Towards a unified multiphysics framework applied to reactive bubbly flows

Development of a unified multiphysics framework for OpenFOAM® and its application to reactive bubbly flows using ALE interface tracking

Master thesis by Constantin Habes

Date of submission: October 2, 2023

1. Review: Prof. Dr. Dieter Bothe
2. Review: Dr. Holger Marschall
Darmstadt

TECHNISCHE UNIVERSITÄT DARMSTADT

Mathematical Modeling and Analysis

Constantin Habes
Matrikelnummer: 2340988
Studiengang: Computational Engineering

Masterthesis: Towards a unified multiphysics framework applied to reactive bubbly flows

Eingereicht: 02. Oktober 2022

Technische Universität Darmstadt
Institut: Mathematical Modeling and Analysis
Prof. Dr. Dieter Bothe
Alarich-Weiss-Straße 10
64287 Darmstadt

Contents

Publications

[1] C. Habes, H. Alkafri, and H. Marschall. *On the generic structure of coupling conditions at fluid interfaces*. In preparation.

[2] C. Habes et al. *LCS4FOAM - An OpenFOAM Function Object to Compute Lagrangian Coherent Structures*. Submitted.

The above publications emerged from the research presented in this thesis and are either submitted or in preparation. Parts of this work are therefore closely related to these articles.

List of Figures

List of Tables

List of Symbols

Greek symbols

α	Relaxation parameter
Δ	Laplace operator
Γ	Generic diffusion coefficient
κ	Total surface curvature
λ	Eigenvalue
μ	Dynamic viscosity
ν	Kinematic viscosity
Ω	Angular velocity of moving reference frame
Ω	Bulk
$\hat{\Phi}$	Volume flux over cell face due to fluid motion
Φ	Extensive quantity
ϕ	Mass density-related volume specific density of extensive quantity
$\Phi_{t_0}^{t_1}$	Flow map
$\tilde{\Phi}$	Volume flux over cell face relative to mesh motion
Ψ	Scalar potential
ψ	Stream function
ρ	Mass density
Σ	Interface
σ	Surface tension
$\sigma_{t_0}^{t_1}$	Finite-time Lyapunov Exponent

σ_0	Surface tension of clean surface
$\boldsymbol{\tau}$	Deviatoric stress tensor
T	Time period
Θ	Scalar potential

Latin symbols

$\boldsymbol{A}$	Coefficient matrix
$\boldsymbol{a}$	Finite volume coefficient vector
a	Finite volume coefficient
$\boldsymbol{b}$	Finite volume source coefficient vector
b	Finite volume source coefficient
$\boldsymbol{C}$	Right Cauchy-Green tensor
c	Concentration
$\tilde{D}$	Species diffusion coefficient (concentration related)
D	Species diffusion coefficient (mass fraction related)
$\mathcal{F}$	Flux jump magnitude across interface
f	Primitive field
$\boldsymbol{g}$	Gravitation vector
h'	Interface face displacement
H	Henry constant
$\boldsymbol{I}$	Unity matrix
$\boldsymbol{j}$	Diffusive flux density of extensive quantity
$\mathcal{J}$	Jump magnitude of primitive field across interface
J	Surface flux of extensive quantity
k	Reaction rate coefficient
l_{PN}	Geodetic distance between two face centres on the interface
$\dot{m}$	Mass flux density

M	Molar mass
$\boldsymbol{n}$	Unit normal
N	Number of processors
n	Normalisation factor
$\hat{p}$	Kinematic pressure
$\tilde{p}$	Dynamic pressure
p	Pressure
q	Source/sink density of extensive quantity
q_e	Explicit component of source/sink density of extensive quantity
$\boldsymbol{r}$	Residual vector
$\hat{r}$	Normalised residual
$\tilde{r}$	Reaction rate (concentration related)
r	Reaction rate (mass fraction related)
R	Universal gas constant
$\boldsymbol{S}$	Face area normal vector
S	Source/Sink of extensive quantity
T	Absolute temperature
t	Time
$\boldsymbol{u}$	Velocity
$\boldsymbol{v}$	Velocity of moving reference frame
$\dot{V}$	Volume flux due to mesh motion
$\partial V'$	Correction swept volume
∂V	Swept volume
V	Volume
$\boldsymbol{w}$	Control volume boundary velocity
$\boldsymbol{y}$	Generic vector

Y	Mass fraction

Other symbols

$[\![\cdot]\!]$	Jump bracket notation
∇	Nabla operator
$\overline{(\cdot)}$	Interpolated quantity
$\partial(\cdot)$	Boundary of a set

Indices

$(\cdot)^0$	Initial value of quantity
$(\cdot)^{\pm}$	Quantity in the $+$ or $-$ bulk phase
$(\cdot)^{\Sigma}$	Quantity on the interface which also exists in the bulk
$(\cdot)^n$	Quantity at time step n
$(\cdot)^o$	Quantity at previous time step
$(\cdot)^{oo}$	Quantity at previous of previous time step
$(\cdot)_\phi$	Quantity related to a generic interface coupled field
$(\cdot)_{\partial\Sigma}$	Quantity on the interface boundary
$(\cdot)_\Sigma$	Quantity or operator only defined on the interface
$(\cdot)_c$	Quantity related to the interface coupled species concentration field
$(\cdot)_D$	Quantity related to an interface coupled Dirichlet boundary condition
$(\cdot)_f$	Quantity at cell face centre
$(\cdot)_f$	Quantity at interface face centre
$(\cdot)_G$	Quantity related to the gaseous phase
$(\cdot)_i$	Quantity related to species i (if not defined other in text)
$(\cdot)_L$	Quantity related to the liquid phase
$(\cdot)_{max}$	Maximal value of quantity
$(\cdot)_N$	Quantity related to an interface coupled Neumann boundary condition
$(\cdot)_P$	Quantity at cell centroid

$(\cdot)_p$	Quantity related to the interface coupled pressure field
$(\cdot)_s$	Quantity related to surfactant s
$(\cdot)_{\boldsymbol{u}}$	Quantity related to the interface coupled velocity field

1 Introduction

The production of basic chemical substances and their further industrial chemical and biochemical processing largely take place in liquids with dispersed gas and/or liquid or solid phases [6]. In these chemical engineering processes, one of the main tasks is to perform chemical reactions with high yield and selectivity to pave the way for a more sustainable and climate-friendly economy. Especially in gas-liquid reaction systems, it is crucial to get detailed knowledge of the local interplay of two-phase fluid dynamics, interfacial transport, mixing processes and chemical reactions in order to reduce the formation of undesired by-products and thus save resources, purification steps, and waste production [67].

The simulation of these bubbly flows brings severe challenges for the numerical methods, since not only the two-phase hydrodynamics but also the species transport including reactions have to be modelled correctly. Especially the hydrodynamics, the transport of species over, along, and in close proximity to the interface and their mixing and reaction in the bubble wake are of special interest. Because of these complexities, there exist many different methods with different advantages and disadvantages that can be used to simulate bubbly flows. However, the use of the Arbitrary Lagrangian-Eulerian Interface Tracking method (ALE-IT) has proven to be particularly suitable for the closer examination of single bubbles, taking into account mass transfer, reactions and surfactants [86, 61]. In contradiction to the most popular methods for two-phase systems, namely the Front Tracking method (FT), the Volume-Of-Fluid method (VOF) and the Level-Set method (LS), each phase in the ALE-IT is represented by its own computational mesh [86]. Hence each phase is specified by different sets of equations with accurate enforcement of coupling conditions at the interface. Moreover, the interface is represented by its own surface mesh which is especially beneficial for the modelling of surface tension dominated cases as well as for sorption processes and species transport along the interface [85].

Historically, Computational Fluid Dynamics (CFD) codes implementing ALE-IT for the simulation of bubbly flows were highly customised for this specific application. However, taking a broader look at multiphase flows in general, it is noticeable that they resemble other multiphysics systems in their basic structure. Analogous to Fluid-Structure Interaction (FSI) or Conjugate Heat Transfer (CHT) problems, multiphase systems also involve several regions with special physical properties that interact with each other across interfaces [41]. The approach of Spalding and his research group, which became known as the "Imperial College approach", has already shown that the aggregation of such structural similarities in a unified theory contributes to faster development and better extensibility of CFD codes [64]. Along this line, a unified theory for interface coupled multiphysics problems is presented in this thesis, with the main focus being on the generic formulation of interfacial jump and transmission conditions, which can be regarded as a complement to Spalding's generic transport equation [64]. Based on this theory, a new unified multiphysics framework for OpenFOAM[1] named multiRegionFOAM is introduced, which incorporates the ALE-IT and is tested for its application towards reactive bubbly flows. In this context, the parallelisability of the new framework will be addressed in particular, since interface coupled multiphysics simulations in general and simulations of reactive bubbly flows in particular are very computationally intensive.

[1]In this work, OpenFOAM refers to the OpenFOAM® fork foam-extend, with version 4.1 being used in particular.

As the name suggests, the algorithmic coupling across interfaces in the simulation of interface-coupled multiphysics systems is of major importance. The coupling procedures across interfaces can be divided into two classes, implicit/monolithic and explicit/partitioned interface coupling procedures. This work focuses on the Dirichlet–Neumann Algorithm (DNA) and its use in the multiRegionFOAM framework. The DNA is part of the explicit/partitioned interface coupling class, is widely used in FSI simulations and has also been studied by Weber et al. [86] for bubbly flows. Until now, however, no satisfactory criteria existed in the literature known to the author that quantified the quality of the interface coupling using DNA. Therefore, a DNA convergence control is developed in this thesis, which allows to steer the partitioned interface coupled solution process based on the residuals of the interface coupling conditions.

In the simulative study of reactive flows in general, the characterisation of the mixing processes is elementary. To facilitate the understanding as well as to quantitatively assess the mixing processes, Lagrangian Coherent Structures (LCS) of the flow can be utilised. These LCS order the material transport into dynamically distinct regions at large scales which resist diffusion or mixing and are therefore key to quantitatively assess the material transport processes in time-dependent flows. Since OpenFOAM does not provide the functionalities to compute Lagrangian Coherent Structures during the simulation out of the box, a function object is introduced and validated against established benchmark cases that enables the computation of Lagrangian Coherent Structures candidates on the fly based on Finite-Time Lyapunov Exponent (FTLE) fields.

In summary, the aim of this work is to introduce the theory and implementation of a unified, highly parallelisable multiphysics framework that implements a DNA convergence control based on interface residuals and is tested for its application towards reactive bubbly flows using the Arbitrary Lagrangian-Eulerian Interface Tracking method. Also a new OpenFOAM function object is introduced that enables the evaluation of mixing processes based on LCS in all kinds of flow fields.

2 Multiphysics Simulations

Aircraft or car manufacturing, printing processes, medical treatments, production of chemical substances, power generation, climate modelling, etc. The list of engineering research topics in which multiphysics simulations play an increasingly important role is constantly growing. Hence, complex systems with multiple interacting physical processes that had historically been considered separately are now studied in a coupled manner [41]. While this is not only due to the steadily increasing computational capacities and the continuous improvement of mathematical and numerical models, it is also due to the growing importance of simulations in scientific predictions, technical developments and political decision-making.

The broad range of applications leads to a frequent use of the general term ”multiphysics” in literature without defining it precisely. To establish a basis for the terminology and methods used in this thesis, the term ”multiphysics” and its different categories will be explained more detailed in this chapter. In particular, the structural similarity of different multiphysics problems is discussed, which in turn can be seen as a motivation for the development of the new unified multiphysics framework multiRegionFOAM (see section 5.1).

2.1 Multiphysics problems and their structural similarities

D. E. Keyes et al. formulated a definition of multiphysics from the perspective of computational mechanics in their 2011 report *Multiphysics simulation: challenges and opportunities* [41, p. 5]:

> ”Semantically, a multiphysics system consists of more than one component governed by its own principle(s) for evolution or equilibrium, typically conservation or constitutive laws. A major classification in such systems is whether the coupling occurs in the bulk (e.g., through source terms or constitutive relations that are active in the overlapping domains of the individual components) or whether it occurs over an idealised interface that is lower dimensional or a narrow buffer zone (e.g., through boundary conditions that transmit fluxes, pressures, or displacements).”

So in essence, there are two subcategories of multiphysics that can either occur individually or in combination. Following Michopoulos et al. [50] the first category is called ”multifield”. Here the multiphysics system is defined by the simultaneous interaction of multiple physical fields, whose coupling occurs in the same bulk. Therefore, we will call this type of coupling ”volumetric coupling”. Examples of these multifield systems appear among others in the fields of thermoelasticity [57], radiation hydrodynamics [51] or reactive flows [58]. Strictly speaking, an incompressible single phase flow already represents a multifield system, since it is described by two interacting fields. Namely the pressure and velocity field. The second multiphysics category is called ”multidomain” or ”multiregion”. In a multiregion system, multiple domains/regions with different properties are interacting with each other through interfaces. This results in the coupling of the physical fields across these boundaries, which we will therefore call ”interface coupling”. Since multiregion and multifield systems can appear in a combined manner, each region of a multiregion system can be a multifield system

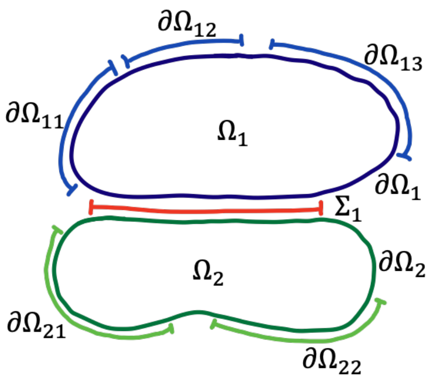


Figure 2.1: A two-region multifield system interacting over shared boundary Σ_1

itself. Examples of such multiregion systems, where one or more regions are a multifield system, appear in the fields of fluid structure interaction [10], conjugate heat transfer [90] or multiphase flows [42].

It is possible to formulate a general mathematical model that describes the interaction of multiple interacting multifield regions. This mathematical model is represented by the following system of coupled equations [50]:

$$\frac{d}{dt}\Phi_{di}(f_{dj}) = J_{di}(f_{dj}) + S_{di}(f_{dj}), \qquad \text{on } \Omega_d, \tag{2.1}$$

where the indices are defined by, $i \in [1, n_e]$, $j \in [1, n_f]$, $d \in [1, n_d]$ and n_e, n_f, n_d being the number of equations, fields and domains, respectively. Each of the n_e equations expresses a conservation law of an extensive quantity Φ with surface flux J and sources/sinks S in one of the n_d domains. The fields f_{dj} represent the dependent and independent variables of the conservation formulation and can therefore be tensors of various order. Figure 2.1 displays an example for a two-region multifield system.

Furthermore, a corresponding set of Dirichlet, Neumann, Robin or Cauchy boundary conditions needs to be imposed on every domain boundary $\partial\Omega_{dk} \subset \partial\Omega_d$, with index $k \in [1, n_b]$ and n_b being the number of boundaries of each domain. In the case of fields being present in more than one adjacent domains, special boundary conditions over the interfaces are required [50, p. 199]. Typically these are transmission conditions (2.2) preserving the conservation principles, and jump conditions (2.3) that mostly arise from constitutive laws:

$$[\![J_i]\!] \cdot \boldsymbol{n}_\Sigma = \mathcal{F}_i, \qquad \text{on } \Sigma_l \ , \tag{2.2}$$

$$[\![f_j]\!] = \mathcal{J}_j, \qquad \text{on } \Sigma_l \ . \tag{2.3}$$

Here, $[\![\cdot]\!]$ denotes the difference of a given value between each side of a shared boundary Σ_l, with index $l \in [1, n_s]$ and n_s being the number of interfaces in a multiregion system. A rigorous mathematical description of the so called jump bracket $[\![\cdot]\!]$ can be found in section 3.1. The transmission condition therefore states that the flux J_i of a conserved quantity in the normal direction $\boldsymbol{n}_\Sigma$ of an interface can have a jump of value $\mathcal{F}_i$. This means that the fluxes in the normal direction on each side of the interface my not be equal, which implies that the interface may store a conserved quantity to some extent. In contrast to the transmission conditions, which exist for each conserved quantity, jump conditions can be formulated for each of the primitive variables.

Here the jump condition states that the value of a primitive field f_j can have a discontinuity of value J_j across the interface. As already mentioned, such jump conditions usually arise from constitutive laws or can be derived from the previously considered transmission conditions.

This general mathematical model shows the structural similarity of multiphysics problems which can be exploited to develop a templated framework for the simulation of general multiphysics problems (see section 5.1).

2.2 Strong versus weak coupling and their algorithmic implications

As discussed in the previous section, the physical coupling in a multiphysics system can either occur in the bulk or across an interface. This physical volumetric or interface coupling of different fields or domains can either be strong or weak, which means that physical components of the multiphysics systems are either coupled by a strong or weak interaction [41, p. 9]. A strong volumetric and interface coupling, for example, occurs in the case of a rising gas bubble in a liquid. Here, the effect of the pressure field inside and outside of the bubble on the deformation of the bubble, the deformation of the bubble on the velocity field inside and outside of the bubble and the corresponding pressure and velocity fields on each other are instantaneous and bidirectional. In contrast, a weak physical volumetric coupling may appear in a reactive flow where the chemical species has an insignificant influence on the thermodynamic state of the fluid and may also be insensitive to the thermodynamic state itself [41, p. 10].

When simulating multiphysics systems, coupling algorithms are used that need to account for the nature of coupling in physics. This can either be done in an implicit/monolithic or explicit/partitioned way, where the implicit/monolithic approach couples the physical models more tightly while the explicit/partitioned approach is more loosely coupled. It is important to note that there is no direct correspondence between the strong (weak) coupling in physics and the monolithic (partitioned) coupling in numerics, since it is possible to use a monolithic or partitioned numerical coupling algorithm for strongly or weakly coupled physical models [41, p. 9]. This also holds true when both volumetric and interface coupling occurs in a multiregion system. Therefore, the numerical model for one multifield system might be coupled in a monolithic way while the interaction with another region across the interface might be coupled with a partitioned approach or vice versa. Hence, the used numerical coupling algorithms do not necessarily represent how strong the underlying physical coupling is, because either numerical coupling algorithms may be more computational efficient while satisfying a given accuracy [41, p. 10]. Nevertheless, a stronger underlying physical coupling requires well designed numerical coupling algorithms.

In the course of this work, the volumetric and interface coupling methods used for the simulation of reacting two-phase flows will be discussed. These are a semi-implicit (segregated) and a fully-implicit (fully-coupled) algorithm for the volumetric pressure-velocity coupling (see section 4.3.1), an explicit approach to volumetric coupling of the different reacting species to each other and to the underlying flow (see section 4.3.2) and a partitioned interface coupling algorithm that numerically couples the conservation equations across the interface (see section 4.4). Since the latter resolves a strong physical coupling in a partitioned way, special attention is paid to it.

