{x}_{N_f} - \mathbf{x}_{O_f}\|_2. \tag{6.9}$$

6.2. Advection schemes

In this section a comprehensive evaluation of different advection schemes for LS advection with the equation 3.5 is performed. The discretization of the advection term of any PDE is a major challenge in CFD. It greatly affects the stability and accuracy of the method. The LS advection equation is a pure advection equation without a diffusive term, so the advection scheme has an even more important role and the following preliminary studies on advection schemes are carried out.

The following studies include the standard second order accurate *linear* scheme, the upwind biased second order accurate *LinearUpwind* scheme and first order *upwind* scheme. In addition, the TVD schemes *vanLeer*, *OSPRE*, *SuperBee* and *vanAlbada* from the group of TVD schemes are tested. Similarly, *Gamma 1*, *MUSCL*, *Minmod*, *SFCD* and *UMIST* are tested from the group of NVD schemes. The 3D rotation test case is examined because it does not use an oscillating velocity field that might compensate for advection errors. In addition, the rotationally symmetric velocity field is a challenge for any mesh type as the face normals are generally not aligned with the local velocity field. The oscillating 3D deformation and 3D shear test cases are also investigated as examples of complex flow fields and deformations. The following plots mainly show the convergence of the geometrical errors, as the geometrical error is more sensitive than the volume conservation error and thus better captures the accuracy of an advection scheme.

3D rotation

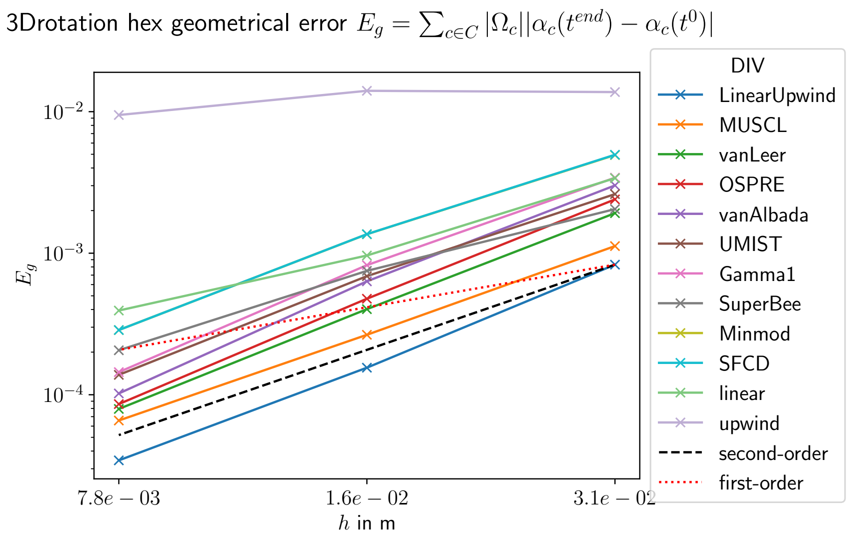


Figure 6.5.: Geometrical errors of the advection schemes with the LS method for the 3D rotation test case on hexahedral meshes

Starting with the 3D rotation case, figures 6.5, 6.6 and 6.7 show the convergence of the geometrical error E_g, equation (6.6), over the characteristic grid spacing h of all advection schemes on hexahedral, perturbed hexahedral and polyhedral meshes. On all mesh types, the *LinearUpwind* and *MUSCL* advection schemes show the best geometrical errors and second order convergence. The prominent *linear* scheme performs second worst. It has an error of 3.9×10^{-4} at the finest resolution and shows a convergence order of only 1.5. Simulations with the *linear*, *Gamma1* and *SFCD* schemes even crashed on polyhedral meshes. The volume errors only confirm these results and are therefore included in the appendix.

3D deformation

For the 3D deformation test case on a hexahedral mesh, it can be seen in figure 6.8 that the *linear* scheme dominates all other schemes by more than three orders of magnitude and shows second order convergence. It is clearly the most accurate scheme for this case.

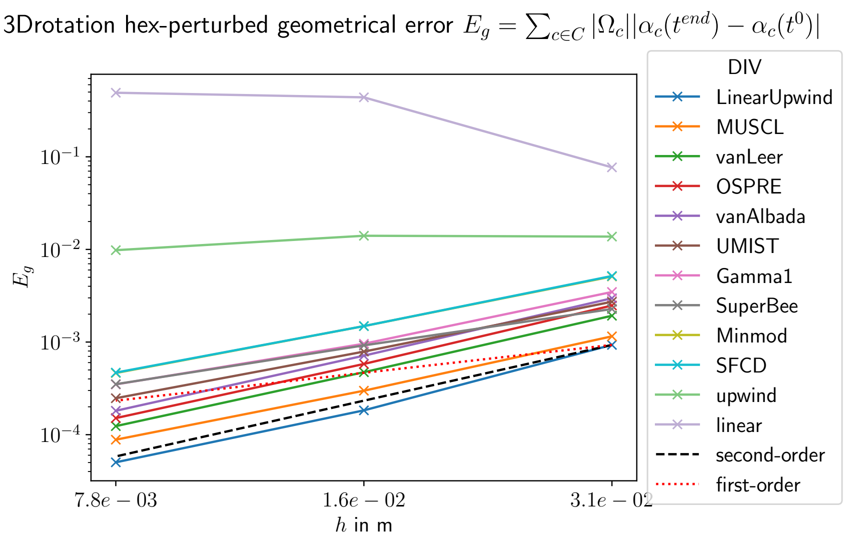


Figure 6.6.: Geometrical errors of the advection schemes with the LS method for the 3D rotation test case on perturbed hexahedral meshes

As the lines of the other schemes are not clearly visible, figure 6.9 shows the same plot but with the *linear* scheme filtered out. Then the *Superbee*, followed by *MUSCL* and *LinearUpwind* schemes perform best. Unlike the case of 3D rotation, the convergence order of *MUSCL* and *LinearUpwind* is not second order, but of the order of ≈ 0.75. Similar performance of the *linear* scheme can be seen on perturbed hexahedral meshes, figure

This clear dominance of the *linear* scheme is surprising as it clearly failed for the geometrical and volume conservation error in the 3D rotation test case. The evaluation of the geometrical error is done at the last time step t^{end}, where the state should correspond to the initial state. Unlike the 3D rotation test case, the 3D deformation test case ensures this by reversing the flow field. This oscillating velocity field can cause error cancellation and thus affect the geometrical error. To verify the performance of the *linear* scheme, its volume conservation error E_v is discussed next. The volume conservation error is evaluated over all times. It is therefore not influenced by error cancellation due to an oscillating velocity field for all times $t \leq \tau/2$. Figure 6.12 shows the volume conservation error E_v over all times for the 3D deformation test case with different advection schemes on hexahedral meshes with the finest resolution of 128 cells per dimension leading to $h = 7.8 \times 10^{-3}$. It can be seen that with the *linear* advection scheme, error cancellation occurs after time $t = \tau/2$, but it also shows the best volume conservation at all times

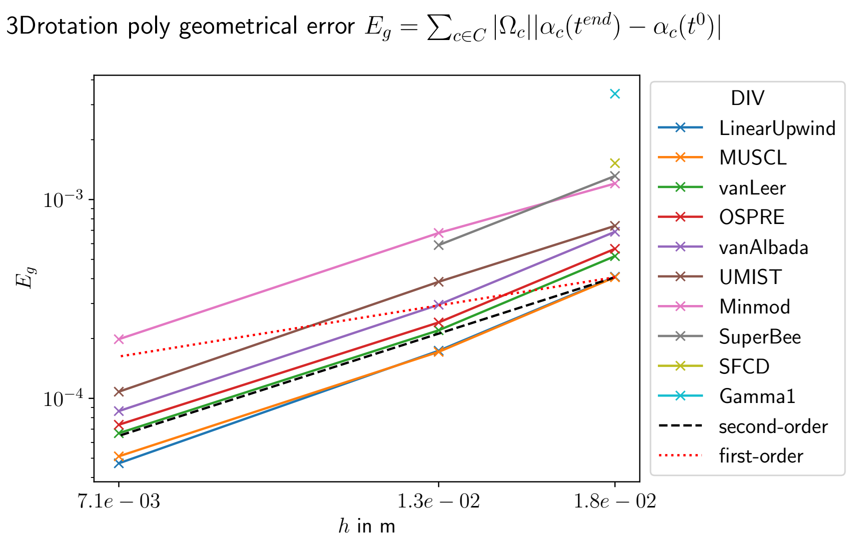


Figure 6.7.: Geometrical errors of the advection schemes with the LS method for the 3D rotation test case on polyhedral meshes

before that. Convergence plots for the volume conservation error for all mesh types are in the appendix. They all show that the *linear* scheme conserves volume the best and indeed performs well with the 3D deformation test case. This is not in contradiction with the literature, which states that the *linear* scheme produces oscillations near the interface [45, 3, 83]. Figure 6.13 shows the signed distance errors of all schemes over all resolutions. Despite the high accuracy of the geometrical and volume conservation errors, the *linear* scheme has the worst signed distance errors. At the finest mesh resolution its signed distance error is 9 and is four error points above *LinearUpwind*, probably due to oscillations with the *linear* scheme.

3D shear

For the 3D shear test case with hexahedral meshes, figure 6.14, the *linear* scheme also shows the lowest geometrical errors and second order convergence. *LinearUpwind* shows only a convergence order of 1.75. Volume conservation is also led by the *linear* scheme. Its convergence plot can be found in the appendix.

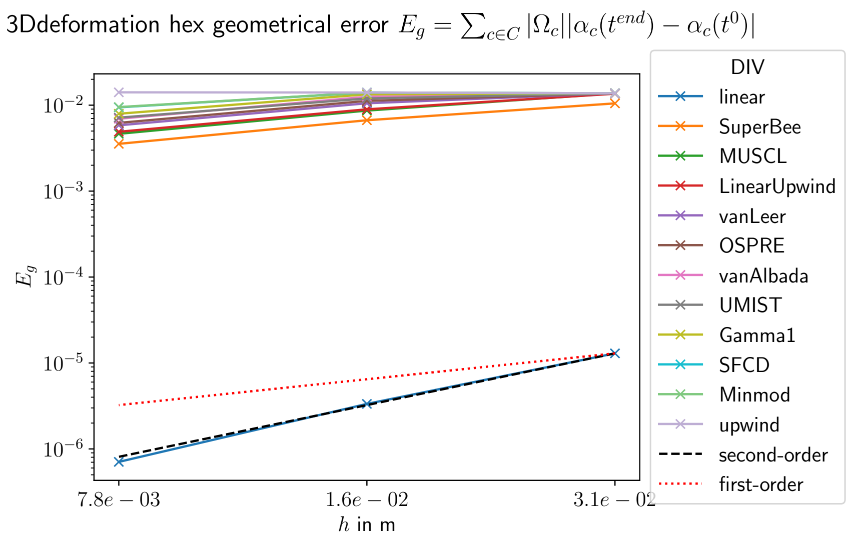


Figure 6.8.: Geometrical errors of the advection schemes with the LS method for the 3D deformation test case on hexahedral meshes

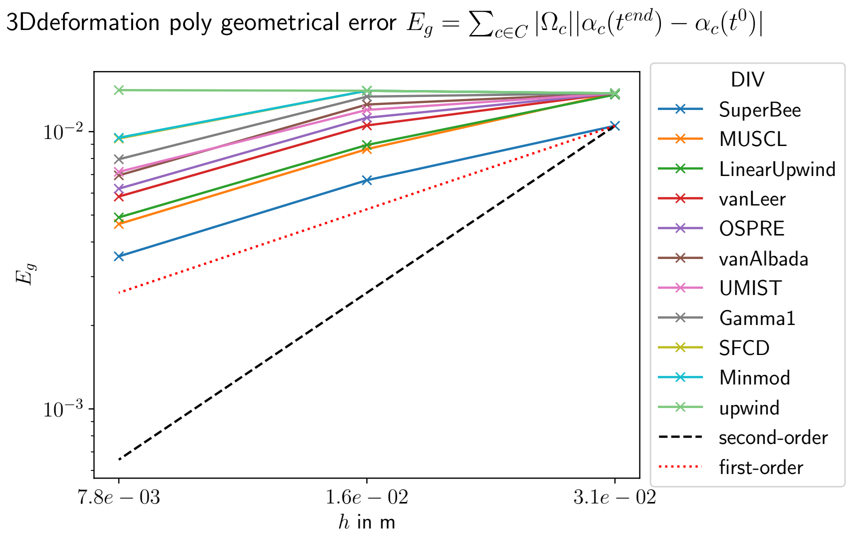


Figure 6.9.: Geometrical errors of the advection schemes without *linear* with the LS method for the 3D deformation test case on hexahedral meshes

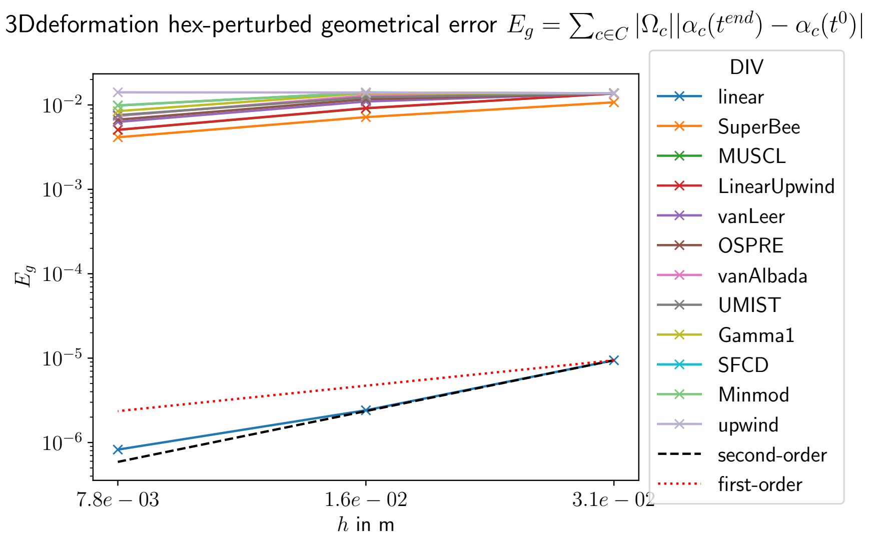


Figure 6.10.: Geometrical errors of the advection schemes with the LS method for the 3D deformation test case on perturbed hexahedral meshes