3 Mathematical Model

In this chapter, we first derive the conservation equations for fluids with internal phase interfaces and ideally dilute reactive species. During this process the sharp interface model for a generic conservation equation is derived at first and then the special transport equations for mass momentum and reactive species are formulated based on it. This includes the derivation of the specialised transmission conditions for mass, momentum and reactive species and the formulation of jump conditions for all primitive variables by introducing constitutive equations and assumptions. Since the transport of surface-active agents (surfactants) is to be considered additionally to the transport of ideally dilute reactive species, a special surfactant transport model is described in section 3.3.2. This model also takes the sorption processes of surfactants at the interface and their transport along it into account.

3.1 Generic sharp interface model

The following derivation is based on the derivations of Slattery et al. [73], Ishi & Hibiki [33], Cermelli et al.[12] and Casey [35] and assumes a sharp, mass-less interface of zero thickness.

We consider a material control volume $V(t)$ under the continuum hypothesis (see figure 3.1). In this material volume $V(t)$, there are two immiscible fluids with their bulk phases $\Omega^+(t)$ and $\Omega^-(t)$, such that

$$V(t) = \Omega^+(t) \cup \Omega^-(t) \ . \tag{3.1}$$

These two fluids are in contact with each other over a deformable sharp interface

$$\Sigma(t) = \partial\Omega^+(t) \cap \partial\Omega^-(t) \tag{3.2}$$

with its boundary curve $\partial\Sigma(t)$, and $\partial\Omega^\pm(t)$ being the boundary of the corresponding bulk phase. Hence, the material volume $V(t)$ is bounded by

$$\partial V(t) = \partial V^+(t) \cup \partial V^-(t) \cup \partial\Sigma(t) \tag{3.3}$$

where

$$\partial V^\pm(t) = \partial\Omega^\pm(t) \setminus \Sigma(t) \ . \tag{3.4}$$

$\boldsymbol{n}$ is the outer unit normal to the control volume V. The unit normal to the interface $\Sigma(t)$ is $\boldsymbol{n}_\Sigma$ and always points into $\Omega^+(t)$. $\boldsymbol{n}_{\partial\Sigma}$ is the outer unit normal to the boundary curve of the interface, such that $\boldsymbol{n}_{\partial\Sigma} \perp \boldsymbol{n}_\Sigma$.

Since both fluids in V are governed by the same conservation equations we can formulate a general conservation equation that is analogue to (2.1):

$$\frac{d\Phi}{dt} = J + S, \quad \text{in } V \ . \tag{3.5}$$

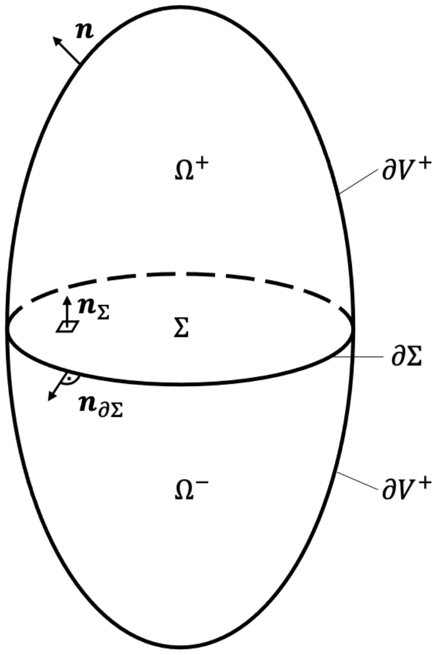


Figure 3.1: Material volume V of two immiscible fluid phases Ω^+ and Ω^- separated by a sharp interface Σ

The extensive quantity Φ (e.g. mass, momentum, energy) can then be expressed through a volume integral of its mass density-related volume-specific density ϕ which leads to the expression

$$\begin{aligned}\Phi &= \int_V \rho\phi dV \\ &= \int_{\Omega^+} \rho^+\phi^+ dV + \int_{\Omega^-} \rho^-\phi^- dV \ .\end{aligned} \tag{3.6}$$

Because of (3.1) this volume integral is split up into two integrals over Ω^+ and Ω^-. Therefore the superscripts $+$ and $-$ indicate the fluid phase to which the density fields belong. It is also important to note that all fields inside of V are assumed to be continuously differentiable, except for the interface being a singular surface. Similar to (3.6) the sources/sinks S of the extensive quantity can be expressed through a volume integral over its source/sink density field q:

$$\begin{aligned}S &= \int_V q dV \\ &= \int_{\Omega^+} q^+ dV + \int_{\Omega^-} q^- dV \ .\end{aligned} \tag{3.7}$$

An analogous expression can be formulated for the flux J of the extensive quantity over the boundary ∂V. Here J can be represented by the surface integral of the flux density field $\boldsymbol{j}$:

$$\begin{aligned}J &= -\int_{\partial V} \boldsymbol{j} \cdot \boldsymbol{n} dA \\ &= -\int_{\partial V^+} \boldsymbol{j}^+ \cdot \boldsymbol{n} dA - \int_{\partial V^-} \boldsymbol{j}^- \cdot \boldsymbol{n} dA - \int_{\partial\Sigma} \boldsymbol{j}^\Sigma \cdot \boldsymbol{n}_{\partial\Sigma} ds \ .\end{aligned} \tag{3.8}$$

Because of (3.3) this surface integral is split up into two surface integrals over ∂V^+ and ∂V^- and a line integral over $\partial\Sigma$ with $\boldsymbol{j}^\pm$ being the flux field of the corresponding fluid phase and $\boldsymbol{j}^\Sigma$ being the interface flux.

The left hand side of the general conservation equation (3.5) consists of the material derivative of Φ. If we consider the expression (3.6) for Φ the material derivative of the two integrals over Ω^+ and Ω^- is taken. In order to move the derivative within the integrals the expanded Reynolds transport theorem is used (see [72, p. 23-24] for a detailed derivation):

$$\begin{aligned}\frac{d}{dt}\int_{\Omega^\pm}\rho^\pm\phi^\pm dV &= \int_{\Omega^\pm}\frac{\partial\rho^\pm\phi^\pm}{\partial t}dV + \int_{\partial V^\pm}\left(\rho^\pm\boldsymbol{u}^\pm\phi^\pm\right)\cdot\boldsymbol{n}dA \mp \int_\Sigma\left(\rho^\pm\boldsymbol{u}^\Sigma\phi^\pm\right)\cdot\boldsymbol{n}_\Sigma dA \\ &= \int_{\Omega^\pm}\frac{\partial\rho^\pm\phi^\pm}{\partial t}dV + \int_{\partial V^\pm\cup\Sigma}\left(\rho^\pm\boldsymbol{u}^\pm\phi^\pm\right)\cdot\boldsymbol{n}dA \mp \int_\Sigma\left(\rho^\pm(\boldsymbol{u}^\Sigma-\boldsymbol{u}^\pm)\phi^\pm\right)\cdot\boldsymbol{n}_\Sigma dA\ .\end{aligned} \tag{3.9}$$

Here, the different signs in front of the last term appear due to the fact that $\boldsymbol{n}_\Sigma$ always points into Ω^+. $\boldsymbol{u}^\pm$ and $\boldsymbol{u}^\Sigma$ that appear in this expression are respectively the velocity field of the corresponding fluid phase and the velocity of the interface.

At this point, the terms in (3.9) represent the left hand side and the terms in (3.7) and (3.8) represent the right hand side of the general conservation equation (3.5). The goal is now to convert all these mentioned terms into integrals over $\Omega^\pm$ or Σ in order to distinguish the bulk terms from the interface terms.

Starting with (3.9), one can easily convert the integral over $\partial V^\pm\cup\Sigma = \partial\Omega^\pm$ into an integral over $\Omega^\pm$ by utilising the Gauß theorem for surface integrals [43]:

$$\int_{\partial V^\pm\cup\Sigma}\left(\rho^\pm\boldsymbol{u}^\pm\phi^\pm\right)\cdot\boldsymbol{n}dA = \int_{\Omega^\pm}\nabla\cdot\left(\rho^\pm\boldsymbol{u}^\pm\phi^\pm\right)dV\ . \tag{3.10}$$

In a similar way the integrals over ∂V^+ and ∂V^- in (3.8) can be converted with the Gauß theorem after a bit of algebraic manipulation:

$$\begin{aligned}-\int_{\partial V^\pm}\boldsymbol{j}^\pm\cdot\boldsymbol{n}dA &= -\int_{\partial V^\pm\cup\Sigma}\boldsymbol{j}^\pm\cdot\boldsymbol{n}dA \mp \int_\Sigma\boldsymbol{j}^\pm\cdot\boldsymbol{n}_\Sigma dA \\ &= -\int_{\Omega^\pm}\nabla\cdot\boldsymbol{j}^\pm dV \mp \int_\Sigma\boldsymbol{j}^\pm\cdot\boldsymbol{n}_\Sigma dA\ .\end{aligned} \tag{3.11}$$

The last integral to be transformed is the integral over $\partial\Sigma$ in (3.8). This can be converted into an integral over Σ by utilising the Stokes theorem for line integrals [43]:

$$\int_{\partial\Sigma}\boldsymbol{j}^\Sigma\cdot\boldsymbol{n}_{\partial\Sigma}ds = \int_\Sigma\left[\nabla_\Sigma\cdot\boldsymbol{j}^\Sigma - \left(\nabla_\Sigma\cdot\boldsymbol{n}_\Sigma\right)\boldsymbol{j}^\Sigma\cdot\boldsymbol{n}_\Sigma\right]dA\ . \tag{3.12}$$

In this expression the interface Nabla operator ∇_Σ is defined by

$$\nabla_\Sigma = \underbrace{\left[\boldsymbol{I}-\boldsymbol{n}_\Sigma\otimes\boldsymbol{n}_\Sigma\right]}_{\boldsymbol{I}_\Sigma}\cdot\nabla \tag{3.13}$$

with the interface divergence of a vector $\boldsymbol{y}$ being the trace of the interface gradient:

$$\nabla_\Sigma\cdot\boldsymbol{y} = tr(\nabla_\Sigma\boldsymbol{y})\ . \tag{3.14}$$

Using these definitions, the interface divergence of the surface normal gives a measure of the total surface curvature κ:

$$-\nabla_\Sigma \cdot \boldsymbol{n}_\Sigma = \kappa \, . \tag{3.15}$$

When substituting the converted integrals (3.10)-(3.12) into (3.9) and (3.8) and then substituting (3.6) - (3.9) into (3.5) one finally obtains the integral form of the generic balance equation:

$$\begin{aligned} &\int_{\Omega^+} \left[\frac{\partial \rho^+ \phi^+}{\partial t} + \nabla \cdot \left(\rho^+ \boldsymbol{u}^+ \phi^+ \right) + \nabla \cdot \boldsymbol{j}^+ - q^+ \right] dV \\ +&\int_{\Omega^-} \left[\frac{\partial \rho^- \phi^-}{\partial t} + \nabla \cdot \left(\rho^- \boldsymbol{u}^- \phi^- \right) + \nabla \cdot \boldsymbol{j}^- - q^+ \right] dV \\ +&\int_{\Sigma} \left\{ \nabla_\Sigma \cdot \boldsymbol{j}^\Sigma + \kappa \boldsymbol{j}^\Sigma \cdot \boldsymbol{n}_\Sigma - [\![\rho\phi (\boldsymbol{u}^\Sigma - \boldsymbol{u}) \cdot \boldsymbol{n}_\Sigma - \boldsymbol{j} \cdot \boldsymbol{n}_\Sigma]\!] \right\} dA = 0 \, . \end{aligned} \tag{3.16}$$

Here, the mathematical definition of the jump bracket $[\![\cdot]\!]$, which has already been introduced in section 2.1, is

$$[\![\phi]\!](t, \boldsymbol{x}) = \lim_{h \to 0^+} \left[\phi \left(t, \boldsymbol{x} + h \boldsymbol{n}_\Sigma \right) - \phi \left(t, \boldsymbol{x} - h \boldsymbol{n}_\Sigma \right) \right] , \quad \boldsymbol{x} \in \Sigma \, . \tag{3.17}$$

However, the integral form of the generic conservation equation still contains both bulk and interface terms. Therefore, these terms will be separated from each other in the next sections.

3.1.1 Bulk conservation equation

The integral form of the generic balance equation (3.16) can be localised by shrinking V to points inside of Ω^+, respectively in Ω^-. This yields two generic bulk conservation equations which have exactly the same structure and are either valid in Ω^+ or in Ω^-. Therefore, they can be written as one equation with the superscript $+$ or $-$ indicating the corresponding bulk phase:

$$\frac{\partial \rho^\pm \phi^\pm}{\partial t} + \nabla \cdot \left(\rho^\pm \boldsymbol{u}^\pm \phi^\pm \right) = -\nabla \cdot \boldsymbol{j}^\pm + q^\pm \quad \text{in } \Omega^\pm \, . \tag{3.18}$$

This equation matches the general transport equation introduced by Spalding [64]. The terms in this general/generic transport equation in the Eulerian reference frame describe, from left to right, the temporal change of an extensive quantity at a location in space, the convective flux of this quantity, the diffusive flux of this quantity and possible sources and sinks. Hence, specific transport equations can be formulated by specifying ϕ, $\boldsymbol{j}$ and q.

3.1.2 Interface transmission condition

Similar to the approach used previously to obtain the generic bulk conservation equation, (3.16) can again be localised by shrinking V to points on Σ. This yields the generic interface transmission condition

$$[\![\rho\phi(\boldsymbol{u}^\Sigma - \boldsymbol{u}) - \boldsymbol{j}]\!] \cdot \boldsymbol{n}_\Sigma = \nabla_\Sigma \cdot \boldsymbol{j}^\Sigma + \kappa \left(\boldsymbol{j}^\Sigma \cdot \boldsymbol{n}_\Sigma \right) \quad \text{on } \Sigma \, . \tag{3.19}$$

The form of this generic transmission condition shows a more precisely formulated and localised version of the already introduced transmission condition for general multi-physics problems (2.2). Thus, the left hand side of the equation corresponds to the jump of the convective and diffusive fluxes in the normal direction to the boundary surface. The right hand side defines the magnitude of this jump. In respect to the flux jump magnitude, it can be clearly seen that it purely depends on the flow of the conservation variable along the interface and the interface shape. Again, this generic transmission condition can be specialised with respect to a corresponding conservation equation by specifying ϕ, $\boldsymbol{j}$ and $\boldsymbol{j}^{\Sigma}$ accordingly.

3.1.3 Interface jump condition

Jump conditions for the individual primitive variables that occur in the conservation equations do not result directly from the derivations described above. Instead, they either arise inherently from the transmission conditions or result from the assumption of constitutive models. However, since they play an important role for the entire model and always have the same structure, a generic interface jump condition is formulated here for the sake of completeness. As already pointed out in section 2.1 the generic structure of an interface jump condition is as follows:

$$[\![f]\!] = \mathcal{J}, \quad \text{on } \Sigma \, . \tag{3.20}$$

Here, the jump of a primitive field f is described by a term denoted with $\mathcal{J}$. Depending on its origin, this term can either be zero, constant or an expression that is dependent on the considered primitive field or other fields. Specific formulations of this generic jump condition for pressure, velocity and species concentration can be found in the following two sections.

3.2 Mass and momentum transport

The mathematical model of flowing fluids is usually based on the conservation equations for mass, momentum and energy [66][75]. In this work, however, it is assumed that all the flow processes considered are isothermal, which is why the energy conservation equation is neglected. The specifications needed to obtain the remaining conservation equations for mass and momentum from the generic bulk conservation equation (3.18) are listed in table 3.1.

Table 3.1: Specifications of the generic bulk conservation equation and interface transmission condition for mass and momentum conservation

Equation	ϕ	$\boldsymbol{j}$	q	$\boldsymbol{j}^{\Sigma}$
Mass conservation	1	0	0	0
Momentum conservation	$\boldsymbol{u}$	$p\boldsymbol{I} - \boldsymbol{\tau}$	$\rho\boldsymbol{g}$	$\sigma\boldsymbol{I}$

If these specifications are inserted into equation (3.18), the following equations for mass conservation (3.21) and momentum conservation (3.22) are obtained:

$$\frac{\partial \rho^{\pm}}{\partial t} + \nabla \cdot \left(\rho^{\pm}\boldsymbol{u}^{\pm}\right) = 0 \quad \text{in } \Omega^{\pm} \, , \tag{3.21}$$

$$\frac{\partial \rho^{\pm}\boldsymbol{u}^{\pm}}{\partial t} + \nabla \cdot \left(\rho^{\pm}\boldsymbol{u}^{\pm} \otimes \boldsymbol{u}^{\pm}\right) = -\nabla \cdot p^{\pm} + \nabla \cdot \boldsymbol{\tau}^{\pm} + \rho^{\pm}\boldsymbol{g} \quad \text{in } \Omega^{\pm} \, . \tag{3.22}$$

Here, p represents the pressure, $\boldsymbol{\tau}$ represents the deviatoric stress tensor and $\boldsymbol{g}$ represents the gravitational vector. When furthermore constant densities and Newtonian flow behaviour specified through

$$\boldsymbol{\tau} = \mu \left(\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^\top \right) - \frac{2}{3}\mu(\nabla \cdot \boldsymbol{u})\boldsymbol{I} \tag{3.23}$$

with a constant dynamic viscosity μ in each phase is assumed, the Navier-Stokes equations for incompressible fluids arise:

$$\nabla \cdot \boldsymbol{u}^\pm = 0 \quad \text{in } \Omega^\pm \,, \tag{3.24}$$

$$\frac{\partial \boldsymbol{u}^\pm}{\partial t} + \left(\boldsymbol{u}^\pm \cdot \nabla\right) \boldsymbol{u}^\pm = -\frac{1}{\rho^\pm}\nabla \cdot p^\pm + \nu^\pm \Delta \boldsymbol{u}^\pm + \boldsymbol{g} \quad \text{in } \Omega^\pm \,. \tag{3.25}$$

In these equations, the mass conservation is simplified to the solenoidal condition (3.24), which in turn is used to simplify the expression of the deviatoric stress tensor for Newtonian fluids in the momentum conservation (3.25). Additionally, ν represents the kinematic viscosity in each phase, since the conservation of momentum was further simplified by dividing it by the constant density.

In order to obtain the transmission conditions for mass and momentum from the generic interface transmission conditions, the specifications of the interfacial fluxes are required, which can also be found in table 3.1. Here, the interfacial mass flux is set to zero since the interface was assumed to be mass-less. Only the interfacial momentum flux is specified by the surface tension σ. This implies that momentum can be transported along the interface and the interface itself can apply momentum on the bulk due to capillary forces. Inserting these specifications into (3.19) gives the transmission condition for mass (3.26) and for momentum (3.27):

$$[\![\rho \left(\boldsymbol{u}^\Sigma - \boldsymbol{u}\right)]\!] \cdot \boldsymbol{n}_\Sigma = 0 \quad \text{on } \Sigma \,, \tag{3.26}$$

$$[\![\rho \boldsymbol{u} \left(\boldsymbol{u}^\Sigma - \boldsymbol{u}\right) + p\boldsymbol{I} - \boldsymbol{\tau}]\!] \cdot \boldsymbol{n}_\Sigma = \nabla_\Sigma \sigma + \kappa \sigma \boldsymbol{n}_\Sigma \quad \text{on } \Sigma \,. \tag{3.27}$$

Here, the mass transmission condition states that the mass flux over the interface must be continuous with the mass flux per surface area being

$$\dot{m}_\Sigma = \rho^+ \left(\boldsymbol{u}^+ - \boldsymbol{u}^\Sigma\right) \cdot \boldsymbol{n}_\Sigma = \rho^- \left(\boldsymbol{u}^- - \boldsymbol{u}^\Sigma\right) \cdot \boldsymbol{n}_\Sigma \,. \tag{3.28}$$

If we now assume that the phase interface represents a material boundary across which no mass is transported ($\dot{m}_\Sigma = 0$), the following applies [35]:

$$\boldsymbol{u}^+ \cdot \boldsymbol{n}_\Sigma = \boldsymbol{u}^- \cdot \boldsymbol{n}_\Sigma = \boldsymbol{u}^\Sigma \cdot \boldsymbol{n}_\Sigma \quad \text{on } \Sigma \,. \tag{3.29}$$

When this relationship is exploited in the momentum transmission condition and we again assume Newtonian fluid behaviour (3.23) and the solenoidal condition (3.24), it simplifies to

$$[\![p\boldsymbol{I} - \mu \left(\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^\top\right)]\!] \cdot \boldsymbol{n}_\Sigma = \nabla_\Sigma \sigma + \kappa \sigma \boldsymbol{n}_\Sigma \quad \text{on } \Sigma \,. \tag{3.30}$$

Here the momentum transmission condition states that the jump of the hydrostatic and deviatoric stresses in normal direction to the interface is dependent on the local surface tension and the surface shape.

At this point, assuming incompressibility and Newtonian flow behaviour of the fluids, the transport equations for mass (3.24) and momentum (3.25) have been derived. Furthermore, under the additional assumption of a material interface, the associated transmission conditions for mass (3.29) and momentum (3.30) were formulated. Now only the jump conditions for the dependent variables pressure p and velocity $\boldsymbol{u}$ are missing. The jump condition for the pressure can be derived by taking the interface normal component from (3.30) and using the results from Chen et al. [13] to reformulate the jump of the deviatoric stress tensor. This yields the equation

$$[\![p]\!] = \sigma\kappa - 2[\![\mu]\!]\nabla_\Sigma \cdot \boldsymbol{u} \quad \text{on } \Sigma\,. \tag{3.31}$$

The jump condition for the velocity results from the mass transmission condition (3.29) if additionally the constitutive no-slip condition [15] between the two phases at the interface is assumed:

$$[\![\boldsymbol{u}]\!] = 0 \quad \text{on } \Sigma\,. \tag{3.32}$$

3.3 Species transport

3.3.1 Reactive species transport

Fluids can consist of an arbitrary number of different species that may also react with each other. To model the transport of these species, additional transport equations are needed (one for each specie of interest). The specifications used to obtain these transport equations from the generic bulk conservation equation under the assumption of a high dilution of the species are listed in table 3.2.

Table 3.2: Specifications of the generic bulk conservation equation and interface transmission condition for reactive species conservation

Equation	ϕ	$\boldsymbol{j}$	q	$\boldsymbol{j}^\Sigma$
Surfactant conservation	Y_i	$-D_i\nabla Y_i$	r_i	0

Here, Y_i denotes the mass fraction and r_i denotes the reaction rate of a species i. Fick's law of diffusion [23] is already assumed in table 3.2. Therefore, $\boldsymbol{j}$ is specified by $-D_i\nabla Y_i$ where D_i is the diffusion coefficient of the respective species, which is assumed to be constant. Since the species mass balance is usually formulated in terms of their concentration, the mass fraction Y_i of a species i can be converted into its concentration by the relation

$$Y_i = \frac{c_i \cdot M_i}{\rho} \tag{3.33}$$

where c_i is the concentration and M_i is the molar mass of the species under consideration. When inserting these specifications into equation (3.18) and using relation (3.33), the following transport equation is obtained:

$$\frac{\partial c_i^\pm}{\partial t} + \nabla \cdot \left(c_i^\pm \boldsymbol{u}^\pm\right) = \tilde{D}_i^\pm \Delta c_i^\pm + \tilde{r}_i^\pm \quad \text{in } \Omega^\pm\,. \tag{3.34}$$

Because of the conversion from mass fraction to concentration, $\tilde{r}_i$ represents the concentration dependent reaction rate instead of the mass fraction dependent reaction rate r_i. The same applies to $\tilde{D}_i$.

The interface transmission condition for a specie i results from again inserting the specifications listed in table 3.2 into the generic transmission condition (3.19):

$$[\![\tilde{D}_i \nabla c_i]\!] \cdot \boldsymbol{n}_\Sigma = 0 \quad \text{on } \Sigma\,. \tag{3.35}$$

Here, a relaxation of the material interface assumption used in the mass transmission condition (3.29) takes place, since the mass transfer of species by diffuse fluxes across the interface is taken into account. In fact, it states that the diffusive flux of a species i is continuous across the interface. Of course, the material interface assumption only applies to systems in which the mass loss or increase due to species transfer is negligible; which is assumed to be true in this work. However, this is no longer true for highly concentrated systems like a bubble of a pure gas dissolving in an ambient liquid [9].

With the transport equation (3.34) and the transmission condition (3.35) for reactive species given, only the jump condition for each specie still has to be acquired. Since the concentration of a species on the respective sides of the interface depends on its chemical potential, the jump condition for the chemical potentials must first be considered in order to derive the concentration jump condition. This has already been done by Bothe and Fleckenstein [9], who concluded that the concentration jump condition can be suitably approximated by Henry's law. Adopting their results gives the jump condition

$$[\![c_i]\!] = c_i^-(H_i - 1) \tag{3.36}$$

where H_i represents the Henry solubility constant.

3.3.2 Surfactant transport

At this point, the mathematical model for a two-phase flow with a sharp, mass-less interface and transport of ideally diluted, non surface active species is described. However, since the transport of surfactants also need to be considered in practical applications of bubbly flows, an extension of the species transport model presented previously is derived in this section.

In most two-phase flows, surfactants appear either as impurities or as additives and will therefore also be considered in this work. Whether they are intentionally or unintentionally introduced into the system, they significantly affect the behaviour of the two-phase system [1, 18]. This is because surfactants are adsorbed at the interface and, depending on their concentration, locally influence the surface tension, i.e. the capillary forces. In order to describe this sorption process and the subsequent transport of the adsorbed surfactants along the interface, the previously made assumption of a mass-less interface must be slightly relaxed. We therefore now assume in respect to the possibility of adsorbed surfactants that the interface holds a negligible mass. We also assume that no reactions of the surfactants take place on the interface. In the previously presented derivations of the generic model this corresponds to adding an additional interface related term to (3.6):

$$\begin{aligned}\Phi &= \int_V \rho\phi dV \\ &= \int_{\Omega^+} \rho^+\phi^+ dV + \int_{\Omega^-} \rho^-\phi^- dV + \int_\Sigma \rho^\Sigma \phi^\Sigma dA\end{aligned} \tag{3.37}$$

where ρ^Σ ($^{\text{kg}}/_{\text{m}^\text{s}}$) and ϕ^Σ are the surface area related equivalences to the volume related $\rho^\pm$ and $\phi^\pm$. A derivation similar to the one given in section 3.1 then yields a generic bulk equation that exactly matches (3.18) and an interface equation that reads

$$\frac{\partial \rho^\Sigma \phi^\Sigma}{\partial t} + \nabla_\Sigma \cdot \left(\rho^\Sigma \boldsymbol{u}^\Sigma \phi^\Sigma\right) + \nabla_\Sigma \cdot \boldsymbol{j}^\Sigma - \kappa \boldsymbol{j}^\Sigma \cdot \boldsymbol{n}_\Sigma = [\![\rho\phi(\boldsymbol{u}^\Sigma - \boldsymbol{u}) \cdot \boldsymbol{n}_\Sigma - \boldsymbol{j} \cdot \boldsymbol{n}_\Sigma]\!] \quad \text{on } \Sigma\,. \tag{3.38}$$

Since a surfactant s behaves like every other non surface-active specie i in the bulk, the transport equation for both are identical when the same assumptions as in 3.3.1 are applied (also see table 3.3):

$$\frac{\partial c_s^\pm}{\partial t} + \nabla \cdot \left(c_s^\pm \boldsymbol{u}^\pm\right) = \tilde{D}_s^\pm \Delta c_s^\pm + \tilde{r}_s^\pm \quad \text{in } \Omega^\pm\,. \tag{3.39}$$

Table 3.3: Specifications of the generic bulk conservation and interface equation for surfactant conservation

Equation	ϕ	$\boldsymbol{j}$	q	ϕ^Σ	$\boldsymbol{j}^\Sigma$
Specie conservation	Y_i	$-D_i \nabla Y_i$	r_i	Y_s^Σ	$-D_s^\Sigma \nabla_\Sigma c_s^\Sigma$

In order to specify the interface equation (3.38) for the transport of a surface active species we introduce Y_s^Σ which is the surface area related mass fraction of the surfactant. It can be converted into its surface area related concentration c_s^Σ (mol/m^2) by

$$Y_s^\Sigma = \frac{c_s^\Sigma \cdot M_s}{\rho^\Sigma} \ . \tag{3.40}$$

When also Fick's law [23] for the interfacial diffusive flux $\boldsymbol{j}^\Sigma$ is assumed (see table 3.3), the interface equation for a surfactant is given by

$$\frac{\partial c_s^\Sigma}{\partial t} + \nabla_\Sigma \cdot \left(c_s^\Sigma \boldsymbol{u}^\Sigma - \tilde{D}_s^\Sigma \nabla_\Sigma c_s^\Sigma \right) = [\![\tilde{D}_s \nabla c_s]\!] \cdot \boldsymbol{n}_\Sigma \quad \text{on } \Sigma \ . \tag{3.41}$$

In order to distinguish between the interfacial transport and the interface transmission condition this equation can be split up into two equations that are coupled through the newly introduced source term s^Σ:

$$\frac{\partial c_s^\Sigma}{\partial t} + \nabla_\Sigma \cdot \left(c_s^\Sigma \boldsymbol{u}^\Sigma - \tilde{D}_s^\Sigma \nabla_\Sigma c_s^\Sigma \right) = s^\Sigma \quad \text{on } \Sigma \ , \tag{3.42}$$

$$[\![\tilde{D}_s \nabla c_s]\!] \cdot \boldsymbol{n}_\Sigma = s^\Sigma \quad \text{on } \Sigma \ . \tag{3.43}$$

On the one hand, the interfacial transport equation (3.42) describes how a surfactant with a surface concentration c_s^Σ is transported along the interface due to convective and diffusive fluxes. Furthermore, it represents a dynamic boundary condition for (3.39) [61]. On the other hand, the interface transmission condition (3.43) describes how much surfactant is adsorbed and desorbed at the interface. Since surfactants typically occur only in one phase and the interface of a two-phase system, implying no surfactant transfer across the interface, (3.43) simplifies to

$$\tilde{D}_s \nabla c_s \cdot \boldsymbol{n}_\Sigma = s^\Sigma \quad \text{on } \Sigma \tag{3.44}$$

when assuming that the bulk phase containing the surfactant is Ω^+. For a rigorous derivation of these equations, please refer to [8], [77] and [61].

Similarly to the jump conditions defined previously, a relation is required that shows the relationship between the surfactant concentration on the interface c_s^Σ and the surfactant concentration in the immediate vicinity in the bulk c_s. Such a relationship is called adsorption isotherm and is dependent on the sorption model used [52]. In this work only a very simple diffusion controlled (fast) sorption model is presented, namely the Henry model [62], but other more elaborate models are available. The adsorption isotherm of this model relates the interface surfactant concentrations to the bulk surfactant concentrations by means of the Henry constant H_s:

$$c_s^\Sigma - c_s = c_s(H_s - 1) \ . \tag{3.45}$$

Consequently, the transport and sorption of surfactants is described sufficiently. Still missing is an equation that describes the surface tension as a function of the surfactant concentration on the boundary surface. This so called surface equation of state is again dependent on the sorption model used and reads for the Henry model

$$\sigma = \sigma_0 - RTc^\Sigma \ . \tag{3.46}$$

Here, σ_0 is the base surface tension, T is the absolute temperature of the system and R is the universal gas constant with $R = 8.3144\,{}^{\mathrm{J}}\!/_{\mathrm{mol\cdot K}}$. Equation (3.46) can then be substituted in the momentum transmission condition (3.30) and the pressure jump condition (3.31) where additional Marangoni stresses are considered subsequently.

4 Numerical Model

Since the behaviour of reactive two-phase flows is to be simulated, the mathematical model presented in chapter 3 must be solved. As this is not possible in an exact way, the Finite Volume Method (FVM) is used as the underlying numerical method for solving the system of the specified bulk transport equations. The principle of this method and its implementation in OpenFOAM has been explained in detail in many previous works (see [30, 54, 22, 83, 36, 38]). Nevertheless, a brief overview of the FVM discretisation process, which can be divided into spatial discretisation, temporal discretisation and equation discretisation, is given at the beginning of this chapter. The particularity of the equation discretisation process described here is that it is directly presented for the ALE formulation of the governing conservation equations. This is due to the fact that as an extension of the classical FVM the ALE-IT is used, in which the two phases are represented by two separate computing domains and a deformable interface. However, since interface gradients and interface divergence fields must also be computed, when the interface jump and transmission conditions or surfactants are taken into account, an additional numerical method complementing the FVM is required. The Finite Area Method (FAM) is used for this purpose, which is a modification of the FVM and operates on a two-dimensional curved surface instead of a three dimensional domain [32]. Since it basically works like the FVM, only on a lower dimensional domain, this method is not described in more detail. Instead, the volumetric coupling of the individual transport equations in the individual fluid phases is addressed in section 4.3. Subsequently, the interface coupling of the two phases will be discussed, which is realised in this thesis in form of the DNA. In particular, the newly formulated interface residuals, which can be used for convergence monitoring of the DNA, are presented in section 4.4.2. At the end of this chapter the algorithmic method of the ALE-IT and the application of a moving reference frame is described.

4.1 Spatial and temporal discretisation

Before the basic transport equations can be transformed into a corresponding system of algebraic equations, the solution domain in which they apply must first be discretised. Therefore, the continuous solution domain must be divided into a set of Control Volumes (CV), which do not overlap, are convex and fill the computational domain completely [36]. Such a set of CVs is known as computational mesh or just mesh. A typical polyhedral CV, also called cell, with volume V_p is displayed in figure 4.1. The cell centroid P represents one of many computational points at which the solution is sought. It is connected to one of its neighbour cells, with centroid N, over a flat face f. Such faces are defined by their surface area vector $\boldsymbol{S}_f$, which are located a the face centre, are normal to the face and have a magnitude equal to the face area. To save notation, the face centre of face f is also denoted by f.

Just as the solution domain must be discretised, so must the time over which the solution is computed. For this purpose, the timeline is divided into finite intervals Δt, also called time steps, which are used for marching in time from the initial condition.

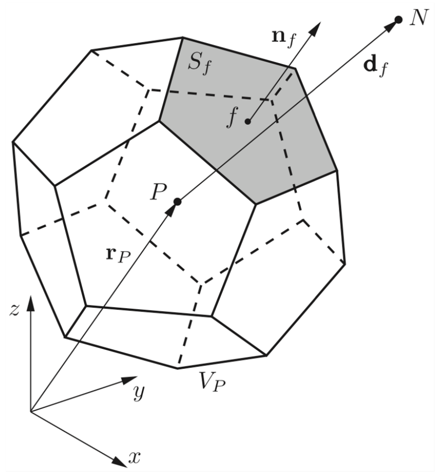


Figure 4.1: Polyhedral Control Volume [81, p. 72]

4.2 Equation discretisation

4.2.1 Arbitrary Lagrangian Eulerian formulation

As the solution domains Ω^+ and Ω^- are already separate in the mathematical model (see chapter 3) each phase is assigned its own computational mesh. However, to represent the motion of the interface, these computational meshes must be able to move and deform according to the interface motion. This implies that the control volumes, that form these meshes, can also move and deform. Thus, the typically used Eulerian formulation of the conservation equations presented in chapter 3 cannot be used. Instead, the conservation equations are reformulated into the so-called Arbitrary Lagrangian-Eulerian formulation which was originally introduced by Hirt et al. [31]. Since all derived conservation equations are specifications of the prototypical bulk equation (3.18), we can perform the reformulation and discretisation on this equation which then applies analogously to the specific conservation equations for mass (3.24), momentum (3.25) and species (3.34). For this purpose, we use a control volume $V_p(t)$ as described previously whose boundary surface $\partial V_p(t)$ with outward normal $\boldsymbol{n}$ can move freely with velocity $\boldsymbol{w}$. Integrating the prototypical bulk equation (3.18) over the described control volume yields its ALE form

$$\underbrace{\frac{d}{dt}\int_{V_p} \rho\phi dV}_{\text{temporal term}} + \underbrace{\int_{V_p} \nabla\cdot\left(\rho\left(\boldsymbol{u}-\boldsymbol{w}\right)\phi\right) dV}_{\text{convection term}} = -\underbrace{\int_{V_p} \nabla\cdot\boldsymbol{j} dV}_{\text{diffusion term}} + \underbrace{\int_{V_p} q dV}_{\text{source term}} \quad . \tag{4.1}$$

Now it also becomes clear why this form is called ALE formulation, because for $\boldsymbol{w} = 0$ the formulation is purely Eulerian and for $\boldsymbol{w} = \boldsymbol{u}$ the formulation is purely Lagrangian. Complementary to (4.1) an additional relation has to be added to the formulation. It reads

$$\frac{d}{dt}\int_{V_p} dV - \int_{\partial V_p} \boldsymbol{w}\cdot\boldsymbol{n} dA = 0 \tag{4.2}$$

and is called Space (Geometric) Conservation law (SCL) because it states that the control volume change in time has to be equal to the volume swept by its boundary with velocity $\boldsymbol{w}$. As shown in [17] it is important to add it to the formulation in order to ensure that the CV motion due to numerically approximated surface velocities does not create artificial mass sources/sinks, which can cause accumulation of solution errors or even divergence of the computations.