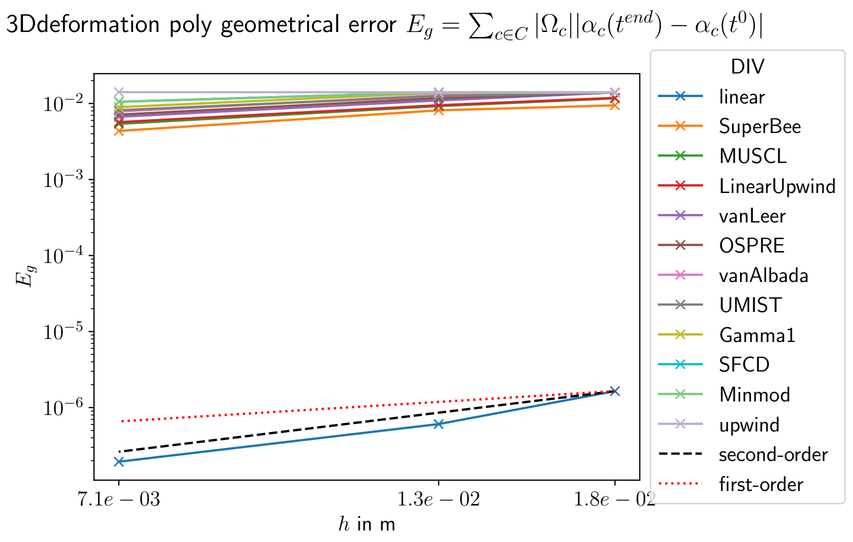


Figure 6.11.: Geometrical errors of the advection schemes with the LS method for the 3D deformation test case on polyhedral meshes

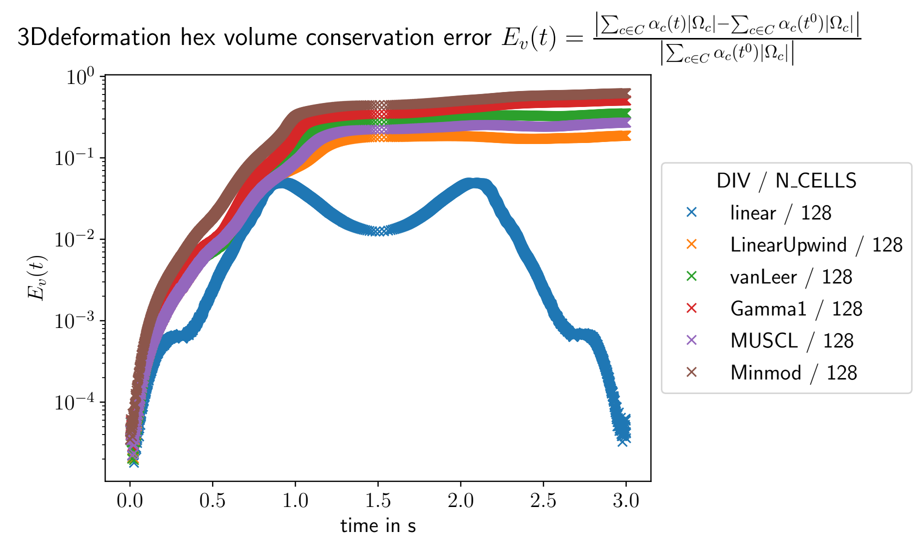


Figure 6.12.: Volume conservation errors over time of *linear*, *LinearUpwind*, *vanLeer*, *Gamm1*, *MUSCL* and the *Minmod* advection schemes with the LS method for the 3D deformation test case on polyhedral meshes

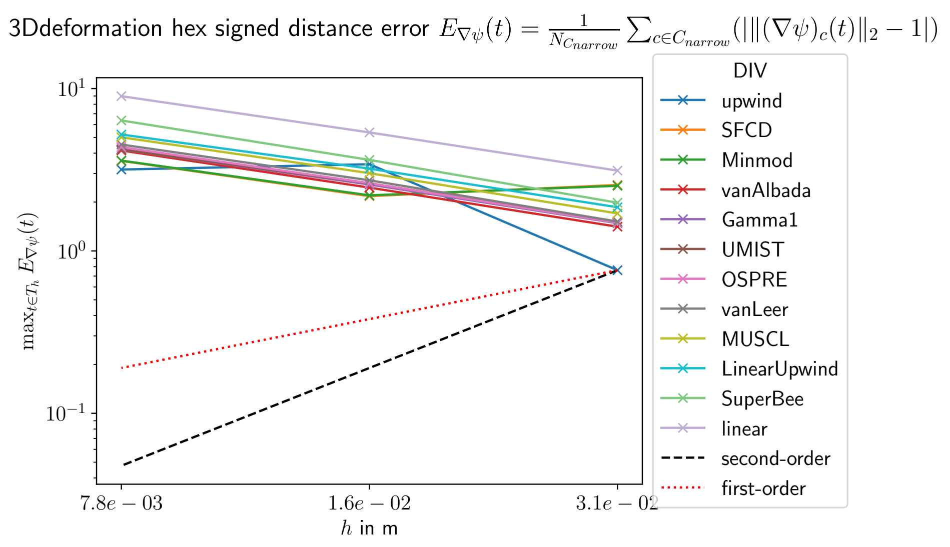


Figure 6.13.: Signed distance errors of all advection schemes with the LS method for the 3D deformation test case on hexahedral meshes

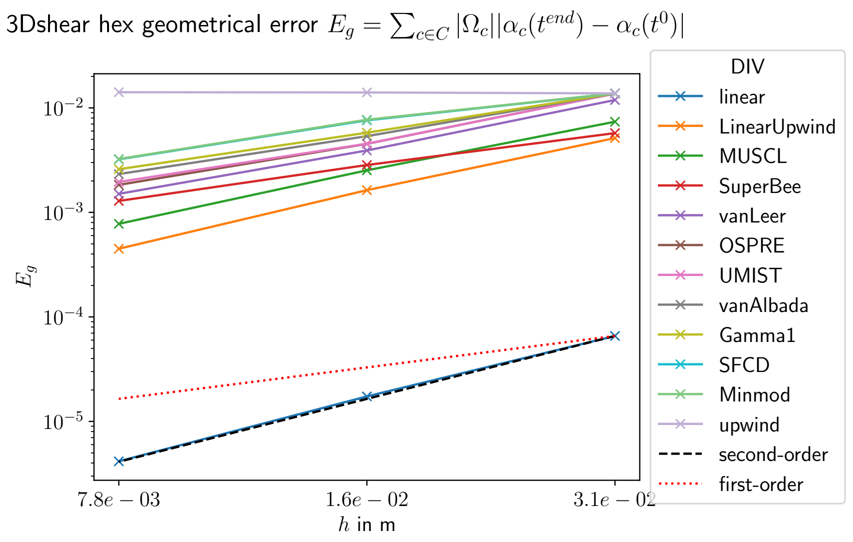


Figure 6.14.: Geometrical errors of best advection schemes with the LS method for the 3D shear test case on hexahedral meshes

6.3. Signed Distance Preserving Level Set Method

In this section, the implemented SDPLS method is tested. The 2D contact line, 3D deformation and 3D shear test cases are investigated with the *linear* or *LinearUpwind* advection schemes, as they performed best for the LS advection.

6.3.1. Verification with Fricke et al.

In this subsection, the implemented SDPLS method is verified with the 2D contact line test case by Fricke et al. [17].

Figure 6.15, compares the geometric error E_g for the SDPLS method and the plain LS advection. The method variation is denoted by the strategy chosen for `SDPLS_SOURCE`. The SDPLS method is denoted by `R` and the simple LS advection is denoted by `noSource`. `MOLLIFIER m1` enables the mollifier proposed by Fricke et al. to restrict the active source term to the vicinity of the interface. For advection, the *LinearUpwind* scheme is selected. The SDPLS source term is discretized with *simpleLinearImplicit*. It can be seen that the LS method performs similarly with and without the SDPLS source term. Both converge to second order, although the absolute error with the source term is slightly worse. Comparing the signed distance error $E_{\nabla\psi}$ in figure 6.16 it can be seen that without source term the signed distance is not conserved, but with source term it is conserved with first order convergence. These results are in agreement with the geometric results, namely contact line position, contact angle and curvature, and the signed distance results of Fricke et al.

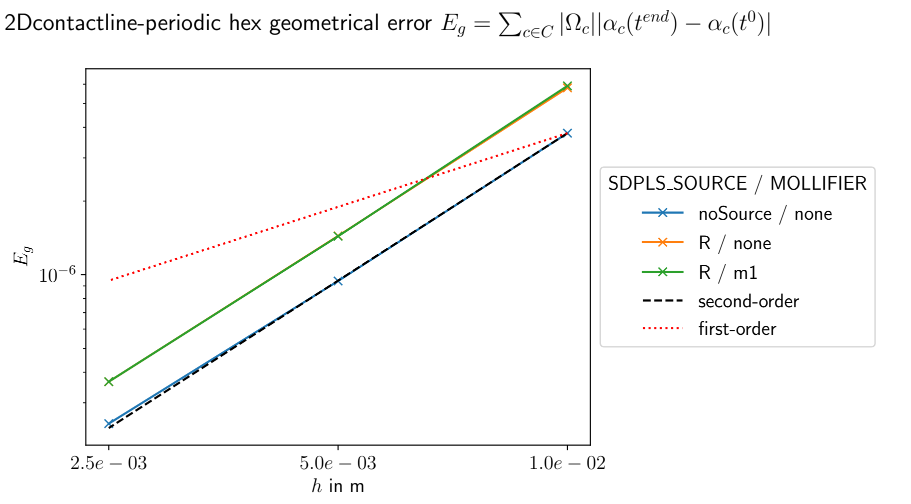


Figure 6.15.: Geometrical errors of the SDPLS method compared to *noSource*, the plain LS method, with *LinearUpwind* for the 3D deformation test case on hexahedral meshes

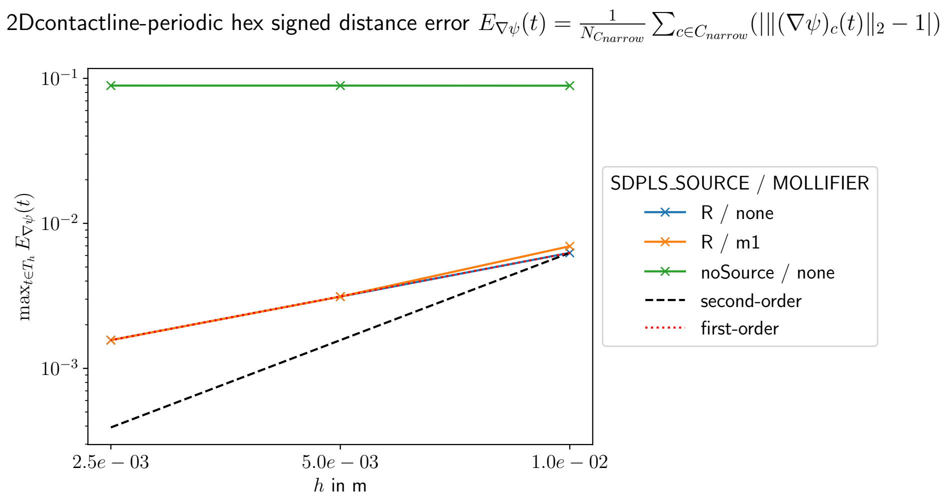


Figure 6.16.: Signed distance errors of the SDPLS method compared to *noSource*, the plain LS method, with *LinearUpwind* for the 3D deformation test case on hexahedral meshes

6.3.2. Different discretizations for the SDPLS source term

The studies presented in this section investigate the influence of the SDPLS source term discretization on the SDPLS method and compare their geometrical errors E_g and signed distance errors $E_{\nabla\psi}$ with those of the plain LS method for the more complex 3D deformation and 3D shear test cases.

3D deformation

Starting with the case of 3D deformation with *LinearUpwind* advection scheme, figure 6.17 shows the geometrical error Eg of the SDPLS method with different discretizations and the plain LS method. The different discretizations are denoted by the strategy chosen for `SOURCE_SCHEME`. It can be seen, that *simpleLinearImplicit* without a mollifier has the lowest geometrical error, even better than *noSource*. Both, *noSource* and SDPLS *simpleLinearImplicit*, show first order convergence.

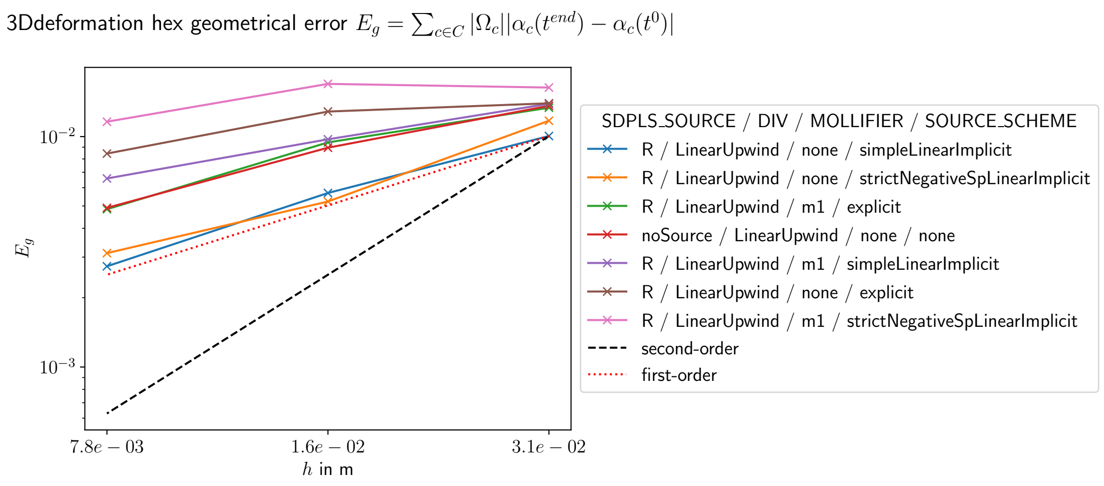


Figure 6.17.: Geometrical errors of different discretizations of the source term of the SDPLS method with *LinearUpwind* for the 3D deformation test case on hexahedral meshes

Figure 6.18 shows the signed distance error $E_{\nabla\psi}$. The SDPLS method with discretization *simpleLinearImplicit* without a mollifier, shows the best signed distance error of 0.3 with a convergence order of 0.2. All other cases show a divergence.