These two equations are the starting point for the equation discretization. First, the discretization of the individual terms of (4.1) will be discussed. Subsequently, the completely discretised generic transport equation is presented before the discretization of the SCL is dealt with.

4.2.2 Temporal term

The discretisation of the temporal term is done by integrating the whole equation (4.1) in time and then using numerical methods like Runge-Kutta methods or backward differentiation formulas. For the approximation of the volume integral, another numerical integration method is needed. In the context of the FVM, the midpoint-rule is typically used. If then, for example, the implicit three time-level Backward Differencing Scheme (BDS) is utilised, the term results in

$$\frac{d}{dt}\int_{V_p} \rho\phi dV \approx \rho_P \frac{3\phi_P^n V_P^n - 4\phi_P^o V_P^o + \phi_P^{oo} V_P^{oo}}{2\Delta t} \ . \tag{4.3}$$

Here, the subscript P represents the CV-centre value and the superscripts n, o and oo represent values evaluated at the new time instance t^n and two previous time instances t^o and t^{oo}.

4.2.3 Convection term

The volume integral of the convection term is first converted into a surface integral using the Gauss divergence theorem. This can then again be approximated with the midpoint-rule over all CV faces, which leads to the following expression:

$$\int_{V_p^n} \nabla \cdot \left(\rho \left(\boldsymbol{u}^n - \boldsymbol{w}^n\right) \phi^n\right) dV = \int_{\partial V_p^n} \rho \left(\boldsymbol{u}^n - \boldsymbol{w}^n\right) \phi^n \cdot \boldsymbol{n}^n dA \approx \sum_f \rho_f \phi_f^n (\boldsymbol{u}_f^n - \boldsymbol{w}_f^n) \cdot \mathbf{S}_f^n \ . \tag{4.4}$$

Here, the subscript f represents the CV-face centre value, which can be obtained through interpolation of the corresponding cell centre values. However, special attention must be given to the interpolation of ϕ_f in this term. This is due to the fact that advection is a highly directional process, which has to be accounted for in the interpolation method used [54]. Therefore special advection interpolation schemes are available. Furthermore, it is possible to distinguish between the cell-face volume flux due to fluid motion

$$\tilde{\Phi} = \boldsymbol{u}_f^n \cdot \mathbf{S}_f^n \ , \tag{4.5}$$

the volume flux due to mesh motion

$$\dot{V}_f^n = \boldsymbol{w}_f^n \cdot \mathbf{S}_f^n \tag{4.6}$$

and the volume flux relative to the mesh motion

$$\hat{\Phi} = (\boldsymbol{u}_f^n - \boldsymbol{w}_f^n) \cdot \mathbf{S}_f^n \ . \tag{4.7}$$

4.2.4 Diffusion term

Analogous to the discretisation procedure used for the convection term, the volume integral of the diffusion term is first converted into a surface integral using the Gauss divergence theorem and then approximated with the midpoint rule:

$$-\int_{V_p^n} \nabla \cdot \boldsymbol{j}^n dV = -\int_{\partial V_p^n} \boldsymbol{j}^n \cdot \boldsymbol{n}^n dA \approx -\sum_f \boldsymbol{j}_f^n \cdot \mathbf{S}_f^n \ . \tag{4.8}$$

Thus, the diffuse flux $\boldsymbol{j}_f^n$ on the cell surfaces centre must be determined in this expression. Since the diffusive flux is typically modelled by some kind of the gradient diffusion hypothesis

$$\boldsymbol{j}_f^n = -\Gamma_f^n \nabla \phi_f^n \ , \tag{4.9}$$

which is also the case for the governing equations presented in sections 3.2 and 3.3, this involves the computation of surface normal gradients of ϕ. This is done by utilising finite differencing schemes for numerical gradient computation. As mesh orthogonality, meaning $\boldsymbol{d}_f$ and $\boldsymbol{n}_f$ being parallel in figure 4.1, is more of an exception rather than the rule, it is important to consider the non-orthogonality in these surface normal gradient computations [36].

4.2.5 Source term

If q depends on ϕ in a nonlinear manner it first needs to be linearised in order to obtain a system of linear equations at the end of the discretisation process [60, 36, 38]:

$$q(\phi) = q_e + q_P \phi \ . \tag{4.10}$$

Then again the midpoint rule can be used to approximate the volume integral resulting in the expression

$$\int_{V_p^n} q_e + q_P \phi dV \approx q_e V_p^n + q_P \phi_p^n V_p^n \ . \tag{4.11}$$

4.2.6 Fully discretised generic transport equation

Assembling the generic transport equation again from the individual discretised terms, the following expression is obtained:

$$\rho_P \frac{3\phi_P^n V_P^n - 4\phi_P^o V_P^o + \phi_P^{oo} V_P^{oo}}{2\Delta t} + \sum_f \rho_f \phi_f^n (\boldsymbol{u}_f^n - \boldsymbol{w}_f^n) \cdot \mathbf{S}_f^n = \sum_f \Gamma_f^n \nabla \phi_f^n \cdot \mathbf{S}_f^n + q_e V_p^n + q_P \phi_p^n V_p^n \ . \tag{4.12}$$

If now the surface values are calculated by interpolation of the cell values, using a suitable differentiation scheme for the convection term and utilising a finite differentiation scheme for the surface normal gradient, the above equation can be converted into the form

$$a_P^n \phi_P^n + \sum_N a_N^n \phi_N^n = b_P \ . \tag{4.13}$$

Here, all explicitly known terms were combined into a source term b_P. In addition, N corresponds to the neighbouring cells of the control volume P which form the computational molecule of the numerical method

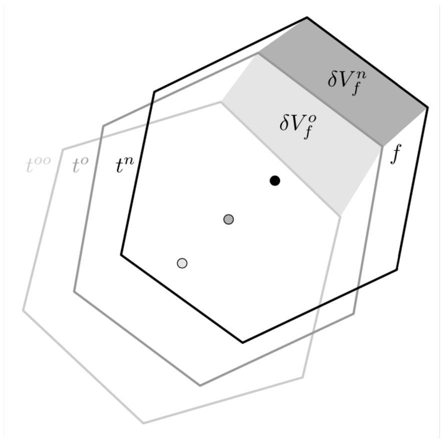


Figure 4.2: Volumes swept by face f during the movement between successive positions [27, 49]

due to the chosen interpolation and differentiation methods. Assembling (4.13) for every CV then leads to a system of linear algebraic equations of type

$$\boldsymbol{A}\boldsymbol{\phi} = \boldsymbol{b}\ . \tag{4.14}$$

This system can subsequently be solved by utilising iterative solving methods for large systems of linear equations like the ones presented in [65].

4.2.7 Space (geometric) conservation law

For the discretisation of the SCL, the same methods as for the temporal and convection term of the generic bulk transport equation can be used. Again applying the BDS for the temporal integration the following discretised form of the SCL is obtained:

$$\frac{3V_P^n - 4V_P^o + V_P^{oo}}{2\Delta t} = \sum_f \boldsymbol{w}_f^n \cdot \boldsymbol{S}_f^n\ . \tag{4.15}$$

With the introduction of the swept volumes ∂V_f (see figure 4.2) for which

$$V_P^n - V_P^o = \sum_f \delta V_f^n \tag{4.16}$$

and

$$V_P^o - V_P^{oo} = \sum_f \delta V_f^o \tag{4.17}$$

applies, (4.15) can be expressed as

$$\sum_f \frac{3\delta V_f^n - \delta V_f^o}{2\Delta t} = \sum_f \boldsymbol{w}_f^n \cdot \boldsymbol{S}_f^n =: \sum_f \dot{V}_f^n\ . \tag{4.18}$$

This expression can then be substituted into (4.12) which brings the advantage that the SCL is always implicitly met and $\boldsymbol{w}_f^n$ does not have to be computed explicitly.

4.3 Volumetric coupling

As presented in chapter 3 each phase is governed by the coupled transport equations for mass (3.24), momentum (3.25), and reactive/surface-active species (3.34)/(3.39). Accordingly, the sought quantities are the velocity field $\boldsymbol{u}$, the pressure field p and the concentration fields c_i or c_s. All equations are coupled in terms of the velocity field. The pressure and velocity field are implicitly coupled in the momentum equation and the concentration fields are coupled due to reactions accounted for in the source terms. In this section it will be discussed how this volumetric coupling is treated numerically, starting with the pressure velocity coupling in the Navier-Stokes equations. Subsequently, the coupling of the reactive/surface-active species transport to the velocity field and the coupling of the reactive species to each other will be explained.

4.3.1 Pressure-velocity coupling

The pressure and velocity field of an incompressible fluid can be computed by solving the Navier-Stokes equations, which are presented in section 3.2 and read in the ALE formulation:

$$\underbrace{\frac{d}{dt}\int_{V_p} dV - \int_{V_p} \nabla \cdot \boldsymbol{w} dV}_{=0(\mathrm{SCL})} + \int_{V_p} \nabla \cdot \boldsymbol{u} dV = 0 \, , \tag{4.19}$$

$$\frac{d}{dt}\int_{V_p} \boldsymbol{u} dV + \int_{V_p} \left(\nabla \cdot (\boldsymbol{u} - \boldsymbol{w})\right) \boldsymbol{u} dV = -\int_{V_p} \nabla \hat{p} dV - \int_{V_p} \nu \Delta \boldsymbol{u} dV \; . \tag{4.20}$$

Since it is more convenient for the numerical formulation, the total pressure in the momentum equation has been reformed by introducing the kinematic pressure

$$\hat{p} = \frac{\tilde{p}}{\rho} = \frac{p - \rho \boldsymbol{g} \cdot \boldsymbol{x}}{\rho} \; . \tag{4.21}$$

If one attempts to solve this set of equations two major problems arise. The first is that the momentum equation (3.25), with $\boldsymbol{u}$ as its primary variable, is nonlinear in the convection term. This can usually be dealt with by utilising a Picard type iteration, where the volume fluxes in the convection terms are used from a previous iteration step [54, p. 562]. The second problem is that, in addition to a strong physical coupling between pressure and velocity, pressure does not appear as a primary variable in one of the equations [54, p. 561]. To overcome this problem a pressure equation can be derived from the mass and momentum equation, which can then be solved in addition to the momentum equation. After deriving this pressure equation in the following, two procedures are presented that can be used to numerically account for the strong coupling between the pressure and velocity when solving for both.

For the derivation of the pressure equation we start with the discretised mass conservation equation

$$\sum_f \underbrace{\boldsymbol{u}_f^n \cdot \boldsymbol{S}_f^n}_{\Phi_f^n} = 0 \; . \tag{4.22}$$

In order to get a substitution for $\boldsymbol{u}_f^n$, the momentum conservation equation (4.20) is discretised as described in section 4.2 and the volume fluxes are assumed to be explicitly known from a previous iteration/time step. This leads to the expression

$$\boldsymbol{a}_P^n \boldsymbol{u}_P^n + \sum_N \boldsymbol{a}_N^n \boldsymbol{u}_N^n = -V_P^n (\nabla \hat{p}^n)_P + \boldsymbol{b}_P \; , \tag{4.23}$$

which can be converted to the form

$$\boldsymbol{u}_P^n = \underbrace{-(\boldsymbol{a}_P^n)^{-1}\sum_N \boldsymbol{a}_N^n \boldsymbol{u}_N^n}_{\boldsymbol{H}_P(\boldsymbol{u})} - (\boldsymbol{a}_P^n)^{-1} V_P^n (\nabla \hat{p}^n)_P + \underbrace{(\boldsymbol{a}_P^n)^{-1} b_P}_{\tilde{\boldsymbol{b}}_P} \; . \tag{4.24}$$

Unfortunately, $\boldsymbol{u}_f^n$ can not be directly approximated by interpolating (4.24) to the face as this results in the pressure chequerboard problem described in [83]. Instead, the interpolation method developed by Rhie and Chow [63] is used. It is based on the construction of a pseudo-momentum equation on the cell face similar to (4.24):

$$\boldsymbol{u}_f^n = -(\boldsymbol{a}_f^n)^{-1}\boldsymbol{H}_f(\boldsymbol{u}) - (\boldsymbol{a}_f^n)^{-1} V_f^n (\nabla \hat{p}^n)_f + \tilde{\boldsymbol{b}}_f \; . \tag{4.25}$$

Now the same equation is formulated with interpolated values from the cell centres, where the overbar indicates the interpolation:

$$\overline{\boldsymbol{u}}_f^n = -\overline{(\boldsymbol{a}_f^n)^{-1}}\,\overline{\boldsymbol{H}_f(\boldsymbol{u})} - \overline{(\boldsymbol{a}_f^n)^{-1}}\,\overline{V_f^n}\,\overline{(\nabla \hat{p}^n)_f} + \overline{\tilde{\boldsymbol{b}}_f} \; . \tag{4.26}$$

Subtracting (4.26) from (4.25) and assuming

$$(\boldsymbol{a}_f^n)^{-1}\boldsymbol{H}_f(\boldsymbol{u}) \approx \overline{(\boldsymbol{a}_f^n)^{-1}}\,\overline{\boldsymbol{H}_f(\boldsymbol{u})} \; , \tag{4.27}$$

$$(\boldsymbol{a}_f^n)^{-1} V_f^n \approx \overline{(\boldsymbol{a}_f^n)^{-1}}\,\overline{V_f^n} \; , \tag{4.28}$$

$$\tilde{\boldsymbol{b}}_f \approx \overline{\tilde{\boldsymbol{b}}_f} \tag{4.29}$$

then gives

$$\boldsymbol{u}_f^n = \overline{\boldsymbol{u}}_f^n - \overline{(\boldsymbol{a}_f^n)^{-1}}\,\overline{V_f^n}\left((\nabla \hat{p}^n)_f - \overline{(\nabla \hat{p}^n)_f}\right) \; . \tag{4.30}$$

This expression for the face velocity can also be written as

$$\boldsymbol{u}_f^n = \overline{(\boldsymbol{a}_f^n)^{-1}}\,\overline{\boldsymbol{H}_f(\boldsymbol{u})} - \overline{(\boldsymbol{a}_f^n)^{-1}}\,\overline{V_f^n} (\nabla \hat{p}^n)_f + \overline{\tilde{\boldsymbol{b}}_f} \tag{4.31}$$

if (4.26) is substituted back into it. Using either of the formulations (4.30) or (4.31) in the discretised mass conservation (4.22) gives rise to the pressure equation:

$$\begin{aligned}\sum_f \overline{(\boldsymbol{a}_f^n)^{-1}}\,\overline{V_f^n} (\nabla \hat{p}^n)_f \cdot \boldsymbol{S}_f^n &= \sum_f \overline{\boldsymbol{u}}_f^n \cdot \boldsymbol{S}_f^n + \sum_f \overline{(\boldsymbol{a}_f^n)^{-1}}\,\overline{V_f^n}\,\overline{(\nabla \hat{p}^n)_f} \cdot \boldsymbol{S}_f^n \\ &= \sum_f \left[\overline{(\boldsymbol{a}_f^n)^{-1}}\,\overline{\boldsymbol{H}_f(\boldsymbol{u})} + \overline{\tilde{\boldsymbol{b}}_f}\right] \cdot \boldsymbol{S}_f^n \; .\end{aligned} \tag{4.32}$$

In this formulation the surface normal gradient $(\nabla \hat{p}^n)_f \cdot \boldsymbol{S}_f^n$ can again be computed by utilising finite differencing methods while accounting for non-orthogonality of the mesh. The interpolated surface normal gradient $\overline{(\nabla \hat{p}^n)_f} \cdot \boldsymbol{S}_f^n$ is handled explicitly by computing the cell-centre gradient $(\nabla \hat{p}^n)_P$ from old time values and interpolating it to the surface.

As already mentioned, the pressure equation (4.32) can be used in addition to the momentum equation (4.23)/(4.24) to solve for $\boldsymbol{u}_P$ and $\hat{p}_P$. How the coupling between these equations can numerically be resolved either semi-implicit (segregated) or fully-implicit (fully-coupled) is discussed in the following.

Semi-implicit (segregated) coupling

The pressure-velocity coupling in transient flows can be resolved in a segregated way by utilising the PISO-Algorithm proposed by Issa [34]. This algorithm is based on the SIMPLE-Algorithm originally developed by Patanker [60] for steady-state flows and therefore relies on a predictor-corrector procedure which can be described as follows:

1. At the beginning of the algorithm an approximation of the velocity field is computed by solving the momentum equation (4.23) and using the pressure field and the volume fluxes from the previous time step. Therefore, this step is called *momentum predictor step*.
2. The approximated velocity field can then be used to construct the $\boldsymbol{H}_P(\boldsymbol{u})$ operator. Together with the $\boldsymbol{a}_P^n$, which is known from the momentum equation of the momentum predictor step, the pressure equation (4.32) can be assembled. The solution of this pressure equation gives a first estimate of the pressure field at the new time step. Therefore, this step is called *pressure solution step*.
3. With this new pressure field the initially approximated velocity field can be corrected in an explicit manner by using equation (4.24) and neglecting all terms but the term containing the pressure gradient. Therefore, this step is called *explicit velocity correction step*. The negligence of the terms in the explicit velocity correction effectively assumes that the whole velocity error is due to the error in the pressure term [36]. As this is not fully true it is necessary to update $\boldsymbol{H}_P(\boldsymbol{u})$ with the new corrected velocity field, return to step 2 and and assemble and solve the pressure equation again. This should then be repeated until a pre-determined tolerance is reached.
4. After the initially approximated pressure and velocity fields are sufficiently corrected, new conservative volume fluxes are computed by calculating the scalar product of either (4.30) or (4.31) with the face area vector $\boldsymbol{S}_f^n$. These fluxes can then again be used in the momentum and pressure equations in the next time step.

Segregated algorithms like the presented PISO-Algorithm or the SIMPLE-Algorithm and their derivatives are still the most widely used methods of resolving the pressure-velocity coupling. They have been studied to a high degree and have proven themselves many times in their application.