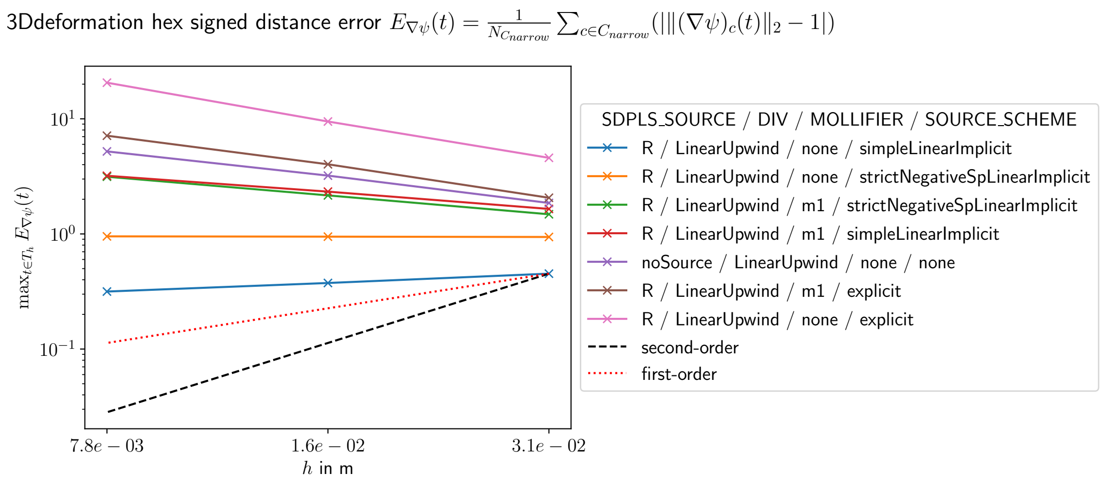


Figure 6.18.: Signed distance errors of different discretizations of the source term of the SDPLS method with *LinearUpwind* for the 3D deformation test case on hexahedral meshes

The same comparison but with the *linear* advection scheme is shown next. Figure 6.19 shows the geometrical error E_g of different SDPLS source term discretizations for the 3D deformation case. The plain LS shows the best geometrical error by more than two orders of magnitude at the finest resolution. The best SDPLS configurations are *simpleLinearImplicit* and *explicit* with mollifier. Besides having a much larger geometrical error compared to *noSoure*, the second order convergence is reduced to 0.8 for *simpleLinearImplicit*.

Figure 6.20 shows the signed distance error $E_{\nabla\psi}$ for this comparison. Here another SDPLS configuration, *strictNegativeSpLinearImplicit* without mollifier, shows the best performance compared to all other discretization and *noSource*. But it is still in the magnitude of 1 and shows a slight divergence like all the other cases.

3D shear

The performance of different SDPLS source term discretization compared to the plain LS method for the 3D shear test case is shown next. Starting with the study using the *LinearUpwind* advection scheme, figure 6.21 shows the plain LS method with the best

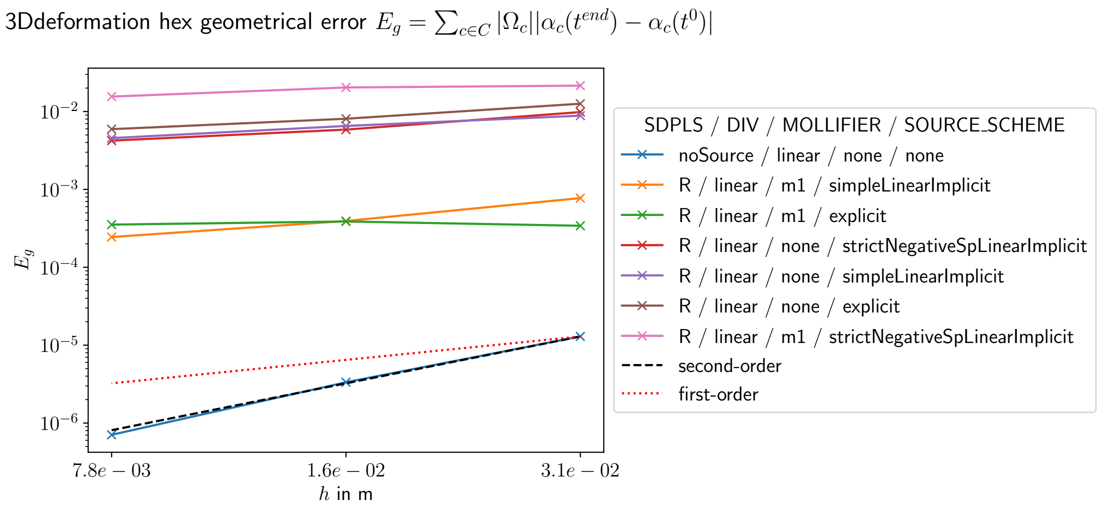


Figure 6.19.: Geometrical errors of different discretizations of the source term of the SDPLS method with *linear* for the 3D deformation test case on hexahedral meshes

geometrical error and an order of convergence of 1.9. Half an order of magnitude behind are the cases of the SDPLS method with *strictNegativeSpLinearImplicit* and *simpleLinearImplicit* discretization and no mollifier with 1.3 and 1.0 local order convergence at the two finer resolutions.

The signed distance errors are shown in figure 6.22. As for the 3D deformation test case, the SDPLS discretization *simpleLinearImplicit* without mollifier shows the best signed distance error of 0.2 at the finest resolution with an order of convergence of 0.5.

The 3D shear comparison with the *linear* advection scheme is shown next. Figure 6.23 shows the geometrical error E_g. As for the 3D deformation case with the *linear* advection scheme, the plain LS clearly shows the best geometrical error. The SDPLS method is more than two orders of magnitude behind the *simpleLinearImplicit* and *explicit* with mollifier discretization, also with reduced order of convergence.

The signed distance error $E_{\nabla\psi}$ is shown in figure 6.24. Here the SDPLS source term discretizations *strictNegativeSpLinearImplicit* and *simpleLinearImplicit* with mollifier, show the lowest signed distance error of magnitude 1 constant over all resolutions. The plain LS method, with the third lowest signed distance error, shares this constant behaviour over all resolutions, but is only 0.1 more off.

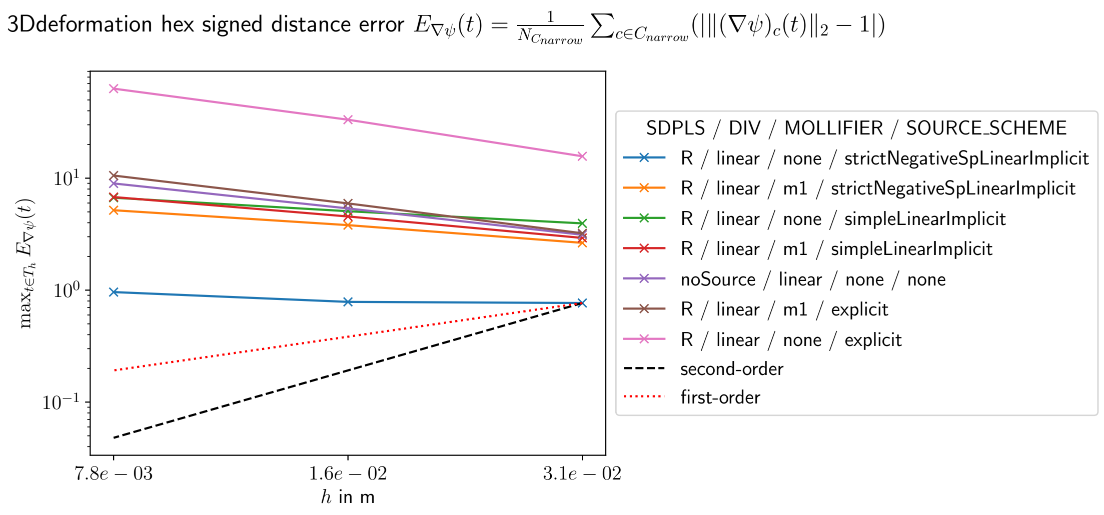


Figure 6.20.: Signed distance errors of different discretizations of the source term of the SDPLS method with *linear* for the 3D deformation test case on hexahedral meshes

Summary

In summary, for the 3D deformation test case with *LinearUpwind* advection, the discretization *simpleLinearImplicit* without mollifier performs clearly best and better than the plain LS method for both the geometrical and the signed distance error. It has a signed distance error of 0.3 with a convergence order of 0.2, while the plain LS method has an error of 5 and diverges. Its geometrical error is 3×10^{-3} better than for the plain LS method with 5×10^{-3}, while sharing the first order convergence. For the 3D shear test case with *LinearUpwind* advection, the discretization *simpleLinearImplicit* without mollifier also shows the best results for the SDPLS method. It improves the signed distance error to 0.2 with a convergence order of 0.5, while the plain LS method shows an error of 2 and diverges. For the geometrical error, the SDPLS method does not perform better than the plain LS method. It has an error of 1×10^{-3} with a convergence order of 0.7. The plain LS method has an error of 5×10^{-4} with a convergence order of 1.9.

For the *linear* advection scheme, the results of the SDPLS method are not as competitive or clear. For the 3D deformation case, the SDPLS source term discretization *strictNegativeSpLinearIplicit* without mollifier shows significant improvements in the signed distance,

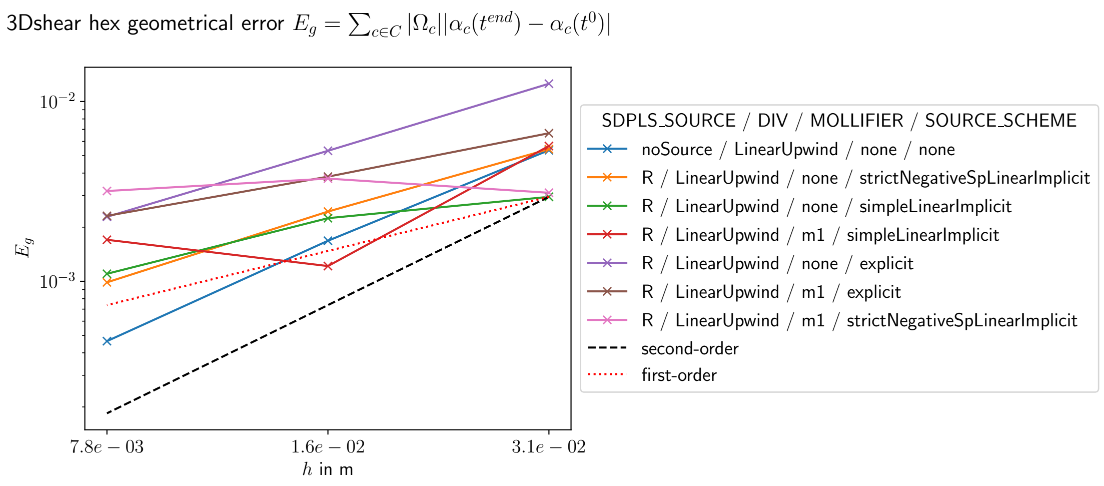


Figure 6.21.: Geometrical errors of different discretizations of the source term of the SDPLS method with *LinearUpwind* for the 3D shear test case on hexahedral meshes

while the signed distance error is still 1 and thus far from the signed distance. It also shows a slight divergence. On the other hand, this discretization shows a geometrical error of 4×10^{-3} and a convergence order of only 0.6, while the plain LS method shows an error of 1×10^{-6} and a second order convergence. For the 3D shear test case, the discretizations *strictNegativeSpLinearImplicit* and *simpleLinearImplicit* with mollifier show the lowest signed distance errors, but not far ahead of the plain LS method. None of them show convergence.

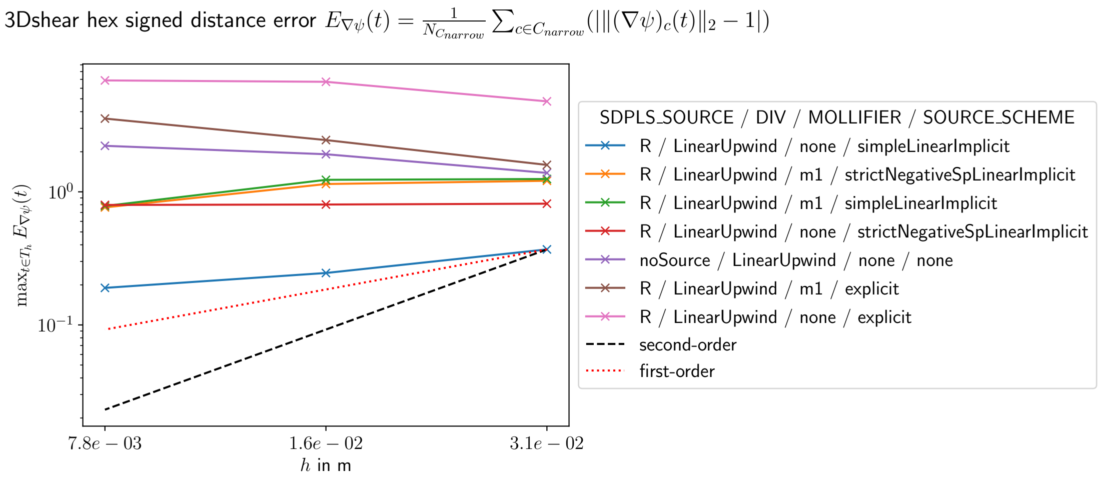


Figure 6.22.: Signed distance errors of different discretizations of the source term of the SDPLS method with *LinearUpwind* for the 3D shear test case on hexahedral meshes

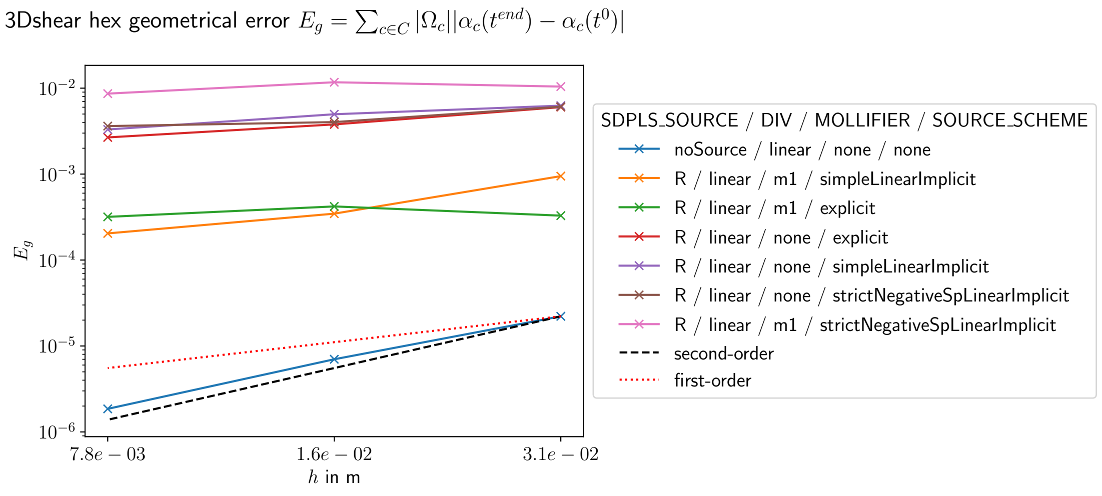


Figure 6.23.: Geometrical errors of different discretizations of the source term of the SDPLS method with *linear* for the 3D shear test case on hexahedral meshes

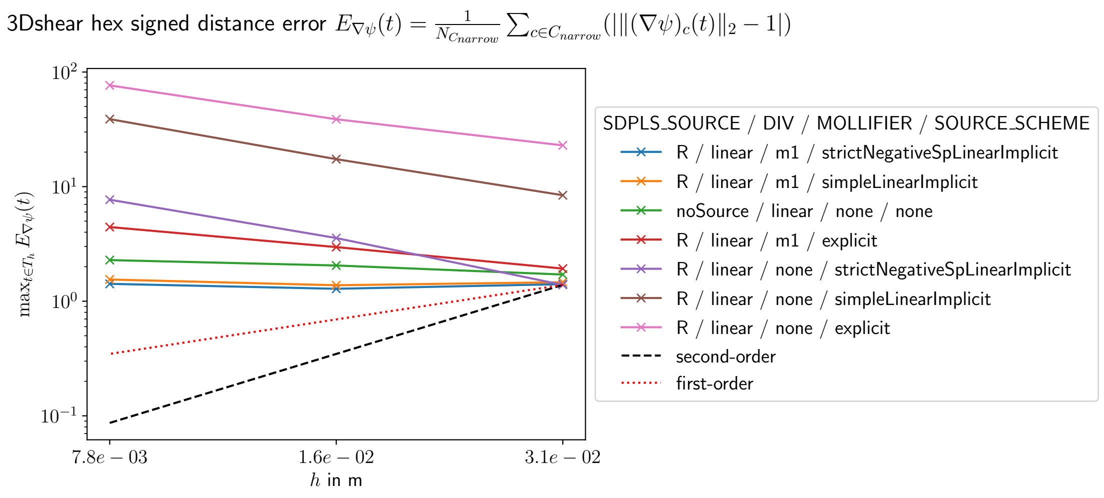


Figure 6.24.: Signed distance errors of different discretizations of the source term of the SDPLS method with *linear* for the 3D shear test case on hexahedral meshes

6.3.3. Different deformation intensities with the SDPLS method

Fricke et al. [17] presented first order signed distance convergence for the 2D contact line test case. In section 6.3.1 these results are verified with the implementation in this work. For the 3D deformation and 3D shear test cases, where a droplet undergoes much more deformations, the SDPLS method with the *LinearUpwind* advection scheme shows significant performance but convergence orders of 0.2 for 3D deformation and 0.5 for 3D shear. This section presents studies investigating the performance of the SDPLS method for different deformation intensities to see how long the SDPLS method can maintain the signed distance with first order convergence. The discretization *simpleLinearImplicit* without mollifier is chosen because it performed best with the *LinearUpwind* advection scheme.

3D deformation

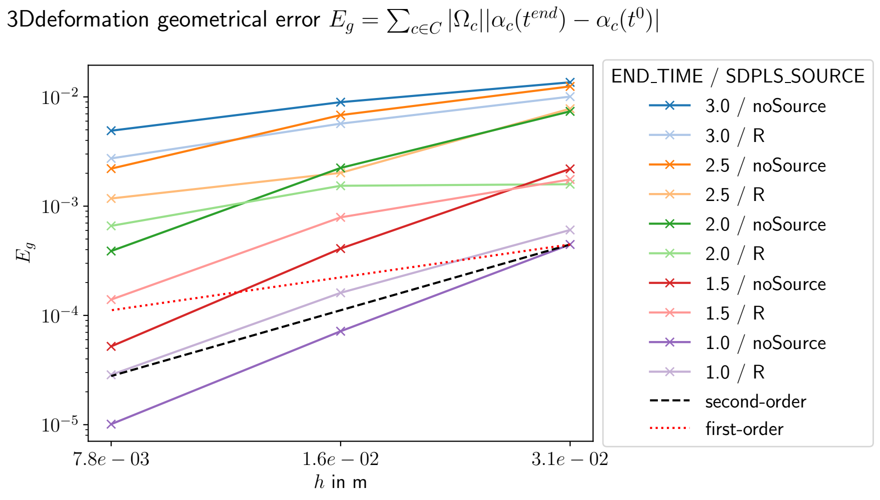


Figure 6.25.: Geometrical errors for different end times, corresponding to different levels of deformation, compared between the plain LS and the SDPLS method with *simpleLinearImplicit* and no mollifier discretization and the *LinearUpwind* advection scheme for the 3D deformation test case on hexahedral meshes

The deformation intensity is scaled by varying the period duration τ in the velocity

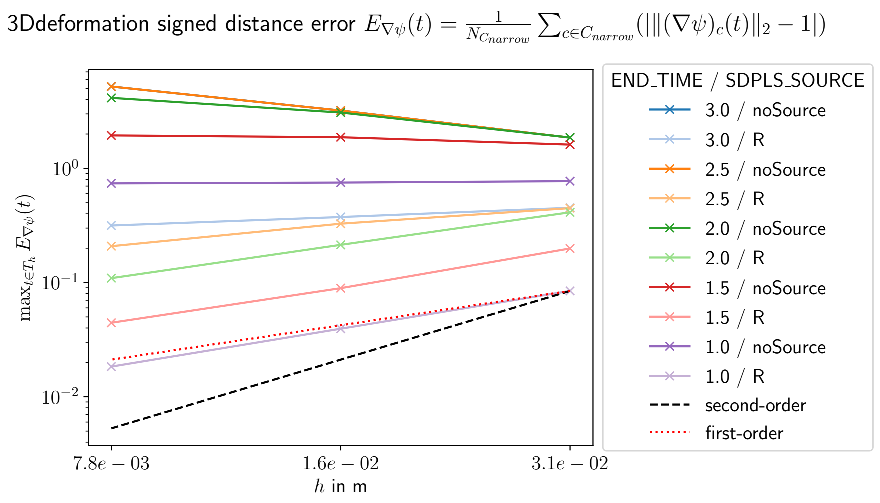


Figure 6.26.: Signed distance errors for different end times, corresponding to different deformation intensities, compared between the plain LS and the SDPLS method with *simpleLinearImplicit* and no mollifier discretization and the *LinearUpwind* advection scheme for the 3D deformation test case on hexahedral meshes

field, equation (6.5). Since only one deformation cycle is desired, $t^{end} = \tau$ is set. The figure 6.25 shows a comparison of the SDPLS method with the plain LS advection for the geometrical error E_g. There are five different levels of deformation denoted by the END_TIME value. The SDPLS and plain LS instances belonging to a deformation group share the line colour, with the noSource case having the stronger colour and the SDPLS instance having the paler colour. Looking at one group (colour) at a time, it can be seen that for the case of END_TIME $= 1.0$ (purple lines) the plain LS method performs better than the SDPLS method. For high deformations, e.g. END_TIME $= 3.0$ (blue lines), the SDPLS method performs better than the plain LS method. Overall, for large deformations (END_TIME ≥ 2.5) the SDPLS method shows better geometrical errors, while for small deformations (END_TIME ≤ 1.5) the plain LS method shows better geometrical errors.

Figure 6.26 shows the same comparison for the signed distance error $E_{\nabla\psi}$. All cases of the plain LS method show a signed distance error of magnitude 10^0, indicating that the signed distance is not preserved. Furthermore, none of these cases converge. For the cases with an active SDPLS source term, however, convergence can be seen. For small deformations with END_TIME ≤ 2.0 a first order convergence is visible. For higher deformations the order of the convergence flattens out.

3D shear

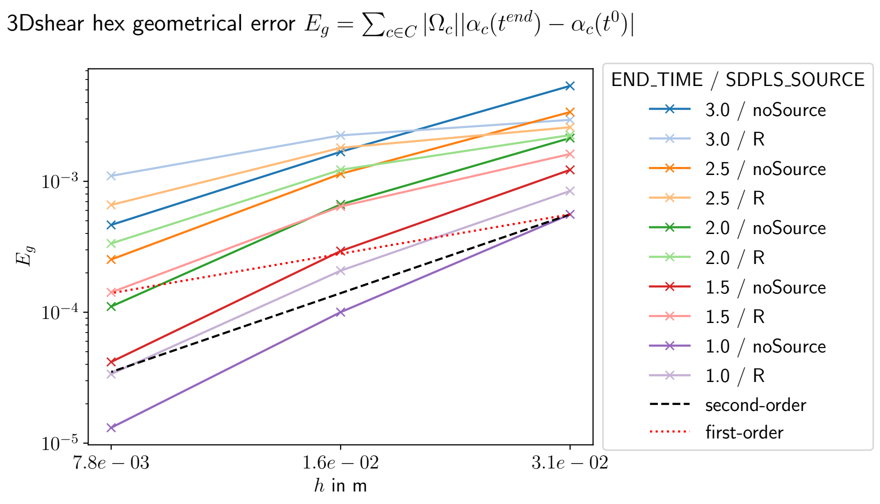


Figure 6.27.: Geometrical errors for different end times, corresponding to different levels of deformation, compared between the plain LS and the SDPLS method with *simpleLinearImplicit* and no mollifier discretization and the *LinearUpwind* advection scheme for the 3D shear test case on hexahedral meshes

Comparing the geometrical error E_g with and without the SDPLS source term during advection with the *LinearUpwind* in the 3D shear case, it becomes clear that `noSource` performs better at finer resolutions for all deformation intensities. The higher the deformation intensity, the greater the discrepancy between the plain LS method and the SDPLS method. The SDPLS method shows better geometrical errors only for high deformations on coarse meshes.

Looking at the signed distance error $E_{\nabla\psi}$ in figure 6.28 the SDPLS method performs better and preserves the signed distance at all deformation intensities. First order convergence is maintained up to a deformation intensity of `END_TIME` ≤ 2.0. For the deformation of `END_TIME` $= 1.0$ even a higher order of convergence of 1.5 is visible.

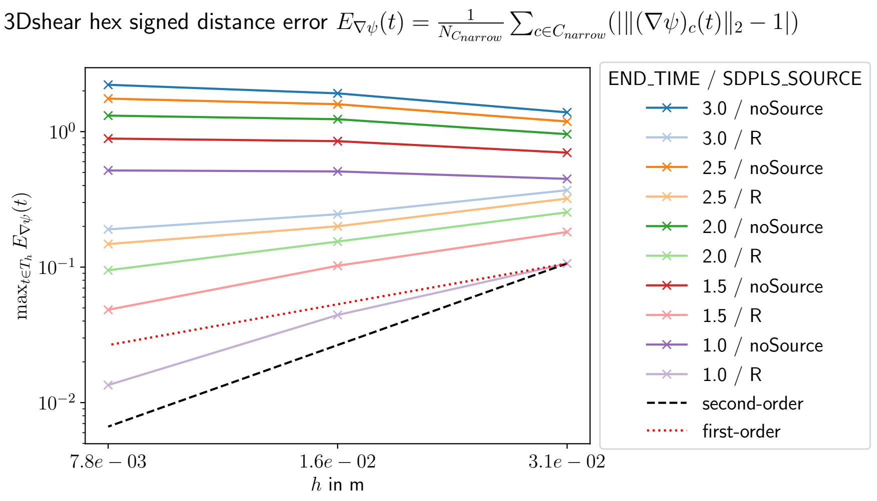
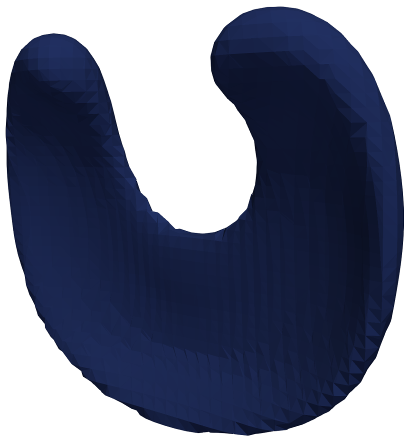


Figure 6.28.: Signed distance errors for different end times, corresponding to different levels of deformation, compared between the plain LS and the SDPLS method with *simpleLinearImplicit* and no mollifier discretization and the *LinearUpwind* advection scheme for the 3D shear test case on hexahedral meshes

In summary, the SDPLS method preserves the signed distance with first order accuracy for the test cases 3D deformation and 3D shear up to a deformation intensity corresponding to $\tau = 2.0$ for one cycle. The deformation amplitude at time $t = 1.0$ is visualised for both cases in figure 6.29. The geometrical error for the 3D deformation case is improved with the SDPLS method for high deformation, while for the 3D shear case it is improved only for high deformation on coarse meshes.