Fully-implicit (fully-coupled) coupling

In contrast to the segregated approach described above, where the pressure term in the momentum equation and the velocity term in the pressure equation are handled explicitly, one can solve these equations simultaneously while treating the mentioned terms implicit. Such a solution procedure for unstructured grids was derived by Mangani et al. [48], who based their work on the previous publications of Darwish et al. [14], Webster [87] and Lonsdale [47]. Following their method and approximating the pressure gradient term in the momentum equation implicitly, (4.23) can be written as

$$\boldsymbol{a}_P^{\boldsymbol{uu}} \boldsymbol{u}_P^n + \underline{\boldsymbol{a}_P^{\boldsymbol{u}p} \hat{p}_P^n} + \sum_N \boldsymbol{a}_N^{\boldsymbol{uu}} \boldsymbol{u}_N^n + \underline{\sum_N \boldsymbol{a}_N^{\boldsymbol{u}p} \hat{p}_N^n} = b_P^{\boldsymbol{u}} \,. \tag{4.33}$$

Here, the underlined terms represent the pressure gradient term in its discretised form. In the pressure equation a finite differencing scheme for the surface normal gradient $(\nabla \hat{p}^n)_f$ can be used while accounting for mesh non-orthogonality. When also computing $\overline{\boldsymbol{u}}_f^n$ by interpolation and treating $\overline{(\nabla \hat{p}^n)_f}$ explicitly by calculating it from previously obtained values, the pressure equation (4.32) results in the form

$$\underline{\boldsymbol{a}_P^{p\boldsymbol{u}} \cdot \boldsymbol{u}_P^n} + a_P^{pp} \hat{p}_P^n + \underline{\sum_N \boldsymbol{a}_N^{p\boldsymbol{u}} \cdot \boldsymbol{u}_N^n} + \sum_N a_N^{pp} \hat{p}_N^n = \boldsymbol{b}_P^p \,. \tag{4.34}$$

In the source term $\boldsymbol{b}_P^p$ both the non-orthogonal correction of $(\nabla \hat{p}^n)_f$ and the explicit calculated $\overline{(\nabla \hat{p}^n)_f}$ are summarised. The underlined terms represent the implicit interpolated velocities. Combining the two equations (4.33) and (4.34) then yields a system of equation

$$\begin{bmatrix} a_P^{uu} & a_P^{uv} & a_P^{uw} & a_P^{up} \\ a_P^{vu} & a_P^{vv} & a_P^{vw} & a_P^{vp} \\ a_P^{wu} & a_P^{wv} & a_P^{ww} & a_P^{wp} \\ a_P^{pu} & a_P^{pv} & a_P^{pw} & a_P^{pp} \end{bmatrix} \cdot \begin{bmatrix} u_P \\ v_P \\ w_P \\ p_P \end{bmatrix} + \sum_N \begin{bmatrix} a_N^{uu} & a_N^{uv} & a_N^{uw} & a_N^{up} \\ a_N^{vu} & a_N^{vv} & a_N^{vw} & a_N^{vp} \\ a_N^{wu} & a_N^{wv} & a_N^{ww} & a_N^{wp} \\ a_N^{pu} & a_N^{pv} & a_N^{pw} & a_N^{pp} \end{bmatrix} \cdot \begin{bmatrix} u_N \\ v_N \\ w_N \\ p_N \end{bmatrix} = \begin{bmatrix} b_P^u \\ b_P^v \\ b_P^w \\ b_P^p \end{bmatrix} \tag{4.35}$$

where the values of the different coefficients a_P and a_N are dependent on the used discretisation schemes. Assembling this pressure velocity equation system for all cells in the mesh results again in a system of equations that can be written in the form

$$\boldsymbol{A}\boldsymbol{\phi} = \boldsymbol{b} \,. \tag{4.36}$$

Solving this system implies solving for all variables $\boldsymbol{\phi} = (\boldsymbol{u}, p)$ simultaneously. However, since $\overline{(\nabla \hat{p}^n)_f}$ is treated explicitly in this formulation, an iterative procedure has to be used in order to reach a converged solution of the system in every time step.

Even though this method is more memory demanding than its segregated counterpart, it was shown that it is more stable and it convergence rate scales better with increasing mesh sizes [84, 14]. For these reasons and the constantly increasing availability of computer memory, fully-coupled methods gain increasingly more interest [21].

4.3.2 Species coupling

The reactive species transport equation reads in ALE-formulation

$$\frac{d}{dt} \int_{V_p} c_i dV + \underline{\int_{V_p} \nabla \cdot (c_i (\boldsymbol{u} - \boldsymbol{w})) \, dV} = - \int_{V_p} \tilde{D}_i \Delta c_i dV + \underline{\underline{\int_{V_p} \tilde{r}_i dV}} \,. \tag{4.37}$$

Two types of volumetric coupling occur during the transport of reactive species. On the one hand, the advective transport of the species is coupled to the velocity field in the respective bulk (single underlined term). On the other hand, the concentrations of the individual species are coupled to one another via reactions (double underlined term). If one assumes that the considered reactive species are ideally diluted, it can be expected that they do not influence the hydrodynamic behaviour due to their transport. Therefore, the species are advected like passive particles which in turn leads to the fact that the coupling between the species transport and the underlying velocity field is unidirectional. Consequently the volumetric coupling between the species transport equations and the Navier-Stokes equations can be designed in a very simple way. For each time step, the Navier-Stokes equations are first solved as explained in the previous subsection. This leads to the velocity field and the field of conservative volume fluxes Φ. Subsequently, these volume fluxes can be used in the discretised convection term when solving the discretised version of (4.37). However, before (4.37) can actually be solved, the reaction term has to be determined. As already mentioned, this term results from the reaction of the different species and therefore couples all reactive species transport equations with each other. Furthermore, the reaction rate itself is dependent on the concentration of the reacting species. To simplify this circumstance, the reaction rate is treated explicitly and is calculated using the concentration fields of the previous time step and a specified reaction mechanism.

4.4 Dirichlet-Neumann interface coupling

The Dirichlet–Neumann Algorithm belongs to the class of nonoverlapping Domain Decomposition Methods (DDM). Such methods were originally developed to speed up the numerical solution of boundary value problems by splitting the solution domain into nonoverlapping subdomains on which smaller boundary value problems are defined [19]. This involves specifying boundary conditions on the new boundaries that connect the solutions on the subdomains. Then an iterative procedure is applied where the subproblems are solved alternately while the connecting boundary conditions are updated in between. The individual names of these methods indicate the way in which these connecting boundary conditions are defined and how the iteration procedure is performed. Thus, the solutions on the subdomains, using the Dirichlet–Neumann Algorithm, are connected via both Dirichlet and Neumann boundary conditions where for each connecting boundary the Dirichlet condition is applied on one side and the Neumann condition is applied on the other [80]. The respective iteration procedure of the DNA is as follows: First the solution ϕ_D^n on the subdomain with the Dirichlet boundary condition is sought. Then the Neumann boundary condition on the other side of the connecting boundary is updated using the previously obtained solution ϕ_D^n. Subsequently, the solution on the subdomain with the updated Neumann boundary condition is calculated, which we call ϕ_N^n. Before solving for the next iterate ϕ_D^{n+1} on the Dirichlet side, the Dirichlet boundary condition is updated using ϕ_N^n. This procedure is then repeated until convergence. It is also possible to perform the prescribed algorithm in the reverse order. In this case the starting point is the region on which the Neumann boundary condition is applied. This is then referred to as a Neumann-Dirichlet Algorithm (NDA). Even though these two algorithms may have different convergence rates [80], both will be named DNA in the following if not specified explicitly.

Considering the generic sharp interface model derived in section 3.1

$$
\begin{aligned}
\frac{\partial \rho^+\phi^+}{\partial t} + \nabla\cdot\left(\rho^+\boldsymbol{u}^+\phi^+\right) &= -\nabla\cdot\boldsymbol{j}^+ + q^+ && \text{in } \Omega^+ \, , \\
[\![f]\!] &= \mathcal{J}, && \text{on } \Sigma \, , \\
[\![\rho\phi(\boldsymbol{u}^\Sigma-\boldsymbol{u})-\boldsymbol{j}]\!]\cdot\boldsymbol{n}_\Sigma &= \nabla_\Sigma\cdot\boldsymbol{j}^\Sigma - \kappa\left(\boldsymbol{j}^\Sigma\cdot\boldsymbol{n}_\Sigma\right) && \text{on } \Sigma \, , \\
\frac{\partial \rho^-\phi^-}{\partial t} + \nabla\cdot\left(\rho^-\boldsymbol{u}^-\phi^-\right) &= -\nabla\cdot\boldsymbol{j}^- + q^- && \text{in } \Omega^- \, ,
\end{aligned}
\tag{4.38}
$$

it becomes obvious that the problem is inherently one in which the solution domain is divided into two subdomains. Here, the connecting boundary corresponds to the sharp interface and the connecting boundary conditions are directly given in form of the derived jump and transmission condition. Thus, DDMs are in principle well suited to numerically resolve the interface coupling. Even though, no rigorous proof for the convergence of DDMs exists for such a system, they have been extensively tested for the simulation of interface coupled multiphysics systems and are able to produce good results (e.g. [44, 53, 86, 81]). It should also be noted that these DDMs, and consequently the DNA considered here, are explicit/partitioned interface coupling procedures, since the equation systems of the two phases are solved and the boundary conditions at the interface are updated iteratively. However, before the DNA can be applied to the generic sharp interface problem (4.38), appropriate Dirichlet and Neumann boundary conditions must be derived from the jump and transmission conditions. These derivations for the generic case as well as for the special cases of the Navier-Stokes equations and the conservation of species are explained in the next subsection. Furthermore, it must be clarified when the result of the DNA is considered converged and the next time step can be computed. For this purpose, so-called interface residuals are formulated in subsection 4.4.2 on the basis of which the convergence of the solution of the DNA can be evaluated.

4.4.1 Coupled Dirichlet-Neumann boundary conditions

Generic coupled boundary conditions

In order to derive generic Dirichlet and Neumann boundary conditions for the generic sharp interface model, a few assumptions have to be made. Of course, these limit the general applicability of the model. However, the assumptions are such that the model is still valid for a large number of problems which can be modelled with specialisations of it.

The first assumption is that ϕ itself is the only primitive field f for which a jump condition has to be defined. This directly leads to the Dirichlet boundary condition of ϕ that can be written as

$$\begin{aligned} & [\![\phi]\!] = \mathcal{J}_\phi \\ \Leftrightarrow \quad & \phi^+ = \mathcal{J}_\phi + \phi^- \ . \end{aligned} \tag{4.39}$$

The second assumption is that no mass transfer happens across the interface. Thirdly and finally, it is also assumed that the diffusive flux can be modelled with the gradient hypothesis $\boldsymbol{j} = -\Gamma_\phi \nabla\phi$. Applying these assumptions to the generic interface transmission condition (3.19), one obtains the Neumann boundary condition for ϕ:

$$\begin{aligned} & [\![\Gamma_\phi \nabla\phi]\!] \cdot \boldsymbol{n}_\Sigma = \mathcal{F}_\phi \\ \Leftrightarrow \quad & \nabla\phi^+ \cdot \boldsymbol{n}_\Sigma = \frac{1}{\Gamma_\phi^+} \left(\mathcal{F}_\phi + \Gamma_\phi^- \nabla\phi^- \cdot \boldsymbol{n}_\Sigma \right) \ . \end{aligned} \tag{4.40}$$

Here all terms on the right hand side were summarised in the generic flux jump $\mathcal{F}_\phi$.

Regarding the naming of the two types of coupled boundary conditions, the terms jump and flux boundary conditions will be used synonymously for Dirichlet and Neumann boundary conditions in the further course of the work.

To confirm this generic formulation, the next two subsections show how the special Dirichlet/jump and Neumann/flux boundary conditions for pressure, velocity and species concentration can be related back to it.

Coupled boundary conditions for the Navier-Stokes equations

For the pressure-velocity coupled system of the Navier-Stokes equations, both Dirichlet and Neumann boundary conditions are needed for the two primitive fields of pressure and velocity. Starting with the Dirichlet condition for the velocity, the velocity jump condition (3.32) directly provides the needed relation:

$$[\![\boldsymbol{u}]\!] = \underbrace{0}_{\mathcal{J}_u} \ . \tag{4.41}$$

The same applies to the pressure, where the already derived pressure jump condition (3.31) can be used for the Dirichlet boundary condition:

$$[\![p]\!] = \underbrace{\sigma\kappa - 2[\![\mu]\!]\nabla_\Sigma \cdot \boldsymbol{u}}_{\mathcal{J}_p} \ . \tag{4.42}$$

Analogous to the derivation of the pressure jump condition in section 3.2, the momentum transmission condition (3.30) is again used as a starting point for the derivation of a Neumann velocity condition. Rearranging it and inserting the pressure jump equation (3.31) gives

$$[\![\mu(\nabla\boldsymbol{u})^\top]\!] \cdot \boldsymbol{n}_\Sigma = -[\![\mu]\!] \left(\nabla_\Sigma \cdot \boldsymbol{u}\right) \boldsymbol{n}_\Sigma - [\![\mu]\!] \left(\nabla_\Sigma \boldsymbol{u}\right) \cdot \boldsymbol{n}_\Sigma - \nabla_\Sigma \sigma \ . \tag{4.43}$$

It represents the tangential component of the momentum transmission condition [61]. Assuming a spin free flow field along the interface, which implies a vanishing antisymmetric velocity gradient tensor (spin tensor) [2]

$$\boldsymbol{S} = \frac{1}{2}\left(\nabla\boldsymbol{u} - (\nabla\boldsymbol{u})^\top\right) = \boldsymbol{0} \quad \Rightarrow \quad \nabla\boldsymbol{u} = (\nabla\boldsymbol{u})^\top \ , \tag{4.44}$$

one obtains the coupled Neumann velocity boundary condition:

$$[\![\underbrace{\mu}_{\Gamma_{\boldsymbol{u}}} \nabla\boldsymbol{u}]\!] \cdot \boldsymbol{n}_\Sigma = \underbrace{-[\![\mu]\!] \left(\nabla_\Sigma \cdot \boldsymbol{u}\right) \boldsymbol{n}_\Sigma - [\![\mu]\!] \left(\nabla_\Sigma \boldsymbol{u}\right) \cdot \boldsymbol{n}_\Sigma - \nabla_\Sigma \sigma}_{\mathcal{F}_{\boldsymbol{u}}} \ . \tag{4.45}$$

A flux boundary condition for the pressure can not be directly derived from the momentum transmission condition. Nevertheless, it can be obtained by taking the jump of the momentum equation (3.25) and only considering its normal component

$$\left[\!\!\left[\frac{\partial\boldsymbol{u}}{\partial t} + \left(\nabla \cdot \boldsymbol{u}\right)\boldsymbol{u}\right]\!\!\right] \cdot \boldsymbol{n}_\Sigma = \left[\!\!\left[-\frac{1}{\rho}\nabla \cdot p\right]\!\!\right] \cdot \boldsymbol{n}_\Sigma + [\![\nu\Delta\boldsymbol{u}]\!] \cdot \boldsymbol{n}_\Sigma + [\![\boldsymbol{g}]\!] \cdot \boldsymbol{n}_\Sigma \ . \tag{4.46}$$

Because of the sharp interface assumption, the jump of the gravitational term is zero. The left-hand side of this equation also vanishes since the continuity of the velocity across the interface leads to continuous normal accelerations across the interface [71]. Therefore, the pressure flux boundary condition can be written as

$$\left[\!\!\left[\frac{1}{\rho}\nabla \cdot p\right]\!\!\right] \cdot \boldsymbol{n}_\Sigma = \underbrace{[\![\nu\Delta\boldsymbol{u}]\!] \cdot \boldsymbol{n}_\Sigma}_{\mathcal{F}_p} \ . \tag{4.47}$$

Here, $1/\rho$ is not a diffusion coefficient in the physical sense. Nevertheless, the form of this pressure Neumann boundary condition, as well as the other boundary conditions for pressure and velocity, correspond to the form of their generic counterparts (4.39) and (4.40). This is also indicated through the underbraced terms in the equations above.

Coupled boundary conditions for species transport

In the case of the reactive species transport, no further derivations need to be made. The jump (3.36) and transmission condition (3.35), which have already been formulated in section 3.3.1, can be used as Dirichlet and Neumann boundary conditions for the species concentration without any further steps:

$$[\![c_i]\!] = \underbrace{c_i^-(H-1)}_{\mathcal{J}_c} \ , \tag{4.48}$$

$$[\![\underbrace{\tilde{D}_i}_{\Gamma_c} \nabla c_i]\!] \cdot \boldsymbol{n}_\Sigma = \underbrace{0}_{\mathcal{F}_c} \ . \qquad (4.49)$$

Again, it can be easily seen that these boundary conditions exactly match the form of the generic boundary conditions for $\phi = c_i$.

Application of coupled boundary conditions in the DNA

As presented in [86] and noted in [81] the DNA can be subject to a stability criterion which indicates the convergence or divergence of the interface coupling procedure. It was proven that the choice of sides on which the Dirichlet or Neumann conditions are applied significantly influences the stability of the procedure. For example Tuković and Jasak [81] noted that it is necessary to apply the pressure Dirichlet and the velocity Neumann condition at the interface side of the denser fluid and the pressure Neumann and the velocity Dirichlet condition on the other. However, there are situations in which it is not always possible to freely select in which order the Neumann and Dirichlet conditions are applied. Therefore, it can be possible to stabilise an unstable DNA through relaxation of the Dirichlet boundary condition update [86]. For the generic Dirichlet boundary condition (4.39) this means that the new value of ϕ^+ is calculated via

$$(\phi^+)^{j+1} = (1-\alpha)(\phi^+)^j + \alpha \left(\mathcal{J}_\phi^j + (\phi^-)^j \right) \qquad (4.50)$$

when the boundary condition is updated. Here, $(\phi^+)^{j+1}$ is the set value of ϕ on the "+" side of the interface at the beginning of the (j+1)th DNA iteration. The relaxation parameter is denoted with α. In the generic case, it is not possible to derive a formulation for the relaxation parameter values that guarantee to stabilise the DNA or achieve ideal convergence rates. In the special case of species transport, however, it is possible and the exact derivation and formulation of the optimal relaxation parameters can be found in [86].