(a) 3D deformation at time $t = 1.0$

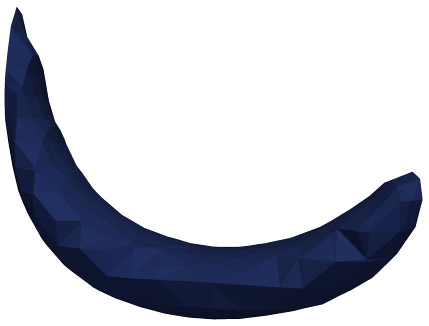

(b) 3D shear at time $t = 1.0$

Figure 6.29.: Deformation amplitude of the 3D deformation and 3D shear test cases with $\tau = 2.0$

6.4. Two-phase flows

This section presents the results of the simulation of the 3D stationary droplet two-phase flow test case using the developed solver, which is described in chapter 5.2. The following studies differ in the implementations used to calculate the surface tension force with the CSF model by Brackbill et al. [4] (see chapter 5.3), in the advection scheme used for the LS equation (4.30) and the momentum equation (4.18) and in the fluid combinations used. Plots are shown for parasitic currents as they are critical for accuracy and for volume conservation as it is a known weakness of the LS method. The following studies test the

Table 6.1.: Fluid Properties

Fluid	ν in $\mathrm{m^2/s}$	ρ in $\mathrm{kg/m^3}$
Water	1.0×10^{-6}	998.2
Air	1.53×10^{-5}	1.19
Oil (Novec7500)	0.77×10^{-6}	1614.0

fluid combination oil(Novec7500)—water with $\sigma = 49.5 \times 10^{-3}\,\mathrm{N/m}$ and water—air with $\sigma = 72.74 \times 10^{-3}\,\mathrm{N/m}$. The fluid properties can be read from the table 6.1. The advection schemes used are *linear*, *LinearUpwind*, *MUSCL* and *upwind*. Temporal discretization is performed using the implicit Euler method. For the gradient discretization the *Gauss linear* scheme is used.

oil(Novec7500)—water

This subsection presents the parasitic currents and volume conservation results for the 3D stationary droplet test case with the fluid pairing of oil(Novec7500)—water. This fluid pair has similar densities, which keeps the momentum jump at the interface small and the computation more stable.

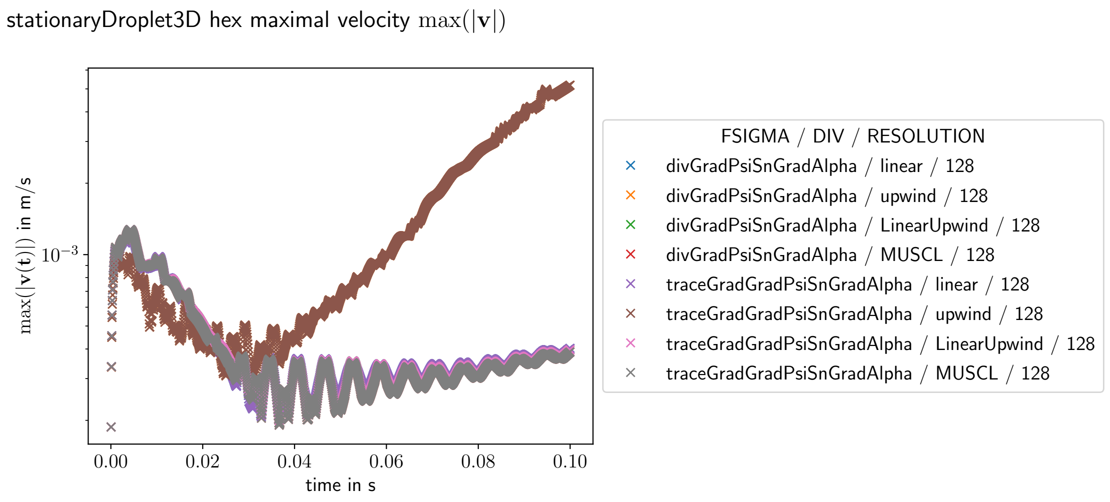


Figure 6.30.: Parasitic currents over time at the finest resolutions for the 3D stationary droplet test case with oil(Novec7500)—water fluid pairing on hexahedral meshes

Figure 6.30 shows the maximum velocity of the flow field over time at the finest resolution of 128 cells per dimension, resulting in a characteristic grid spacing of $h = 7.8 \times 10^{-5}$. Since the analytical solution does not include any motion, all resulting velocities correspond to parasitic currents. For the *linear*, *LinearUpwind* and *MUSCL* schemes, the parasitic currents initially decay. Then they oscillate with decreasing amplitude and converge to a value of 3.9×10^{-4}. Parasitic currents with the *upwind* advection scheme diverge. Also the surface tension implementations *divGradPsiSnGradAlpha* and *traceGradGradPsiSnGradAlpha* don't show any difference, which may be difficult to see but is verified with tabulated data.

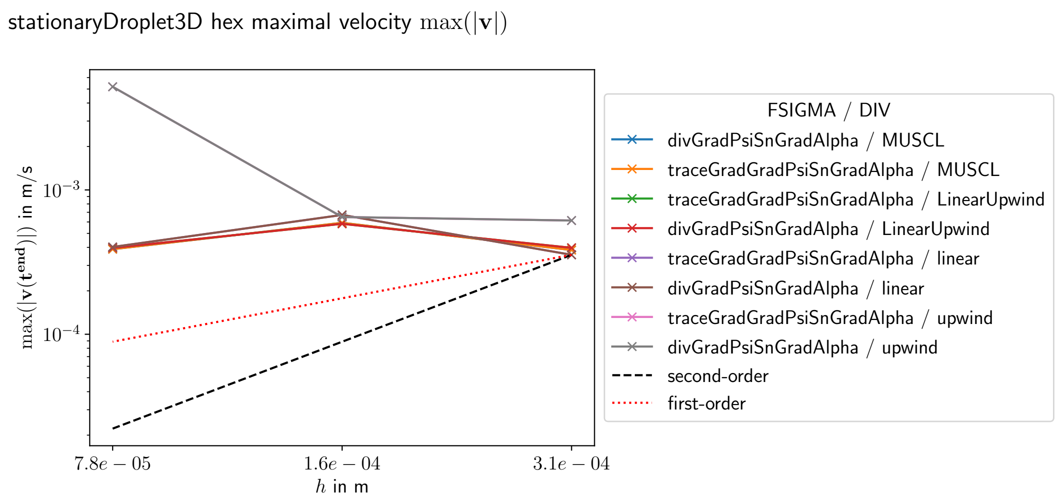


Figure 6.31.: Parasitic currents at different resolutions for the 3D stationary droplet test case with oil(Novec7500)—water fluid pairing on hexahedral meshes

Figure 6.31 shows the parasitic currents for all three mesh resolutions. No convergence is visible. But the parasitic currents for the schemes *linear*, *LinearUpwind* and *MUSCL* stay almost constant over all resolutions.

Figure 6.32 shows the volume conservation errors over the three mesh resolutions. For the schemes *linear*, *LinearUpwind* and *MUSCL* a second order convergence with an absolute error of 3.2×10^{-4} for *linear* and 4.6×10^{-4} for the other schemes is visible. The case with the *upwind* scheme at the finest mesh resolution if worse by one magnitude and *upwind* does not show a clear convergence.

Comparing the maximum parasitic current results with those of Lippert et al. [34, figure 9.c], the developed LS method performs comparable to the VoF methods implemented in *interFoam*, *interIsoFoam* and *Fluent* in the case of a 3D stationary droplet with oil(Novec7500)—water fluid pairing.

water—air

This subsection presents the parasitic currents and volume conservation results for the 3D stationary droplet test case with the fluid pairing of oil(Novec7500)—water. This

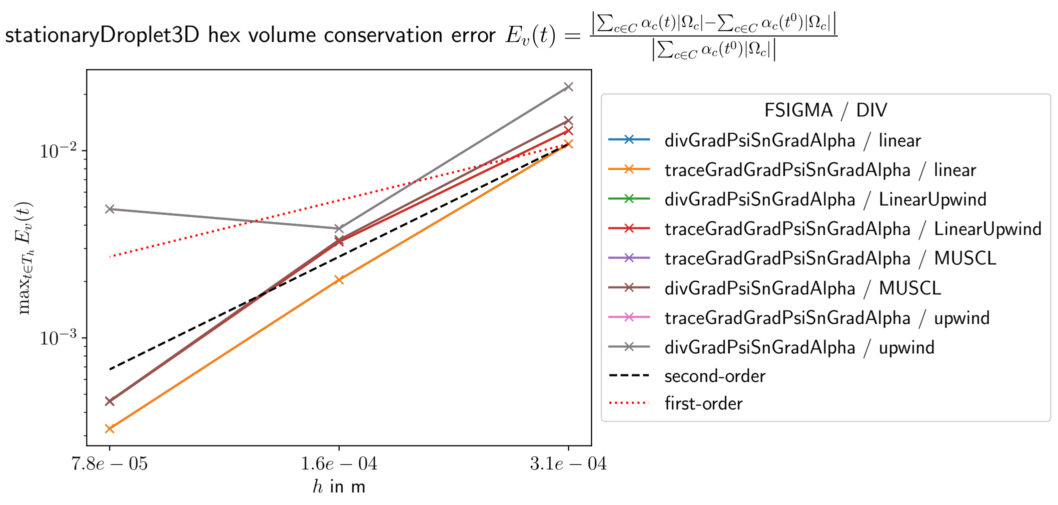


Figure 6.32.: Volume conservation errors at different resolutions for the 3D stationary droplet test case with oil(Novec7500)—water fluid pairing on hexahedral meshes

fluid pair has a high density ratio of $\left|\frac{\rho^{\text{water}}}{\rho^{\text{air}}}\right| = 838.8$, which introduces a high momentum jump across the interface and makes the calculation prone to instabilities. Avoiding these instabilities is a separate area of research and the method developed in this work does not include any measures to address this.

Figure 6.33 shows the parasitic currents plotted over time at the highest resolution of $h = 7.8 \times 10^{-5}$. Cases with the *linear* and *MUSCL* advection schemes crashed and are therefore not shown. The remaining cases show stable parasitic flows. At the beginning the parasitic currents increase to the magnitude of $100\,\mathrm{m/s}$. At the end they all seem to converge to $0.3\,\mathrm{m/s}$ with *LinearUpwind* showing high oscillations up to $100\,\mathrm{m/s}$.

Figure 6.34 shows the parasitic currents over all mesh resolutions. It shows that the parasitic currents increase with the mesh resolution. There is no significant difference between the different surface tension calculation methods *divGradPsiSnGradAlpha* and *traceGradGradPsiSnGradAlpha*.

Figure 6.35 shows the volume conservation errors over the three mesh resolutions. All combinations lose $100\,\%$ of their volume from the second resolution, which verify a complete failure of the method for this setup.

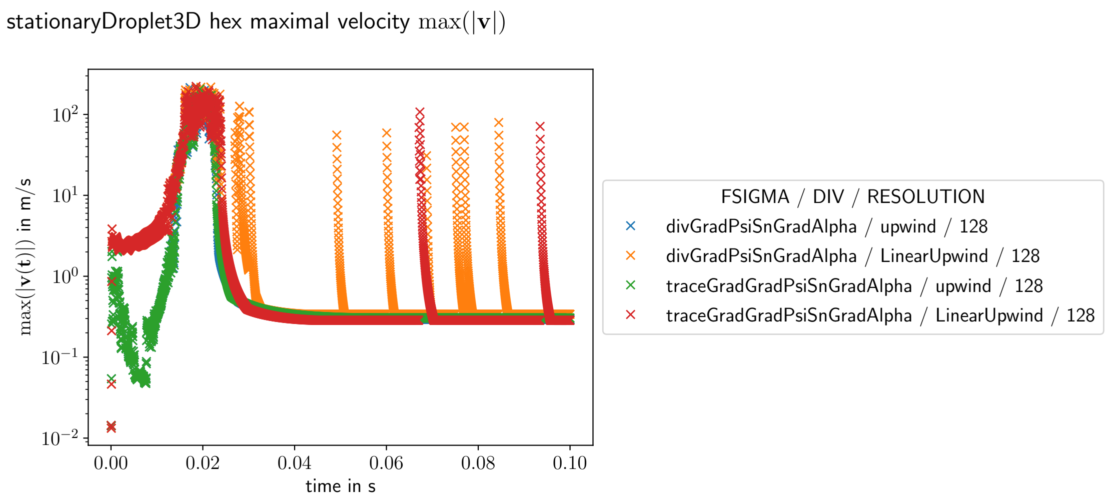


Figure 6.33.: Parasitic currents over time at the finest resolutions for the 3D stationary droplet test case with water—air fluid pairing on hexahedral meshes