4.4.2 Interface residuals

Standard residual formulation

Before the interface residuals are formulated, the residuals, which occur during the iterative solution of an equation system, should be considered. Hence, if a system of linear algebraic equations

$$\boldsymbol{A}\boldsymbol{\phi} = \boldsymbol{b} \qquad (4.51)$$

is solved with an iterative method, an approximate solution $\boldsymbol{\phi}^m$ is obtained after m iterations, which does not exactly satisfy the system of equations. Due to the deviation from the exact solution a residual

$$\boldsymbol{r}^m = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{\phi}^m \qquad (4.52)$$

remains. The task of iterative solution procedures is therefore to minimise the residual throughout the iterations [22]. Furthermore, a convergence criteria for the iterative solution can be formulated based on this residual, since it is usually not necessary and possible to drive the solution to full convergence. In general, however, it is difficult to evaluate convergence directly using the residual from (4.52), since its value is strongly problem-dependent. Therefore, it is necessary to scale and normalise the residual in order to better estimate how much a residual has decreased during calculations [3]. In OpenFOAM this is done by defining the normalisation factor

$$n^m = \left\| \left(|\boldsymbol{A}\boldsymbol{\phi}^m - \boldsymbol{A}\overline{\boldsymbol{\phi}^m}| + |\boldsymbol{b} - \boldsymbol{A}\overline{\boldsymbol{\phi}^m}| \right) \right\|_1 , \qquad (4.53)$$

where $\overline{\phi^m}$ is a vector of the same size as ϕ^m but with the average value of ϕ^m in every component [59]. This normalisation factor is then used to scale and normalise the $\mathcal{L}^1$-norm of the residual:

$$\hat{r}^m = \frac{1}{n^m} \|\boldsymbol{r}^m\|_1 \ . \tag{4.54}$$

Normalisation and scaling in this manner is motivated by the fact that for uniform conditions $\phi^m = \overline{\phi^m}$ the residual $\hat{r}^m$ is equal to one [59]. Especially in simulations with uniform initial conditions, this makes judging the overall convergence more easily. A similar procedure using interface residuals will therefore be formulated in the following, which can be used to estimate the convergence of the DNA.

Generic interface residual formulation

As already explained at the beginning of section 4.4, the classical DNA starts with the solution of the problem on which the Dirichlet coupling condition is applied. Or in other words, it starts to solve the region where the generic jump condition across the interface is enforced. Assuming that the "+" side represents this region, the generic coupled Dirichlet boundary condition is set by

$$(\boldsymbol{\phi}^+)^j = \boldsymbol{\mathcal{J}}_\phi^{j-1} + (\boldsymbol{\phi}^-)^{j-1} \ . \tag{4.55}$$

Here, $\boldsymbol{\phi}$ and $\boldsymbol{\mathcal{J}}$ are the vectors whose components are the jump and the boundary value at each interface face centroid, respectively. After a solution for the first region is computed, the region on the other side of the interface is attended. In doing so, the interface coupled Neumann boundary condition is enforced on the "−" side of the interface. With the definition

$$[\![\Gamma_\phi \nabla \phi]\!] \cdot \boldsymbol{n}_\Sigma =: [\![\varphi]\!] \tag{4.56}$$

this can be written as

$$(\boldsymbol{\varphi}^-)^j = \boldsymbol{\mathcal{F}}_\phi^{j-1} - (\boldsymbol{\varphi}^+)^{j-1} \ . \tag{4.57}$$

Again $\boldsymbol{\varphi}$ and $\boldsymbol{\mathcal{F}}$ are vectors that represent the flux jump and the interface normal flux at each interface face centroid, respectively. After a solution for this region is computed, one DNA iteration is completed. At this point, the generic interface transmission condition is fulfilled, because the Neumann boundary condition was enforced last. However, this also means that the generic interface jump condition, which has been enforced at the beginning of the DNA, may no longer be met, since the absolute values of $\boldsymbol{\phi}^-$ at each interface face may have changed during the solution of the "−" region. It is therefore possible to formulate a residual of the generic interface jump condition after the j-th DNA iteration:

$$\boldsymbol{r}_{D,\phi}^j = \boldsymbol{\mathcal{J}}_\phi^j - [\![\boldsymbol{\phi}^j]\!] \ . \tag{4.58}$$

As the generic interface transmission condition is implicitly met, this residual gives a measure of how well the generic interface jump condition is satisfied. Additionally, when the Dirichlet boundary condition on the "+" side is updated at the beginning of the next DNA iteration, it can be shown that this residual is equal to the correction which is applied to the previously set boundary condition:

$$\begin{aligned} (\boldsymbol{\phi}^+)^{j+1} &= \boldsymbol{\mathcal{J}}_\phi^j + (\boldsymbol{\phi}^-)^j \\ &= (\boldsymbol{\phi}^+)^j + \boldsymbol{r}_D \ . \end{aligned} \tag{4.59}$$

However, before this residual can be used as an indicator for the convergence of the DNA, it must also be scaled and normalised. Following the example of the classical residual scaling and normalisation in OpenFOAM, a normalisation factor can be defined analogously to (4.53):

$$n_D^j = \left\| \left(|[\![\phi^j]\!] - [\![\overline{\phi^j}]\!]| - |\mathcal{J}_\phi^j - [\![\overline{\phi^j}]\!]| \right) \right\|_1 . \tag{4.60}$$

Here, $[\![\overline{\phi^j}]\!]$ is the average interfacial jump of ϕ. The $\mathcal{L}^1$ norm of $\boldsymbol{r}_{D,\phi}^j$ can subsequently be scaled and normalised using this factor:

$$\hat{r}_{D,\phi}^j = \frac{1}{n_D^j} \left\| \boldsymbol{r}_{D,\phi}^j \right\|_1 . \tag{4.61}$$

If the NDA is now considered instead of the DNA, a different situation arises. In this case, the region, on which the interface coupled Neumann boundary condition is applied, is solved first. Followed by the solution of the region on which the interface coupled Dirichlet boundary condition is enforced. Hence, the generic jump condition is implicitly met at the end of one NDA iteration. The same does not apply for the generic transmission condition and a residual similar to (4.58) can be formulated:

$$\boldsymbol{r}_{N,\phi}^j = \boldsymbol{\mathcal{F}}_\phi^j - [\![\boldsymbol{\varphi}^j]\!] . \tag{4.62}$$

This residual gives a measure of how well the generic interface transmission condition is met after the j-th NDA iteration. It can also be shown that it represents the correction that is applied to the interface coupled Neumann boundary condition at the beginning of the next NDA iteration.

$$\begin{aligned} (\boldsymbol{\varphi}^+)^{j+1} &= \boldsymbol{\mathcal{F}}_\phi^j + (\boldsymbol{\varphi}^-)^j \\ &= (\boldsymbol{\varphi}^+)^j + \boldsymbol{r}_{N,\phi}^j \end{aligned} \tag{4.63}$$

Again, $\boldsymbol{r}_{N,\phi}^j$ needs to be scaled and normalised to use it as an indicator for convergence of the NDA. For this purpose a normalisation factor

$$n_N^j = \left\| \left(|[\![\boldsymbol{\varphi}^j]\!] - [\![\overline{\boldsymbol{\varphi}^j}]\!]| - |\boldsymbol{\mathcal{F}}_\phi^j - [\![\overline{\boldsymbol{\varphi}^j}]\!]| \right) \right\|_1 \tag{4.64}$$

analogous to (4.53) and (4.60) is formulated which can subsequently be used to scale and normalise the $\mathcal{L}^1$-norm of the residual:

$$\hat{r}_{N,\phi}^j = \frac{1}{n_N^j} \left\| \boldsymbol{r}_{N,\phi}^j \right\|_1 . \tag{4.65}$$

In equation (4.64) $[\![\overline{\boldsymbol{\varphi}^j}]\!]$ is the average jump of the interface normal flux of ϕ after the j-th NDA iteration.

In summary, two different residuals could be derived which reflect the convergence of the segregated interface coupling algorithm. When used as a measure of convergence, two different cases have to be considered. In the first case, the classical DNA is used which starts each iteration by enforcing the generic interface jump condition and ends with enforcing the generic interface transmission condition. Therefore the residuals for each boundary condition at the end of each iteration are:

$$\begin{aligned} \hat{r}_{D,\phi}^j &\neq 0 , \\ \hat{r}_{N,\phi}^j &= 0 . \end{aligned} \tag{4.66}$$

In the second case, the order in which the coupled boundary conditions are enforced in each iteration is changed, implying that the NDA is used. This leads to the fact that the interface residual that is equal to zero and the one that is not also changed:

$$\begin{aligned} \hat{r}^j_{D,\phi} &= 0 \ , \\ \hat{r}^j_{N,\phi} &\neq 0 \ . \end{aligned} \tag{4.67}$$

For this reason it is important to consider which iteration order is used when judging the convergence based on these residuals. This is especially important when considering the coupled Navier-Stokes-Equations, as will be explained in the following. In general, however, it is a good assumption to consider the interface coupling algorithm as converged when the scaled and normalised residual, which is not zero, has fallen by at least three orders of magnitude.

Interface residuals for the Navier-Stokes-Equations

When the interface coupling of the Navier-Stokes equations is considered, particularities arise regarding the convergence evaluation due to the pressure-velocity coupling. To obtain a stable interface coupling algorithm, different boundary condition types for pressure and velocity must be applied on the same side of the interface. In particular, the pressure Dirichlet (4.42) and velocity Neumann (4.45) boundary conditions must be employed on the interface side of the denser fluid and the pressure Neumann (4.47) and the velocity Dirichlet boundary condition (4.41) must be employed on the other side [81]. Hence, if an interface coupling algorithm is performed that starts at the denser fluid region and then moves to the lighter fluid region, a DNA for the pressure and an NDA for the velocity is obtained. The residuals by which the convergence of the coupling iteration can be judged are thus $\hat{r}_{D,p}$ and $\hat{r}_{N,\boldsymbol{u}}$. Following the derivations of the coupled pressure Dirichlet boundary condition (4.42), one finds that $\hat{r}_{D,p}$ represents how well the normal component of the momentum transmission condition (3.30) is met. Doing the same for the coupled velocity Neumann boundary condition (4.45), one finds that $\hat{r}_{N,\boldsymbol{u}}$ represents how well the tangential component of the momentum transmission condition is met. Thus, both residuals together give an estimate of how accurately the momentum transmission condition is satisfied, while the mass transfer together with the no slip condition are always implicitly met at the end of one coupling iteration.

Reversing the iteration order and starting with the lighter fluid region in the interface coupling iteration, changes the residuals that need to be considered for the convergence evaluation. In this case, a NDA for the pressure and a DNA for the velocity is performed, which leads to $\hat{r}_{N,p}$ and $\hat{r}_{D,\boldsymbol{u}}$ being the residuals of interest. Here, the momentum transmission condition is enforced at the end of one coupling iteration. Following the derivation of the interface coupled velocity Dirichlet boundary condition (4.41) leads to the fact that $\hat{r}_{D,\boldsymbol{u}}$ gives a measure of how well the mass transmission condition together with the no slip condition is satisfied. Doing the same for the interface coupled pressure Neumann boundary condition (4.47) shows that $\hat{r}_{N,p}$ is a measure of how well the momentum jump in interface normal direction (4.46) is met.

An evaluation of the residuals presented in this subsection is performed in chapter 6.

4.4.3 Handling of pressure velocity coupled systems

As already mentioned in section 4.3.1, the pressure-velocity coupling of the Navier-Stokes equations requires special treatment with respect to the volumetric coupling. However, this must also be taken into account for the interface coupling, as there are different possibilities on how the interface coupling (here DNA) can be combined with the chosen volumetric coupling method.

If the Navier-Stokes equations are volumetrically coupled in a fully-implicit manner (see section 4.3.1), the interface coupling approach using the DNA is straightforward. This means that the procedure can be used as described at the beginning of section 4.4. Therefore, at the start of one DNA iteration, the coupling boundary conditions for pressure and velocity are updated simultaneously in the first region. Then the fully-implicit coupled pressure-velocity system of the first region is solved. Subsequently, the coupling boundary conditions for pressure and velocity of the second region are again updated simultaneously and the fully-implicit coupled pressure-velocity system of the second region is solved. The same procedure can also be used when the volumetric coupling of the Navier-Stokes equations is done in a semi-implicit manner. In this case, the solution of the fully-implicit coupled pressure-velocity system is replaced by the semi-implicit predictor corrector procedure described in 4.3.1. However, using the semi-implicit volumetric coupling procedure, another combination with the DNA is applicable. Here, the momentum predictor and pressure solution/velocity corrector step are split up inside one DNA iteration. This means that a DNA iteration starts by first updating the coupled velocity boundary condition of the first region and solving the momentum predictor equation. Then the coupled velocity boundary condition of the second region is updated and the momentum predictor equation is solved subsequently. Afterwards, the coupled pressure boundary condition of the first region is updated, followed by the pressure solution/velocity corrector loop inside this region. Finally, the pressure boundary condition of the second region is updated and the pressure solution/velocity corrector loop is executed before a new DNA iteration step is carried out.

It is not obvious which of the described methods using semi-implicit volumetric coupling is better suited for simulating a rising bubble. However, due to the limited scope of this work, only the latter approach is used in the test applications presented in 6.

4.5 Interface tracking method

Up to this point, it has already been clarified how the individual conservation equations are solved. In this context, both the volumetric and the interface coupling have been explained. Furthermore, it was already taken into account that the computational mesh is able to move, which is why all considered equations were transformed into the ALE formulation. However, it was not yet considered how the interface is moved, which in turn leads to a deformation/movement of the entire computational mesh. This will therefore be explained in this section.

In the derivation of the mass and momentum transport in section 3.2 it was assumed that the interface is a material boundary and therefore no mass flow takes place across it. Translating this into a discretised setup where the interface is composed of a set of finite area faces, the zero mass flux condition across the interface can be written as

$$\rho^{+}(\boldsymbol{u}^{+}_{Fc} - \boldsymbol{w}_{Fc}) \cdot \boldsymbol{S}_{Fc} = \dot{m}_{Fc} - \rho^{+}\dot{V}_{Fc} \overset{!}{=} 0 \; . \tag{4.68}$$

Here, the subscript Fc denotes the centre of a face that is part of the whole Interface. The same expression can also be set up analogously for the ”−” side (see equation (3.28)). As one can easily deduce from this expression, the motion of the interface is strongly coupled to the surrounding velocity fields. Vice versa, it is known that the domain geometry strongly influences the flow field. However, before clarifying how this coupling can be resolved numerically, it is necessary to consider how the interface is moved in the first place. For this purpose, it is considered that the described conservation equations have been solved for a new time

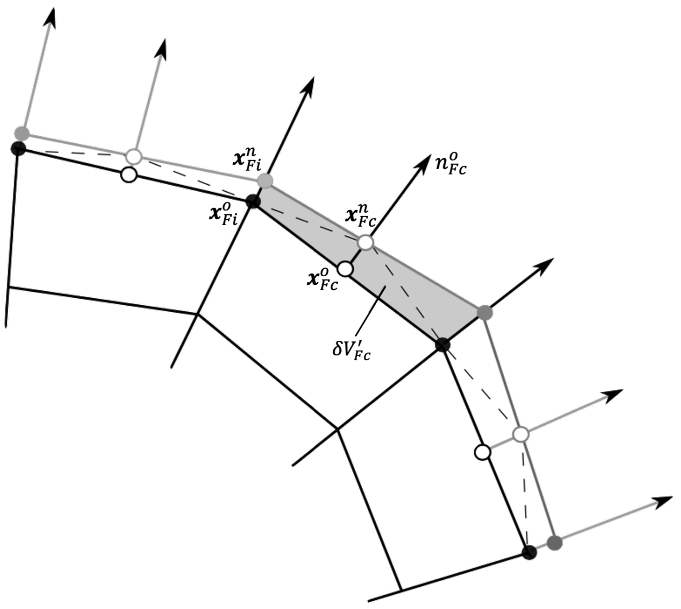


Figure 4.3: Interface movement using control point displacement along the face normals [81, 49]

step on a spatially fixed mesh. Since the mesh has not been moved, expression (4.68) is unlikely to be met. In this case a residual mass flux across the interface occurs and a volume flux correction $\dot{V}'_{Fc}$ can be formulated [81]:

$$\frac{\dot{m}_{Fc}}{\rho^+} - \dot{V}_{Fc} =: \dot{V}'_{Fc} \neq 0 \; . \tag{4.69}$$

With this volume flux correction the volume $\delta V'_{Fc}$, that need to be swept by face F in order to cancel out the the residual mass flux, can be calculated from (4.18):

$$\delta V'_{Fc} = \frac{2}{3}\dot{V}'_{Fc}\Delta t + \frac{1}{3}\delta V'^{o}_{Fc} \; . \tag{4.70}$$

Based on this swept volume the displacement of the face F can be computed while both cancelling out the residual mass flux and obeying the space conservation law. For the calculation of the actual point displacement a procedure proposed by Muzaferia and Perić [55] is used. In this procedure the face centres are used as control points for which the displacement is computed by

$$h' = \frac{\delta V'_{Fc}}{\boldsymbol{S}^o_{Fc} \cdot \boldsymbol{n}^o_{Fc}} \; . \tag{4.71}$$

Subsequently, the new position of the control point can be calculated with

$$\boldsymbol{x}^n_{Fc} = \boldsymbol{x}^o_{Fc} + h'\boldsymbol{n}^o_{Fc} \; , \tag{4.72}$$

where $\boldsymbol{x}^o_{Fc}$ is the position vector of the control point before and $\boldsymbol{x}^n_{Fc}$ is the position vector after the correction. At this point it should be noted that the position correction of the control points does not necessarily have to be performed in the normal direction of the face. Instead an arbitrary correction direction can be used for $\boldsymbol{n}^o_{Fc}$ in (4.71) and (4.72) [81]. After the new control point position is specified, the interface mesh nodes denoted by Fi need to be moved in order to actually move the face F. This can be done by moving the mesh nodes up to a plane that is anchored at the corresponding control point and with its orientation computed using the least squares method. For more details on this computation see [81]. In this way, the movement of the interface is sufficiently described to compensate for the residual interface mass flow and therefore represent the movement of a material boundary. However, if only the mesh vertices at the interface were moved, this would lead to strongly deformed cells next to the interface, which could potentially also have negative volumes. Therefore, the whole mesh has to account for the interface movement like a solid body under the influence of deformations. In analogy to the strain and deformation calculation in solids, the deformation vectors of the individual mesh vertices can be calculated by solving a Laplace equation. For a detailed description on the solution procedure of such a Laplace equation for the mesh deformation see [37].

Now that the interface movement and the consequent grid movement have been explained, the previously mentioned coupling between interface movement and flow field has to be addressed. This coupling can numerically be resolved by applying an iterative procedure. In such a procedure, the conservation equations are solved first at the beginning of a new time step on a spatially fixed mesh; including volumetric and interface coupling. This is then followed by an interface/mesh movement as described above. Subsequently, the governing conservation equations are again solved on the deformed mesh and the interface/mesh is again moved afterwards. In this way, iteration is continued until the changes in the flow field and with respect to the interface position become small [22, 56]. Then the solution is advanced to the next time step.