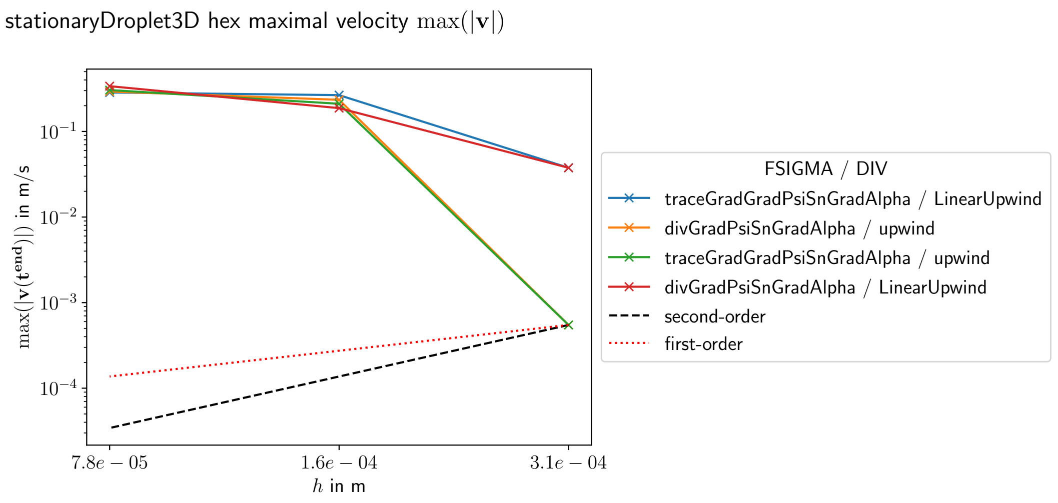


Figure 6.34.: Parasitic currents at different resolutions for the 3D stationary droplet test case with water—air fluid pairing on hexahedral meshes

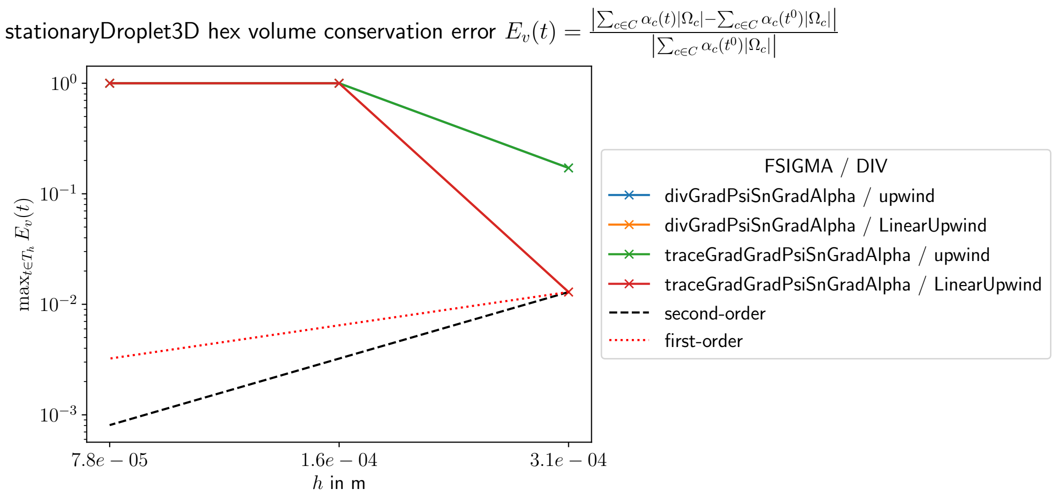


Figure 6.35.: Volume conservation errors at different resolutions for the 3D stationary droplet test case with water—air fluid pairing on hexahedral meshes

7. Conclusion

This chapter serves as a summary and conclusion of the main results of this thesis. It begins by addressing the question of the appropriate advection scheme for the LS method. It then discusses the results with respect to the SDPLS method. Finally, it discusses the studies carried out for the 3D stationary droplet test case.

When it comes to advection schemes for the LS method, there is no universally superior scheme that fits all situations. Among the schemes tested, both the *LinearUpwind* and *MUSCL* schemes show robustness for complex velocity fields and all types of meshes, while also exhibiting the best geometrical accuracy compared to other more robust schemes. On the other hand, the *linear* scheme performs poorly in the case of 3D rotation and even crashes when used with polyhedral meshes. It is therefore not as universally applicable as the *LinearUpwind* scheme. However, the *linear* scheme shows significant accuracy in terms of geometrical and volume conservation errors in the 3D deformation and 3D shear test cases. This is unexpected and surprising as the flow fields in the 3D deformation and 3D shear test cases are complex and generally not aligned with the face normals. Furthermore, it is shown that this result is not artificially generated by error cancellation due to the underlying oscillating velocity field. These observations raise questions about the significant differences between the flow fields in the 3D rotation test case and the 3D deformation and 3D shear test cases. Another interesting finding is that the *linear* scheme, when applied to the 3D stationary droplet case with oil(Novec7500)—water, shows similar magnitudes for parasitic currents and volume conservation in both the momentum and LS equations. This suggests that the *linear* scheme may hold relevance within the context of LS methods. Further research involving more complex flows may provide additional insights.

Considering all the studies on the SDPLS method, it can be concluded that the SDPLS method effectively improves the signed distance in all cases, especially when combined with the more monotonous *LinearUpwind* advection scheme. Among the different discretization options, the *simpleLinearImplicit* method without mollifier gives the best results for advection with the *LinearUpwind* scheme, as supported by Patankar [53], who stated that it is the most accurate linearization because of its tangential fit. Interestingly, the diagonal dominance of the assembled linear system appears to be unaffected by the SDPLS source term, rendering unnecessary the second best discretization option, *strictNegativeSpLinearImplicit*, which switches to explicit discretization on a cell-wise basis. In contrast, advection with the *linear* scheme in LS methods leads to higher signed distance errors overall. The SDPLS method with the *linear* advection scheme does not show significant improvements in signed distance errors compared to the plain LS method, suggesting a possible interference between the SDPLS source term and small oscillations introduced by the non-monotone *linear* advection scheme. The use of the SDPLS method results in improved signed distance measurements in all cases. However, first order convergence, as observed in the 2D contact line case, could not be established for the standard 3D deformation and 3D shear test cases, but for instances with reduced deformation amplitudes. In particular, for the 3D shear test case with a period duration of $\tau = 1$, a convergence order of 1.5 is even observed. In terms of geometrical error, improvements are observed for high deformations in the 3D deformation test case. Conversely, in the 3D shear case, improvements are only observed for high deformations and coarse resolutions. A potential further improvement to the SDPLS method could be to iteratively solve the LS equation and update the explicit non-linear part of the source term to converge to a fully implicit source term. This approach has the potential to improve the accuracy and stability of the method.

The developed LS method for the simulation of two-phase flows gives comparable results to VoF methods for the 3D stationary droplet test case with oil(Novec7500)—water fluid pairing. The parasitic currents remain below $10^{-3}\,\mathrm{m/s}$, and the volume loss is only about $0.033\,\%$ for the highest mesh resolution ($h = 7.8 \times 10^{-5}$). However, the method fails when applied to fluid pairs with high density ratios, such as water and air. To address this limitation, future work could focus on implementing and testing existing techniques designed to handle high density ratios. In addition, exploring the use of curvature corrections within the LS method could further reduce parasitic currents, taking advantage of the method's known ability to easily calculate curvature. Furthermore, subsequent studies could investigate two-phase flow test cases with more complex velocity fields. It would be interesting to evaluate the performance of the method in terms of volume conservation and curvature under such conditions. Also, using the SDPLS method to improve the signed distance and thereby improve the curvature calculations would be a worthwhile study.

A. Appendix

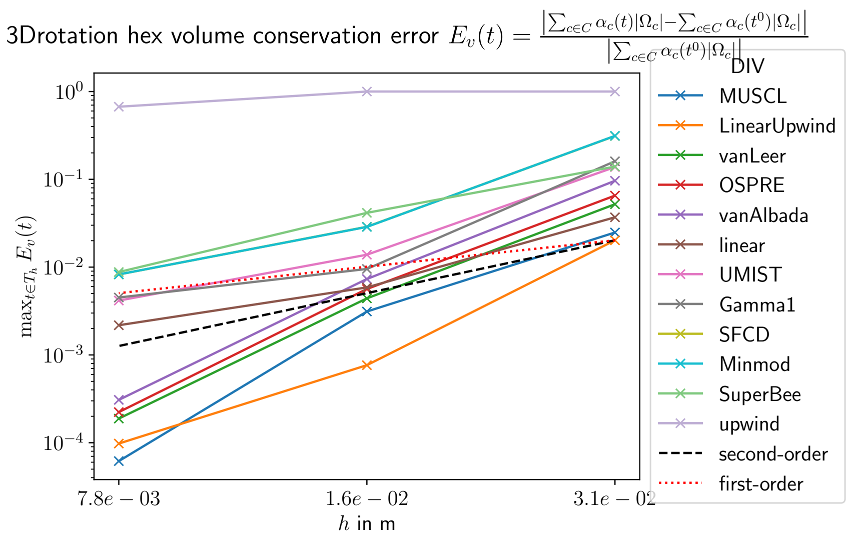


Figure A.1.: Volume conservation errors of the ten best advection schemes with the LS method for the 3D rotation test case on hexahedral meshes

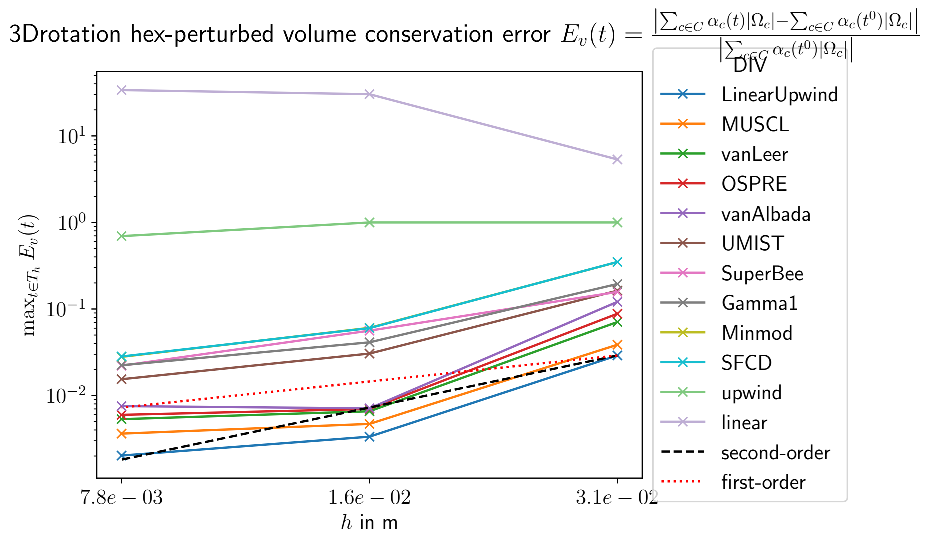


Figure A.2.: Volume conservation errors of the ten best advection schemes with the LS method for the 3D rotation test case on perturbed hexahedral meshes

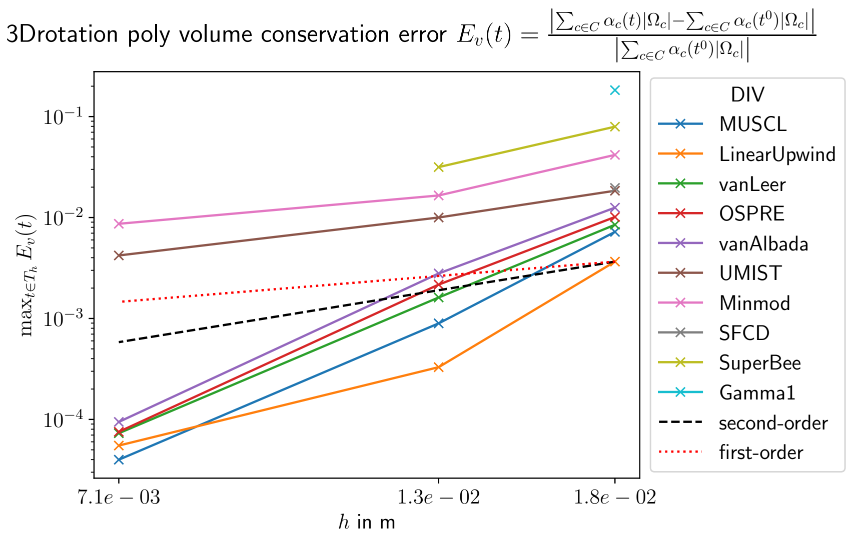


Figure A.3.: Volume conservation errors of the ten best advection schemes with the LS method for the 3D rotation test case on polyhedral meshes

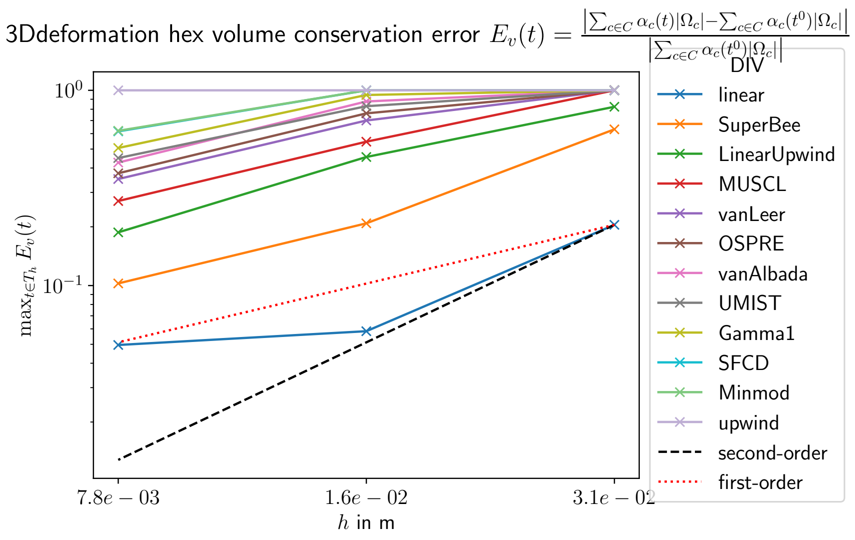


Figure A.4.: Volume conservation errors of the ten best advection schemes with the LS method for the 3D deformation test case on hexahedral meshes