4.6 Moving reference frame

Since moving reference frames are used in the rising bubble flows shown in chapter 6, a very brief explanation of how such moving reference frames are mathematically implemented will follow in this section.

The governing equations derived in chapter 3 have been formulated for inertial reference frames. For some applications, however, it is practical to observe the simulated flows from a non-inertial reference frame. Such cases can be moving objects through a fluid where it is beneficial to move with the objects in order to keep the computational mesh as small as possible, or turbomachineries where a rotational reference frame makes mesh movement obsolete. For such purposes a non-inertial reference frame has to be defined which is translating with velocity $\boldsymbol{v}(t)$ and rotating with angular velocity $\boldsymbol{\omega}(t)$ with respect to the stationary inertial reference frame. Transforming the transport equations for mass (3.24), momentum (3.25) and species (3.34) from the inertial to the non-inertial reference frame leads to the following: The form of the transport equations for mass and species stays unchanged with the only difference that $\boldsymbol{u}$ now represents the relative velocity with respect to the moving reference frame. The momentum transport equation in the moving reference frame, however, reads

$$\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u} \cdot \nabla)\, \boldsymbol{u} = -\frac{1}{\rho} \nabla \cdot p + \nu \Delta \boldsymbol{u} + \boldsymbol{g} - \left[2\boldsymbol{\omega} \times \boldsymbol{u} + \boldsymbol{\omega} \times (\boldsymbol{\omega} \times \boldsymbol{x}) + \frac{\mathrm{d}\boldsymbol{v}}{\mathrm{d}t} + \frac{\mathrm{d}\boldsymbol{\omega}}{\mathrm{d}t} \times \boldsymbol{x} \right] . \tag{4.73}$$

Here, $\boldsymbol{u}$ again represents the relative velocity. Furthermore, an additional source term, enclosed in the square parentheses, arises. This source term contains additional pseudo-forces that appear due to the observation of the flow from a non-inertial reference frame. Namely Coriolis forces (first term), centrifugal forces (second term), forces due to linear acceleration (third term) and angular acceleration (fourth term) [54, 69]. Numerically these terms are treated explicitly and are therefore added to the right-hand side of the discretised momentum equation.

5 Implementation

As Weller et al. already mentioned in their first paper on the precursor of OpenFOAM [89], a lot is published about the development of new and efficient algorithms for CFD, but little about the general code design. Since this is still the case 24 years later, this chapter will explicitly address the implementation and code design of the new unified multiphysics framework for multi-region coupling called multiRegionFOAM. In doing so, section 5.1 discusses the general code structure of multiRegionFOAM. In addition, it is explicitly explained how the parallelization of the computations is realized. In sections 5.2 and 5.3, special attention is paid to the implementation of the generic interface coupled boundary conditions and the Dirichlet-Neumann algorithm solution control based on them. The basic implementation of the interface tracking functionalities and their integration into multiRegionFOAM is discussed in section 5.4.

5.1 multiRegionFOAM

> ”The intention is to develop a C++ class library that makes it possible to implement complicated mathematical and physical models as high-level mathematical expressions.”

This was and is the aspiration formulated by Weller et al. [89] that led to the development of OpenFOAM and continues to drive its further development. However, with respect to multiphysics simulations, which are inherently based on highly ”complicated mathematical and physical models”, OpenFOAM provides only single very problem-specific approaches so far. This gap shall be closed by the development of multiRegionFOAM, a unified multiphysics framework for multi-region coupling. In doing so, the generic structure of interface coupled multiphysics problems, which is explained in the previous chapters, is exploited. As multiRegionFOAM is based on OpenFOAM, the Object-Oriented Programming (OOP) methodology is used for its development. Therefore, the framework is build from classes that represent conceptual objects in the code [78], which will be explained in the following. Before doing so, it is important to note that multiRegionFOAM is still in an early stage of development and is therefore subject to ongoing improvement.

5.1.1 Structure

With the underlying generic sharp interface model, the separation of the conceptual functionalities and the special physical properties of an object was set as a design directive. Hence, the specialisation of different base classes is realised by the extensive use of inheritance and templated code. The basic structure of the multiRegionFOAM framework is shown as an UML class diagram in figure 5.1.

The core of the framework is the `multiRegionSystem` class. It represents the complete system that shall be simulated and can consist of an arbitrary number of regions, each of which can have different physical properties and are coupled to each other through an arbitrary number of interfaces. Therefore, the class is composed of a list of regions (`regionTypeList`) and a list of interfaces (`regionInterfaceList`). Since `multiRegionSystem` is the top level of the framework, it is also responsible for organising the solution of the entire system. This also includes the interface coupling, which is possible both in a monolithic and a partitioned manner. For the partitioned solution in particular, `multiRegionSystem` utilises the DNA solution control, which is discussed separately in section 5.3.

As mentioned above, `multiRegionSystem` consists of several regions. These represent the different bulks of the solution domain. Hence, they occupy a certain volume of the whole computational domain and may have different physical properties, which in turn are defined in terms of the conservation equations and constitutive assumptions that describe the behaviour of each individual region. Therefore, a base class called `regionType` exists that holds and provides the basic data every type of region is based on (see figures 5.1 and 5.2). On the one hand, such data is the computational mesh, which forms the foundation of any region, defines its spatial extent and discretization. Since, it is clear that a region might deform, move, change in size or change its spatial discretization, the `regionType` class holds a `dynamicFvMesh`, which potentially provides all these functionalities via specialised mesh types. On the other hand, a region constructs, stores and provides the discretised system of conservation equations it is governed by. Therefore, `regionType` holds hashed pointer tables that store pointers to the linear equation systems (`fvMatrix`), which can be accessed by a previously defined equation name. In this context, `regionType` also provides the foundations for different volumetric coupling approaches of the conservation equations, like semi-implicit or fully implicit coupling. Additionally, it provides the functionalities for the correction of the material properties during the overall solution process. Based on `regionType` it is then possible to derive specialised types that construct the specialised fields on the mesh and construct the linear equation systems of the specialised bulk equations. An example of such a specialised `regionType` is an incompressible fluid region type called `icoFluid` that handles the volumetric pressure-velocity coupling in a semi-implicit manner (see figure 5.1). At this point it is important to mention that multiRegionFOAM also allows to superimpose multiple specialised region types. This superposition is made possible by the fact that different region types can hold the same mesh and thus also have access to all fields generated by the others.

As mentioned earlier, all regions of which the `multiRegionSystem` consists, are organised in `regionTypeList` (see figure 5.1). This class is a pointer list and therefore holds the computational pointers to each region. Furthermore, it instantiates these regions in the first place by utilising the data of the `regionProperties` class, which it is also composed of. The `regionProperties` class in turn reads and stores the information about which regions exist, what they are called and of which type/superimposed types they are.

An interface, which is coupling two regions, is described by the class `regionInterface`. In principle, this class is based on two fully overlapping boundary patches (`fvPatch`) of two adjacent region meshes. The union of these two patches form the interface. Therefore, `regionInterface` holds references to the two patches and is derived from a class called `interfaceKey`, that itself just represents the name pair of the two patches (see figure 5.3). Thereby the two patches represent the respective sides of the interface on which the interface coupling boundary conditions can then be applied. Since the coupling boundary conditions rely on the field values from the other side of the interface, `regionInterface` provides functionalities that enable the boundary condition on one interface patch to access the field values from the other. This is achieved by the fact that `regionInterface` is also composed of the `interfaceToInterfaceMapping` class, which performs the task of mapping the fields from one interface patch to the other. Since the adjacent regions may be discretised in a different way, the two boundary patches forming the interface may also be discretised differently. Therefore, various specialisations of the `interfaceToInterfaceMapping` class exist that use different mapping methods. Such methods are the direct mapping method utilised in `directMapInterfaceToInterfaceMapping` for conformal interface patches and the general grid interface mapping method [5] utilised in `ggiInterfaceToInterfaceMapping` for non-conformal interface patches.

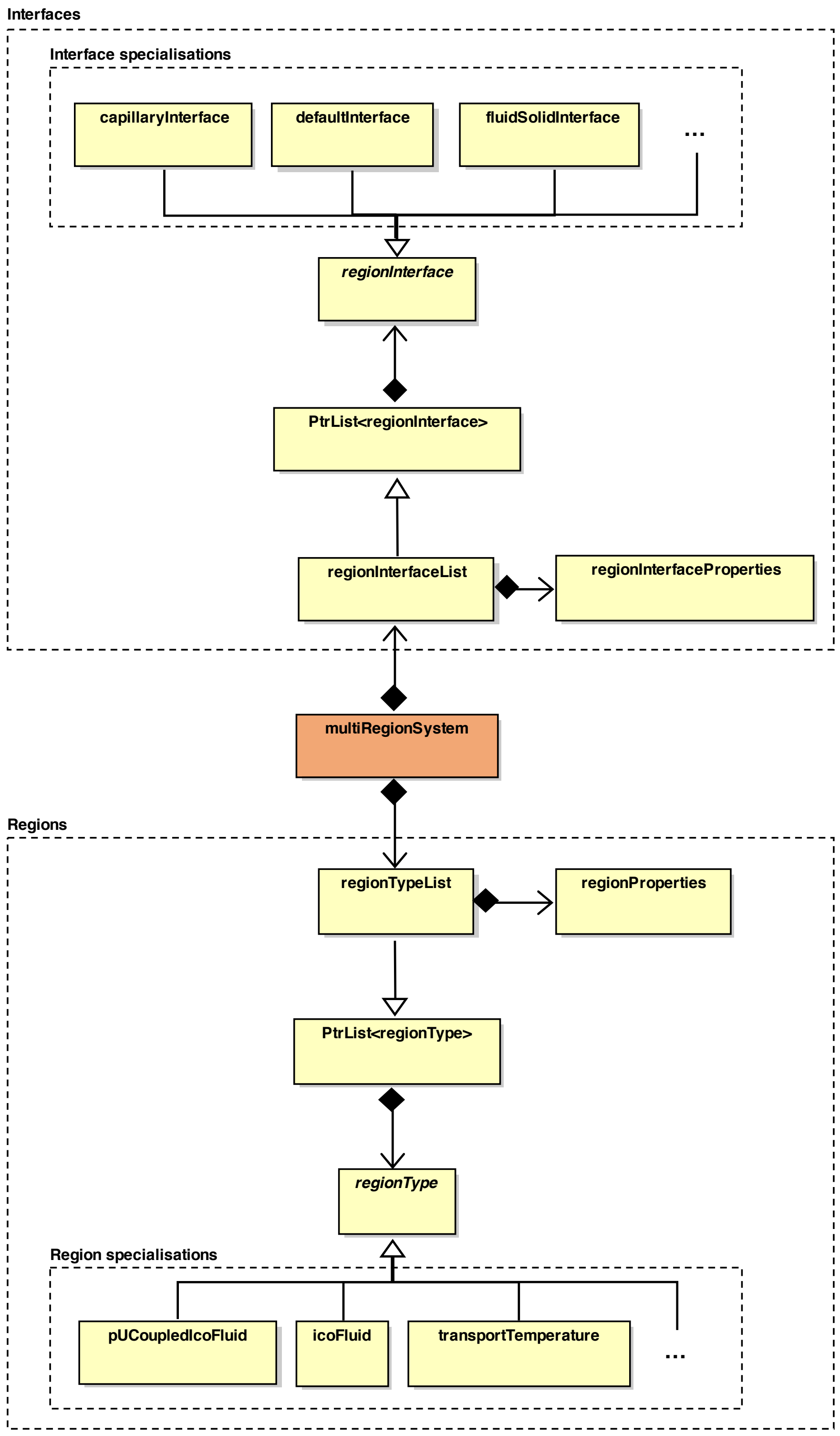


Figure 5.1: General structure of the multiRegionFOAM framework - UML class diagram

The role of `globalPolyPatch`, which is also part of `regionInterface`, is explained in section 5.1.2. To also account for transport processes along the interface, `regionInterface` creates and holds a finite area mesh

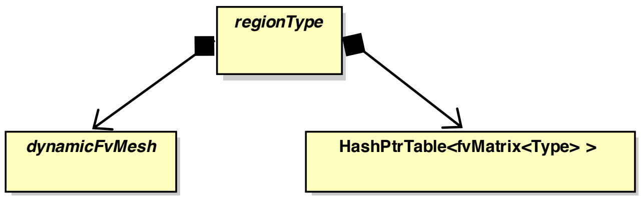


Figure 5.2: General structure of `regionType` - UML class diagram

(`faMesh`), which is generated based on one of the interface patches (see figure 5.3). As `regionInterface` only represents a generic interface it does not implement any specific physics. It only provides geometrical information based on the finite area mesh, like the interface curvature, in addition to the previously explained data and functionalities. Specific interface physics are implemented in specialisations of `regionInterface`. An example of such a specialised interface is the `capilaryInterface` (see figure 5.1). This type of interface represents an interface between two fluids at which a certain amount of surface tension is present.

Similar to all the regions of the `multiRegionSystem`, the interfaces are organised in `regionInterfaceList` (see figure 5.1). This class is again a pointer list and therefore holds the computational pointers to each region. It also instantiates the interfaces in the first place by utilising the data of the `regionInterfaceProperties` class. Here, the `regionInterfaceProperties` class reads and stores the information about which interfaces exist, which regions each one couples, which two boundary patches they are based on, which fields are coupled across each interface and if the interface coupling for the fields should be performed in a partitioned or monolithic manner.

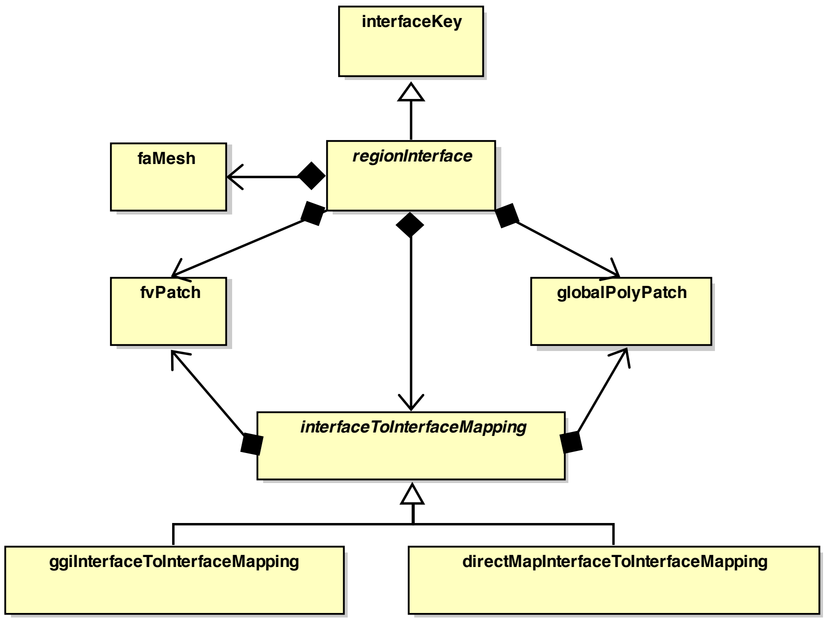

interfaceKey
faMesh
regionInterface
fvPatch
globalPolyPatch
interfaceToInterfaceMapping
ggiInterfaceToInterfaceMapping
directMapInterfaceToInterfaceMapping

Figure 5.3: General structure of `regionInterface` - UML class diagram

5.1.2 Parallelisation

In general, parallelisation in OpenFOAM is achieved by utilising the domain decomposition method, where the computational mesh is partitioned into sub-domains. Each of these subdomains is then solved on a separate

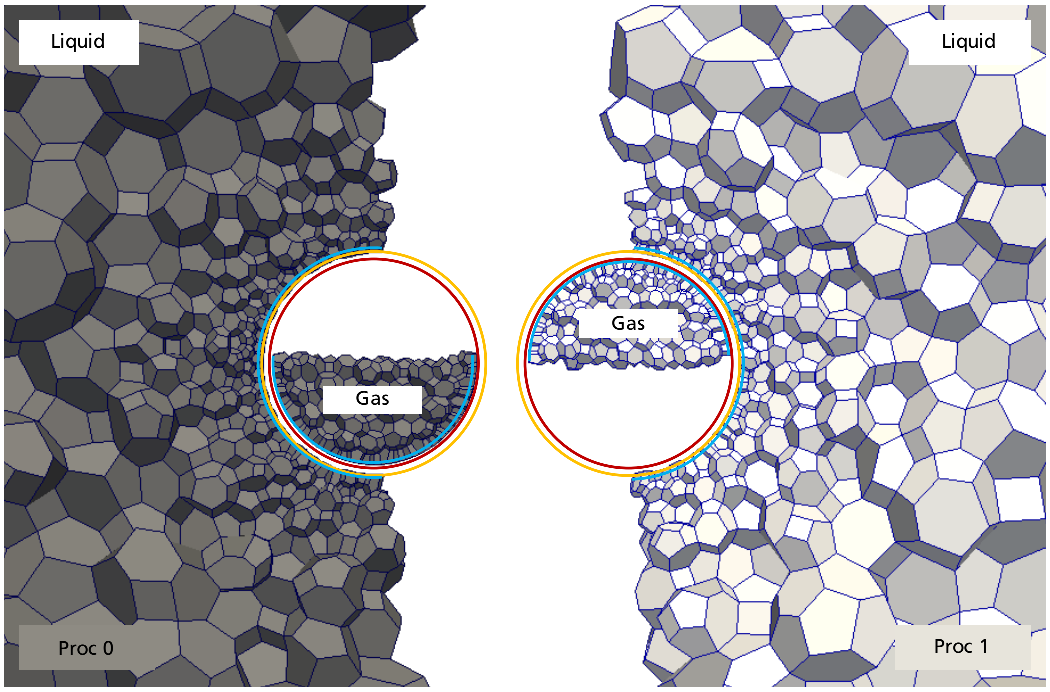
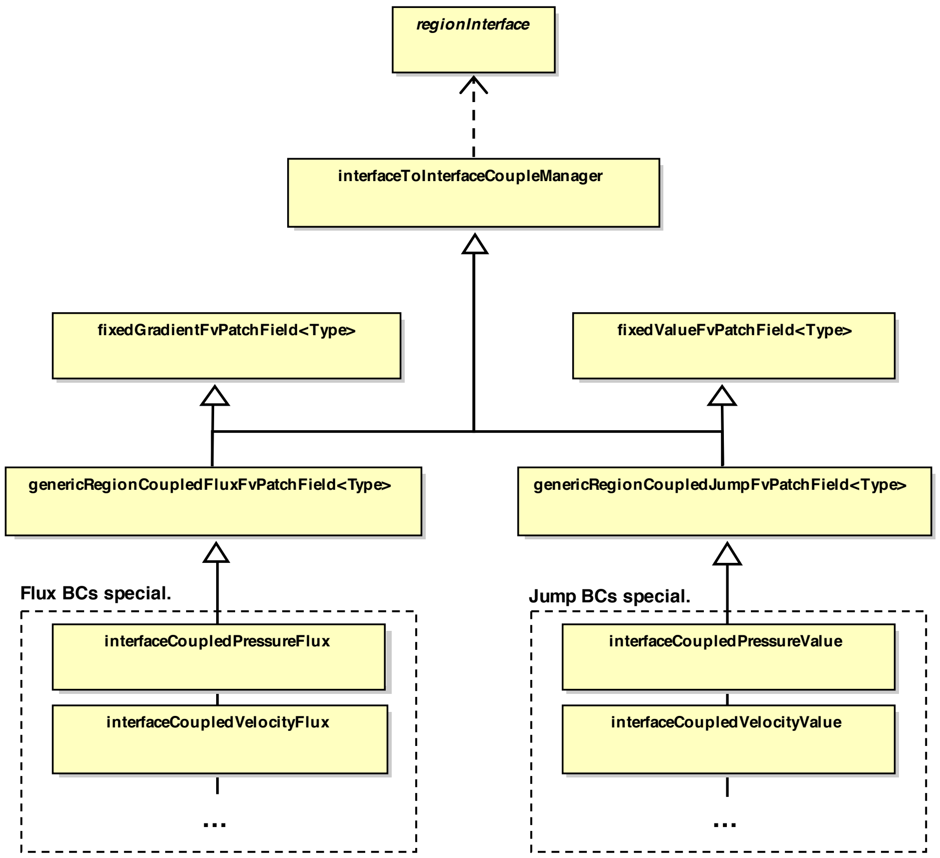