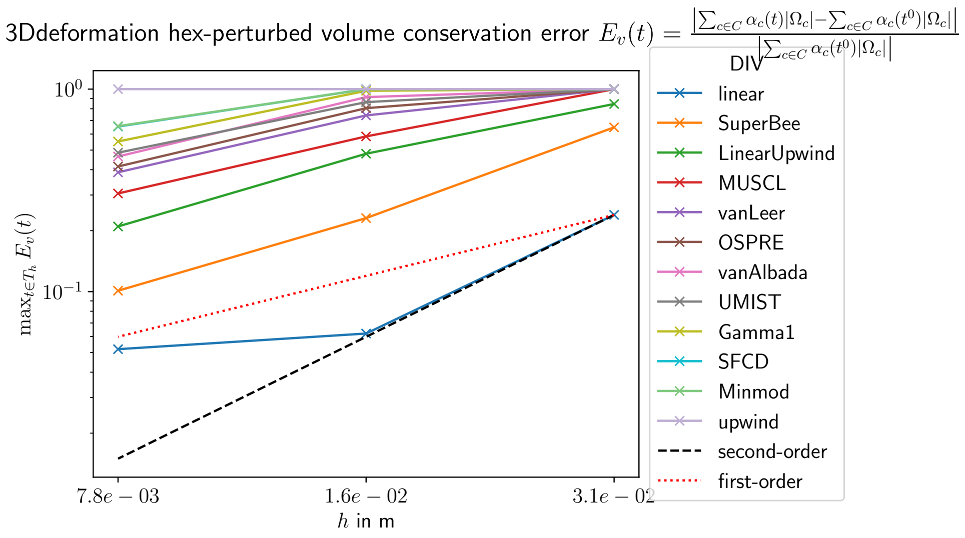


Figure A.5.: Volume conservation errors of the ten best advection schemes with the LS method for the 3D deformation test case on perturbed hexahedral meshes

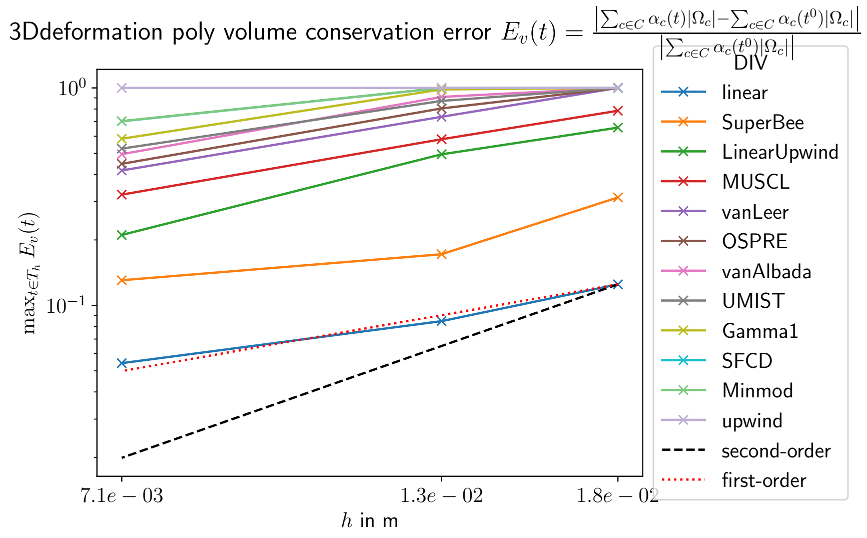


Figure A.6.: Volume conservation errors of the ten best advection schemes with the LS method for the 3D deformation test case on polyhedral meshes

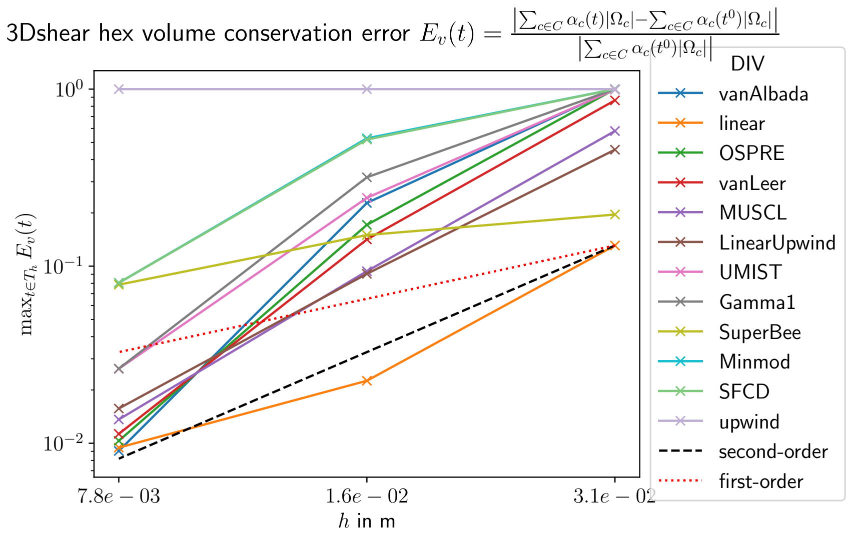


Figure A.7.: Volume conservation errors of the ten best advection schemes with the LS method for the 3D shear test case on hexahedral meshes

Bibliography

[1] D. Adalsteinsson and J. A. Sethian. “The Fast Construction of Extension Velocities in Level Set Methods”. In: *Journal of Computational Physics* 148.1 (Jan. 1, 1999), pp. 2–22. DOI: 10.1006/jcph.1998.6090. (Visited on 04/15/2023).

[2] R. F. Ausas, E. A. Dari, and G. C. Buscaglia. “A geometric mass-preserving redistancing scheme for the level set function”. In: *International Journal for Numerical Methods in Fluids* 65.8 (2011), pp. 989–1010. DOI: 10.1002/fld.2227. (Visited on 04/22/2023).

[3] N. Balcázar et al. “A finite-volume/level-set method for simulating two-phase flows on unstructured grids”. In: *International Journal of Multiphase Flow* 64 (Sept. 2014), pp. 55–72. DOI: 10.1016/j.ijmultiphaseflow.2014.04.008. (Visited on 10/19/2022).

[4] J. U. Brackbill, D. B. Kothe, and C. Zemach. “A continuum method for modeling surface tension”. In: *Journal of Computational Physics* 100.2 (June 1, 1992), pp. 335–354. DOI: 10.1016/0021-9991(92)90240-Y. (Visited on 04/03/2023).

[5] Y. C. Chang et al. “A Level Set Formulation of Eulerian Interface Capturing Methods for Incompressible Fluid Flows”. In: *Journal of Computational Physics* 124.2 (Mar. 15, 1996), pp. 449–464. DOI: 10.1006/jcph.1996.0072. (Visited on 04/03/2023).

[6] M. H. Cho, H. G. Choi, and J. Y. Yoo. “A direct reinitialization approach of level-set/splitting finite element method for simulating incompressible two-phase flows”. In: *International Journal for Numerical Methods in Fluids* 67.11 (2011), pp. 1637–1654. DOI: 10.1002/fld.2437. (Visited on 04/21/2023).

[7] D. L. Chopp. “Another Look at Velocity Extensions in the Level Set Method”. In: *SIAM Journal on Scientific Computing* 31.5 (Jan. 2009), pp. 3255–3273. DOI: 10.1137/070686329. (Visited on 04/20/2023).

[8] D. L. Chopp. “Computing Minimal Surfaces via Level Set Curvature Flow”. In: *Journal of Computational Physics* 106.1 (May 1, 1993), pp. 77–91. DOI: 10.1006/jcph.1993.1092. (Visited on 04/21/2023).

[9] D. L. Chopp. “Some Improvements of the Fast Marching Method”. In: *SIAM Journal on Scientific Computing* 23.1 (Jan. 2001), pp. 230–244. DOI: 10.1137/S106482750037617X. (Visited on 04/15/2023).

[10] M. Coquerelle and S. Glockner. “A fourth-order accurate curvature computation in a level set framework for two-phase flows subjected to surface tension forces”. In: *Journal of Computational Physics* 305 (Jan. 2016), pp. 838–876. DOI: 10.1016/j.jcp.2015.11.014. (Visited on 10/19/2022).

[11] F. Denner and B. G. M. van Wachem. “Fully-Coupled Balanced-Force VOF Framework for Arbitrary Meshes with Least-Squares Curvature Evaluation from Volume Fractions”. In: *Numerical Heat Transfer, Part B: Fundamentals* 65.3 (Mar. 4, 2014), pp. 218–255. DOI: 10.1080/10407790.2013.849996. (Visited on 05/10/2023).

[12] A. Dervieux and F. Thomasset. “A finite element method for the simulation of a Rayleigh-Taylor instability”. In: *Approximation Methods for Navier-Stokes Problems*. Ed. by R. Rautmann. Berlin, Heidelberg: Springer Berlin Heidelberg, 1980, pp. 145–158. DOI: doi.org/10.1007/BFb0086904.

[13] O. Desjardins, V. Moureau, and H. Pitsch. “An accurate conservative level set/ghost fluid method for simulating turbulent atomization”. In: *Journal of Computational Physics* 227.18 (Sept. 2008), pp. 8395–8416. DOI: 10.1016/j.jcp.2008.05.027. (Visited on 10/19/2022).

[14] D. Enright et al. “A Hybrid Particle Level Set Method for Improved Interface Capturing”. In: *Journal of Computational Physics* 183.1 (Nov. 20, 2002), pp. 83–116. DOI: 10.1006/jcph.2002.7166. (Visited on 04/14/2023).

[15] O. Fortmeier and H. Martin Bücker. “Parallel re-initialization of level set functions on distributed unstructured tetrahedral grids”. In: *Journal of Computational Physics* 230.12 (June 1, 2011), pp. 4437–4453. DOI: 10.1016/j.jcp.2011.02.005. (Visited on 04/21/2023).

[16] M. M. Francois et al. “A balanced-force algorithm for continuous and sharp interfacial surface tension models within a volume tracking framework”. In: *Journal of Computational Physics* 213.1 (Mar. 20, 2006), pp. 141–173. DOI: 10.1016/j.jcp.2005.08.004. (Visited on 04/11/2023).

[17] M. Fricke et al. *A locally signed-distance preserving level set method (SDPLS) for moving interfaces*. Aug. 2, 2022. DOI: 10.48550/arXiv.2208.01269. (Visited on 04/03/2023).

[18] F. Gibou, R. Fedkiw, and S. Osher. “A review of level-set methods and some recent applications”. In: *Journal of Computational Physics* 353 (Jan. 2018), pp. 82–109. DOI: 10.1016/j.jcp.2017.10.006. (Visited on 10/19/2022).

[19] J. Giesselmann. “Numerical Methods for Ordinary Differential Equations”. Lecture script. TU Darmstadt, Jan. 31, 2021.

[20] S. Groß. “Numerical methods for three-dimensional incompressible two-phase flow problems”. Medium: online, print. PhD thesis. Aachen: Publikationsserver der RWTH Aachen University, 2008. 216 pp. URL: http://publications.rwth-aachen.de/record/50223/files/Gross_Sven.pdf.

[21] S. Gross and A. Reusken. *Numerical Methods for Two-phase Incompressible Flows*. Series in Computational Mathematics. Springer Science & Business Media, Apr. 26, 2011. 487 pp. ISBN: 978-3-642-19686-7.

[22] A. Harten. “High resolution schemes for hyperbolic conservation laws”. In: *Journal of Computational Physics* 49.3 (Mar. 1, 1983), pp. 357–393. DOI: 10.1016/0021-9991(83)90136-5. (Visited on 04/15/2023).

[23] A. Harten et al. “Uniformly High Order Accurate Essentially Non-oscillatory Schemes, III”. In: *Journal of Computational Physics* 131.1 (Feb. 1, 1997), pp. 3–47. DOI: 10.1006/jcph.1996.5632. (Visited on 04/07/2023).

[24] S. Hysing. “A new implicit surface tension implementation for interfacial flows”. In: *International Journal for Numerical Methods in Fluids* 51.6 (2006). _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/fld.1147, pp. 659–672. DOI: 10.1002/fld.1147. (Visited on 05/05/2023).

[25] R. I. Issa. “Solution of the implicitly discretised fluid flow equations by operator-splitting”. In: *Journal of Computational Physics* 62.1 (Jan. 1, 1986), pp. 40–65. DOI: 10.1016/0021-9991(86)90099-9. (Visited on 05/02/2023).

[26] H. Jasak. “Error Analysis and Estimation for the Finite Volume Method with Applications to Fluid Flows”. PhD thesis. 1996.

[27] H. Jasak. “Finite Volume Discretisation in OpenFOAM”. 2015. URL: https://www.tfd.chalmers.se/~hani/kurser/OS_CFD_2015/HrvojeJasak/DiscretisationBestPractice.pdf.

[28] M. Kang, R. P. Fedkiw, and X.-D. Liu. “A Boundary Condition Capturing Method for Multiphase Incompressible Flow”. In: *Journal of Scientific Computing* 15.3 (Sept. 1, 2000), pp. 323–360. DOI: 10.1023/A:1011178417620. (Visited on 05/05/2023).

[29] R. Kimmel and J. A. Sethian. “Fast Marching Methods on Triangulated Domains”. In: *Proc. Nat. Acad. Sci.* 95 (1998), pp. 8431–8435. URL: https://cir.nii.ac.jp/crid/1572543025008810496 (visited on 04/16/2023).

[30] J. Kromer and D. Bothe. “Face-based Volume-of-Fluid interface positioning in arbitrary polyhedra”. In: *Journal of Computational Physics* 449 (Jan. 2022), p. 110776. DOI: 10.1016/j.jcp.2021.110776. (Visited on 01/09/2023).

[31] B. P. Leonard. “Simple high-accuracy resolution program for convective modelling of discontinuities”. In: *International Journal for Numerical Methods in Fluids* 8.10 (1988). _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/fld.1650081013, pp. 1291–1318. DOI: 10.1002/fld.1650081013. (Visited on 04/18/2023).

[32] R. J. LeVeque. “High-Resolution Conservative Algorithms for Advection in Incompressible Flow”. In: *SIAM Journal on Numerical Analysis* 33.2 (Apr. 1996), pp. 627–665. DOI: 10.1137/0733033. (Visited on 04/14/2023).

[33] P. Liovic et al. “A 3D unsplit-advection volume tracking algorithm with planarity-preserving interface reconstruction”. In: *Computers & Fluids* 35.10 (Dec. 1, 2006), pp. 1011–1032. DOI: 10.1016/j.compfluid.2005.09.003. (Visited on 04/14/2023).

[34] A. Lippert et al. *A benchmark for surface-tension-driven incompressible two-phase flows*. Dec. 6, 2022. DOI: 10.48550/arXiv.2212.02904. (Visited on 05/30/2023).

[35] X.-D. Liu, S. Osher, and T. Chan. “Weighted Essentially Non-oscillatory Schemes”. In: *Journal of Computational Physics* 115.1 (Nov. 1, 1994), pp. 200–212. DOI: 10.1006/jcph.1994.1187. (Visited on 04/18/2023).

[36] R. Malladi, J. Sethian, and B. Vemuri. “Shape modeling with front propagation: a level set approach”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 17.2 (Feb. 1995), pp. 158–175. DOI: 10.1109/34.368173.

[37] T. Maric. “Lagrangian / Eulerian Interface Approximation (LEIA) methods for two-phase flow simulation”. Lecture. TU Darmstadt, 2021. URL: https://gitlab.com/leia-lecture/leia-lecture-code.

[38] T. Maric. “Lagrangian/Eulerian numerical methods for fluid interface advection on unstructured meshes”. PhD thesis. Darmstadt: Technische Universität, Nov. 7, 2017. URL: https://tuprints.ulb.tu-darmstadt.de/7043/ (visited on 04/06/2023).

[39] T. Maric, J. Höpken, and K. Mooney. *The OpenFOAM Technology Primer*. Stan Mott, 2014. 442 pp. ISBN: 978-3-00-046757-8.

[40] T. Marić et al. *An second-order accurate geometrical phase indicator for the Level Set method on unstructured meshes*. in preparation.

[41] M. Mostafaiyan et al. “An Improved Conservative Direct Re-Initialization Method (ICDR) for Two-Phase Flow Simulations”. In: *Fluids* 6.7 (July 2021), p. 261. DOI: `10.3390/fluids6070261`. (Visited on 04/22/2023).

[42] F. Moukalled, L. Mangani, and M. Darwish. *The Finite Volume Method in Computational Fluid Dynamics: An Advanced Introduction with OpenFOAM® and Matlab*. Vol. 113. Fluid Mechanics and Its Applications. Cham: Springer International Publishing, 2016. DOI: `10.1007/978-3-319-16874-6`. (Visited on 11/24/2022).

[43] F. Mut, G. C. Buscaglia, and E. A. Dari. “New Mass-Conserving Algorithm for Level Set Redistancing on Unstructured Meshes”. In: *Journal of Applied Mechanics* 73.6 (Feb. 1, 2006), pp. 1011–1016. DOI: `10.1115/1.2198244`. (Visited on 04/21/2023).

[44] L. C. Ngo and H. G. Choi. “Efficient direct re-initialization approach of a level set method for unstructured meshes”. In: *Computers & Fluids*. ICCFD8 154 (Sept. 1, 2017), pp. 167–183. DOI: `10.1016/j.compfluid.2017.06.002`. (Visited on 04/17/2023).

[45] E. Olsson and G. Kreiss. “A conservative level set method for two phase flow”. In: *Journal of Computational Physics* 210.1 (Nov. 2005), pp. 225–246. DOI: `10.1016/j.jcp.2005.04.007`. (Visited on 10/19/2022).

[46] *OpenFOAM*. URL: `https://www.openfoam.com/` (visited on 04/13/2023).

[47] S. Osher and R. P. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Applied Mathematical Sciences v. 153. New York: Springer, 2003. 273 pp. URL: `https://link.springer.com/book/10.1007/b98879`.

[48] S. Osher and R. P. Fedkiw. “Level Set Methods: An Overview and Some Recent Results”. In: *Journal of Computational Physics* 169.2 (May 20, 2001), pp. 463–502. DOI: `10.1006/jcph.2000.6636`. (Visited on 04/01/2023).

[49] S. Osher and N. Paragios. *Geometric Level Set Methods in Imaging, Vision, and Graphics*. Springer Science & Business Media, July 17, 2003. 523 pp. ISBN: 978-0-387-95488-2.

[50] S. Osher and J. A. Sethian. “Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations”. In: *Journal of Computational Physics* 79.1 (Nov. 1, 1988), pp. 12–49. DOI: `10.1016/0021-9991(88)90002-2`. (Visited on 04/01/2023).

[51] M. L. Overton. “11. Cancellation”. In: *Numerical Computing with IEEE Floating Point Arithmetic*. Other Titles in Applied Mathematics. Society for Industrial and Applied Mathematics, Jan. 2001, pp. 71–75. DOI: `10.1137/1.9780898718072.ch11`. (Visited on 04/18/2023).

[52] S. Parameswaran and J. C. Mandal. “A stable interface-preserving reinitialization equation for conservative level set method”. In: *European Journal of Mechanics - B/Fluids* 98 (Mar. 1, 2023), pp. 40–63. DOI: `10.1016/j.euromechflu.2022.11.001`. (Visited on 04/17/2023).

[53] S. Patankar. *Numerical Heat Transfer and Fluid Flow*. Series in computational methods in mechanics and thermal sciences. CRC Press, 1980. 214 pp. ISBN: 978-0-89116-522-4.

[54] D. Peng et al. “A PDE-Based Fast Local Level Set Method”. In: *Journal of Computational Physics* 155.2 (Nov. 1, 1999), pp. 410–438. DOI: `10.1006/jcph.1999.6345`. (Visited on 04/20/2023).

[55] S. Popinet. “Numerical Models of Surface Tension”. In: *Annual Review of Fluid Mechanics* 50.1 (2018), pp. 49–75. DOI: `10.1146/annurev-fluid-122316-045034`. (Visited on 05/03/2023).

[56] T. Pringuey and R. S. Cant. “High Order Schemes on Three-Dimensional General Polyhedral Meshes — Application to the Level Set Method”. In: *Communications in Computational Physics* 12.1 (July 2012), pp. 1–41. DOI: `10.4208/cicp.260511.050811a`. (Visited on 10/19/2022).

[57] A. Prosperetti and G. Tryggvason. *Computational Methods for Multiphase Flow*. Cambridge University Press, June 25, 2009. 495 pp. ISBN: 978-1-139-45990-7.

[58] V. Ramanuj and R. Sankaran. “High Order Anchoring and Reinitialization of Level Set Function for Simulating Interface Motion”. In: *Journal of Scientific Computing* 81.3 (Dec. 1, 2019), pp. 1963–1986. DOI: `10.1007/s10915-019-01076-0`. (Visited on 04/17/2023).

[59] C. M. Rhie and W. L. Chow. “Numerical study of the turbulent flow past an airfoil with trailing edge separation”. In: *AIAA Journal* 21.11 (1983), pp. 1525–1532. DOI: `10.2514/3.8284`. (Visited on 05/10/2023).

[60] W. J. Rider and D. B. Kothe. “Reconstructing Volume Tracking”. In: *Journal of Computational Physics* 141.2 (Apr. 10, 1998), pp. 112–152. DOI: `10.1006/jcph.1998.5906`. (Visited on 04/14/2023).

[61] W. J. Rider and D. B. Kothe. “Stretching and tearing interface tracking methods”. In: *12th Computational Fluid Dynamics Conference*. 12th Computational Fluid Dynamics Conference. San Diego: American Institute of Aeronautics and Astronautics, June 19, 1995. DOI: `10.2514/6.1995-1717`. (Visited on 04/14/2023).

[62] E. Rouy and A. Tourin. “A Viscosity Solutions Approach to Shape-From-Shading”. In: *SIAM Journal on Numerical Analysis* 29.3 (June 1992). Publisher: Society for Industrial and Applied Mathematics, pp. 867–884. DOI: `10.1137/0729053`. (Visited on 04/17/2023).

[63] Y. Saad. *Iterative methods for sparse linear systems*. 2nd ed. Philadelphia: SIAM, 2003. 528 pp. URL: `http://catdir.loc.gov/catdir/enhancements/fy0665/2002044644-t.html` (visited on 04/05/2023).

[64] R. I. Saye and J. A. Sethian. “A review of level set methods to model interfaces moving under complex physics: Recent challenges and advances”. In: *Handbook of Numerical Analysis*. Vol. 21. Elsevier, 2020, pp. 509–554. DOI: `10.1016/bs.hna.2019.07.003`. (Visited on 10/24/2022).

[65] J. A. Sethian. “Fast Marching Methods”. In: *SIAM Review* 41.2 (Jan. 1999), pp. 199–235. DOI: `10.1137/S0036144598347059`. (Visited on 04/15/2023).

[66] J. A. Sethian and A. Vladimirsky. “Fast methods for the Eikonal and related Hamilton– Jacobi equations on unstructured meshes”. In: *Proceedings of the National Academy of Sciences* 97.11 (May 23, 2000), pp. 5699–5703. DOI: `10.1073/pnas.090060097`. (Visited on 04/16/2023).

[67] J. A. Sethian. “A fast marching level set method for monotonically advancing fronts.” In: *Proceedings of the National Academy of Sciences* 93.4 (Feb. 20, 1996), pp. 1591–1595. DOI: `10.1073/pnas.93.4.1591`. (Visited on 04/11/2023).

[68] J. A. Sethian. *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*. Cambridge University Press, June 13, 1999. 404 pp. ISBN: 978-0-521-64557-7.

[69] J. A. Sethian and P. Smereka. “Level Set Methods for Fluid Interfaces”. In: *Annual Review of Fluid Mechanics* 35.1 (2003), pp. 341–372. DOI: `10.1146/annurev.fluid.35.101101.161105`. (Visited on 04/06/2023).

[70] J. Strain. “Semi-Lagrangian Methods for Level Set Equations”. In: *Journal of Computational Physics* 151.2 (May 1999), pp. 498–533. DOI: `10.1006/jcph.1999.6194`. (Visited on 10/19/2022).

[71] M. Sussman and E. Fatemi. “An Efficient, Interface-Preserving Level Set Redistancing Algorithm and Its Application to Interfacial Incompressible Fluid Flow”. In: *SIAM Journal on Scientific Computing* 20.4 (Jan. 1999), pp. 1165–1191. DOI: `10.1137/S1064827596298245`. (Visited on 04/15/2023).

[72] M. Sussman, P. Smereka, and S. Osher. “A Level Set Approach for Computing Solutions to Incompressible Two-Phase Flow”. In: *Journal of Computational Physics* 114.1 (Sept. 1, 1994), pp. 146–159. DOI: `10.1006/jcph.1994.1155`. (Visited on 04/03/2023).

[73] P. K. Sweby. “High Resolution Schemes Using Flux Limiters for Hyperbolic Conservation Laws”. In: *SIAM Journal on Numerical Analysis* 21.5 (Oct. 1984), pp. 995–1011. DOI: `10.1137/0721062`. (Visited on 04/18/2023).

[74] *The OpenFOAM Foundation*. URL: `https://openfoam.org/` (visited on 04/13/2023).

[75] T. Tolle, D. Bothe, and T. Marić. “SAAMPLE: A Segregated Accuracy-driven Algorithm for Multiphase Pressure-Linked Equations”. In: *Computers & Fluids* 200 (Mar. 30, 2020), p. 104450. DOI: `10.1016/j.compfluid.2020.104450`. (Visited on 05/01/2023).

[76] M. F. Trujillo, L. Anumolu, and D. Ryddner. “The distortion of the level set gradient under advection”. In: *Journal of Computational Physics* 334 (Apr. 1, 2017), pp. 81–101. DOI: `10.1016/j.jcp.2016.11.050`. (Visited on 04/17/2023).

[77] G. Tryggvason et al. “A Front-Tracking Method for the Computations of Multiphase Flow”. In: *Journal of Computational Physics* 169.2 (May 20, 2001), pp. 708–759. DOI: `10.1006/jcph.2001.6726`. (Visited on 05/22/2023).

[78] G. Tryggvason et al. “Direct numerical simulations of gas/liquid multiphase flows”. In: *Fluid Dynamics Research*. Recent Topics in Computational Fluid Dynamics 38.9 (Sept. 1, 2006), pp. 660–681. DOI: `10.1016/j.fluiddyn.2005.08.006`. (Visited on 04/01/2023).

[79] G. Tryggvason, R. Scardovelli, and S. Zaleski. *Direct Numerical Simulations of Gas–Liquid Multiphase Flows*. Cambridge University Press, Mar. 10, 2011. 337 pp. ISBN: 978-1-139-49670-4.

[80] H. K. Versteeg and W. Malalasekera. *An Introduction to Computational Fluid Dynamics: The Finite Volume Method*. Pearson Education, 2007. 530 pp. ISBN: 978-0-13-127498-3.

[81] M. Y. Wang, X. Wang, and D. Guo. “A level set method for structural topology optimization”. In: *Computer Methods in Applied Mechanics and Engineering* 192.1 (Jan. 3, 2003), pp. 227–246. DOI: 10.1016/S0045-7825(02)00559-5. (Visited on 03/31/2023).

[82] H. Zhao. “A fast sweeping method for Eikonal equations”. In: *Mathematics of Computation* 74.250 (Apr. 1, 2005), pp. 603–627. DOI: 10.1090/S0025-5718-04-01678-3. (Visited on 04/17/2023).

[83] H. Zhao et al. “A Variational Level Set Approach to Multiphase Motion”. In: *Journal of Computational Physics* 127.1 (Aug. 1, 1996), pp. 179–195. DOI: 10.1006/jcph.1996.0167. (Visited on 04/11/2023).