Figure 5.4: Local (blue) and global (red, orange) poly patches of a gas and a liquid region differently decomposed into two sub-domains

CPU-core, by running a separate instance of the solver application on each processor. Communication between the CPU-cores, performed using the Message Passing Interface communication protocol [25], provides the necessary coupling information that is needed to solve the underlying system of algebraic equations [11]. In the context of multiRegionFOAM, the domain already consists of different sub-domains, each of which is an individual region. However, this inherent partitioning of the entire domain is not well suited for a useful parallelisation of the computation. On the one hand, the grids of the individual regions can differ greatly in size, which would lead to poor balancing of the computational load. On the other hand, the use of partitioned interface coupling would mean that only one region is solved at the time. This would lead to the fact that only one processor would work at the time while the others would have to wait, which of course undermines the whole point of parallel computations. The approach that multiRegionFOAM follows to achieve parallelisation is therefore to decompose all regions into the same number of sub-domains. In this way each processor performs the calculations for a partition of every region. Although the regions must be decomposed into the same number of sub-domains, the decomposition method may differ between all regions. For example, figure 5.4 shows two interface coupled regions. One region being a gas bubble and the other region being the surrounding liquid. Both regions are decomposed into two sub-domains, while the liquid region is decomposed vertically and the gas region is decomposed horizontally. However, this leads to the fact that the two interface patches, which together should form a part of the interface and to which a processor has access, may no longer completely overlap. This is indicated by the blue lines in figure 5.4. Considering the previously explained mapping of the patch fields from one interface patch to the other, this would no longer be possible. Therefore, multiRegionFOAM, or the `regionInterface` class in particular, uses the *global face zone concept* originally introduced by Cardiff et al. [11] and Tuković et al. [82]. Following this concept, each processor has access to the entire interface patches. This is achieved by the fact that `regionInterface` is not only composed of a pair of the local interface patches but also holds a pair of global patches (see figure 5.3). These global patches, each represented by the `globalPolyPatch` class, are the reconstructions of the partitioned patches and are illustrated by the orange and red lines in figure 5.4. With the help of these global patches it is possible to reconstruct the fields on both sides of the interface using a gather-scatter procedure and subsequently map these fields from one global patch to the other. Thus, each processor has access to all data needed for the application of the coupled boundary conditions and the possibility to independently decompose each region is ensured.

5.2 Generic interface coupled boundary conditions



regionInterface
interfaceToInterfaceCoupleManager
fixedGradientFvPatchField<Type>
fixedValueFvPatchField<Type>
genericRegionCoupledFluxFvPatchField<Type>
genericRegionCoupledJumpFvPatchField<Type>
Flux BCs special.
interfaceCoupledPressureFlux
interfaceCoupledVelocityFlux
...
Jump BCs special.
interfaceCoupledPressureValue
interfaceCoupledVelocityValue
...

Figure 5.5: General structure of the generic coupled boundary condition classes `genericRegionCoupled-ValueFvPatchField` and `genericRegionCoupledFluxFvPatchField` - UML class diagram

In the case of partitioned interface coupling, explained in section 4.4, two different coupled boundary conditions need to be applied to the interface for each interface coupled field; one on each side of the interface. Since each side of the interface is represented by a boundary patch in `regionInterface`, a coupled boundary condition must be applied to each of these patches. As shown in section 4.4.1, these boundary conditions are either of the Dirichlet or the Neumann type and can be derived from a generic formulation. Exploiting this generic form, two base classes for the coupled boundary conditions are implemented in the multiRegionFOAM framework. Here, `genericRegionCoupledValueFvPatchField` represents the Dirichlet type (4.39) and `genericRegionCoupledFluxFvPatchField` the Neumann type (4.40) of the generic coupled boundary conditions. Therefore, they are derived from the `fixedValueFvPatchField` and the `fixedGradientFvPatchField` class, which are the base classes for Dirichlet and Neumann boundary conditions, respectively. Furthermore, the two generic coupled boundary condition classes inherit from `interfaceToInterfaceCoupleManager`. This class contains all functionalities needed to manage the mapping of fields from the other side of the interface to the current side. It can access the `regionInterface` on which the coupled boundary condition is applied. Based on the generic coupled boundary condition classes it is then possible to derive specialisations, by specifying the generic value jump $\mathcal{J}_\phi$ or the generic flux jump $\mathcal{F}_\phi$ and the diffusion coefficient Γ_ϕ. Such specialisations are, for example, `interfaceCoupledPressureValue` and `interfaceCoupledPressureFlux`, which represent the interface coupled boundary conditions for the pressure (4.42) and (4.47), respectively. Since the specialised boundary conditions can be associated to fields of multiple tensor ranks, `genericRegionCoupledValueFvPatchField` and `genericRegionCoupledFluxFvPatchField` are implemented type agnostic through the use of templatisation. Furthermore, these two classes provide the functionalities for the computation of interface residuals as explained in section 4.4.2, which enables the dynamic control of the DNA.

5.3 Dirichlet-Neumann-Algorithm solution control

`multiRegionSystem`, being responsible for managing the solution of the whole domain, implements, among other coupling options, a DNA coupling loop in which interface coupled fields are solved according to the DNA procedure (see section 4.4). In order to dynamically control the number of coupling iterations, `multiRegionSystem` depends on the `dnaControl` class, as displayed in figure 5.6. `dnaControl`, like `multiRegionSystem`, has access to `regionInterfaceList`, allowing it to access all coupled boundary conditions on all interfaces. This in turn leads to the fact that all interface residuals, provided by the coupled boundary condition classes, can also be accessed. By checking these residuals on all interfaces against maximum values previously specified by the user, `dnaControl` can terminate the coupling loop inside of `multiRegionSystem` if the termination criteria are satisfied. Additionally, `dnaControl` allows the output of the unscaled and unnormalised residual fields on the interfaces, which allows a more accurate evaluation of the interface coupling during post-processing.

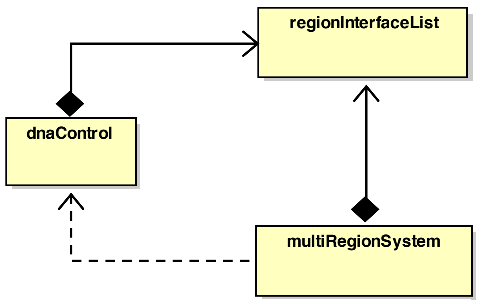


Figure 5.6: General structure and utilisation of `dnaControl` - UML class diagram

5.4 Interface tracking mesh

The implementation of the interface tracking method explained in section 4.5 is essentially done in two classes. These are the `interfaceTrackingFvMesh` and the `movingInterfacePatches` class, as displayed in figure 5.7. The `interfaceTrackingFvMesh` class, represents an entire moving computational mesh containing at least one moving boundary that is part of an interface. Since it is a specialisation of the `dynamicFvMesh` class, it can be specified as the mesh of a region (see figure 5.2). It is important to note that two regions, which are connected via a moving interface, must each hold an `interfaceTrackingFvMesh`, as they both need to move in accordance to the interface. To ensure the coherent movement of such adjacent meshes, one side of the interface is considered as the actively and the other as the passively moving side. For each moving interface in a mutliregion system the `interfaceTrackingFvMesh` on the actively moving side has access to an instance of `movingInterfacePatches`. The purpose of this class is to calculate the movement of the considered interface on the basis of the control point algorithm explained in section 4.5. In doing so, the deformation of the interface patch on the active side is calculated first without actually moving it. Subsequently, this deformation is mapped to the interface patch on the passive side. These deformations are then used to update the boundary conditions for the mesh motion of the `interfaceTrackingFvMesh` on both the passive and the active side. Finally the mesh motion equations for all `interfaceTrackingFvMeshes` are solved through the utilisation of the `meshSolver` class and the meshes are actually moved.

In order to enable the mapping of the interface deformation from the actively to the passively moved side, a similar approach as for the mapping utilised by the interface coupled boundary conditions is used. `movingInterfacePatches` has access to the two `globalPolyPatches`, that represent the entire moving interface, and utilises the mapping functionalities of the `interfaceToInterfaceMapping` class to consistently map the deformation values from one interface side to the other.

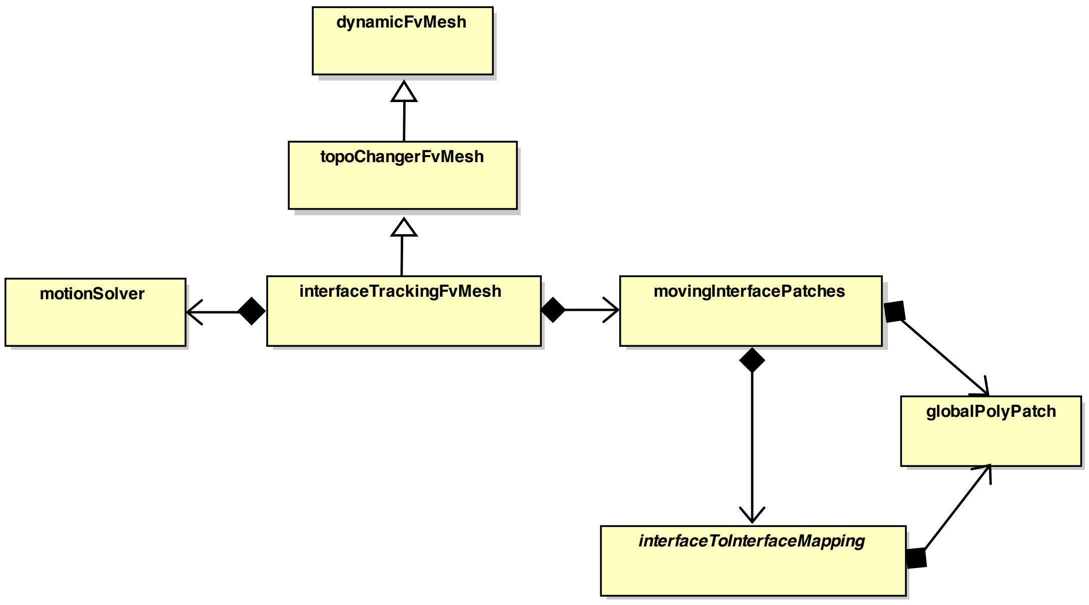


Figure 5.7: General structure of the interface tracking implementation - UML class diagram

6 Bubbly Flow Test Applications

In order to test the capabilities of the newly developed multiphysics framework with respect to the simulation of reactive bubbly flows, multiRegionFOAM is applied to two example cases. Both cases consider a rising air bubble in water. In the first case, both the gas and liquid phases are inert. In the second case, a generic dilute species A transfers from the gas into the liquid phase and subsequently reacts there with another generic dilute species B.

Since both cases only consider a single rising bubble, the underlying numerical setup is the same in each case. The liquid phase and the gas phase each represent a separate region. The two regions are coupled via a single interface that consists of two mesh boundary patches, named ”interface” at the liquid and ”interfaceShadow” at the gas side. The ALE-IT, as described in chapter 4.5, is used to account for the movement of this interface. To reduce the number of cells needed for these simulations, a moving reference frame (see section 4.6) is used to fix the bubble centre. Therefore, the gas bubble centre stays in the same position at all times and the liquid phase is flowing around it. This leads to a configuration of the computational domain that is shown in Figure 6.1. Both cases assume a bubble with an initial spherical shape with radius r_B. The fluid domain surrounding the bubble is also shaped as a sphere with a radius of $20r_B$.

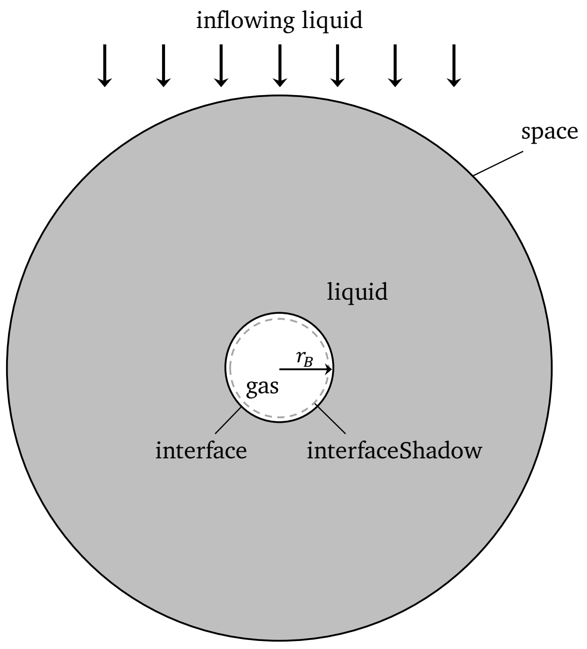
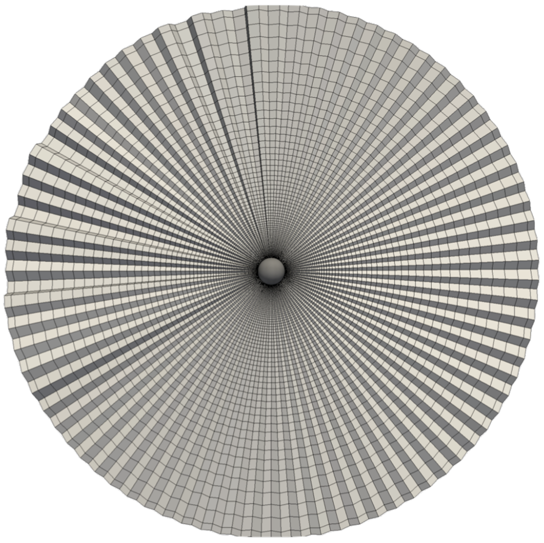


Figure 6.1: Sketch of the computational domain used for the simulation of single rising bubbles [76]

As the gaseous and the liquid phase are two different regions, both are represented by their own computational mesh. The meshes consist of polyhedral cells for the gas bubble and prismatic cells with a polyhedral base for the liquid, as shown in figure 6.2. These meshes are based on the results of a previously performed study by Steinhausen [76].


(a) Cross-section of the full domain

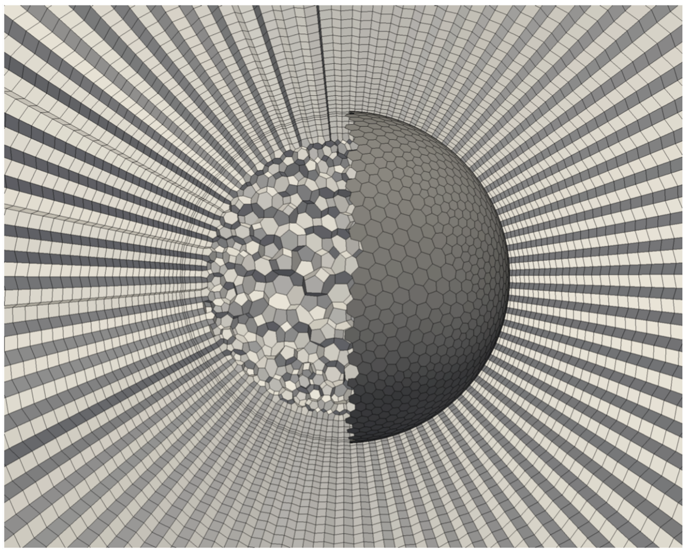

(b) Enlarged view of the bubble, its close proximity and a part of the interface patch

Figure 6.2: Meshes of the gas bubble and the surrounding liquid

For the temporal discretisation, the simulated physical time is split up into fixed time steps Δt and a backward-differencing scheme is used for the time integration. The size of the time step is chosen according to a numerical stability criterion for the interface [81]

$$\Delta t < \sqrt{\frac{\rho_L \min\left(l_{\text{PN}}\right)^3}{2\pi\sigma}}\,, \tag{6.1}$$

where l_{PN} is the geodetic distance between two face centres on the interface patches [76]. ρ_L and σ are the liquid density and the surface tension, respectively. This stability criterion stems from the evaluation of the CFL criterion for capillary waves and is more restrictive on the time step size than the standard formulation of the CFL condition [68].

Since the hydrodynamics of both phases are described by the incompressible Navier-Stokes equations, the basic region type of both regions is `icoFluid`. This particular region type implements the volumetric pressure-velocity coupling semi-implicitly and combines it with the interface coupling by splitting the individual predictor-corrector steps within the DNA (see section 4.4.3). To account for species transport and reactions, this region type is superimposed with another region type, which will be described in more detail in 6.2.

6.1 Single rising bubble

In this case the rise of a single, chemically inter air bubble in still water is considered. Here, an initially spherical bubble is released with a zero initial velocity, accelerates for some time until reaching a terminal value of its rise velocity. The physical properties of the two phases and the surface tension coefficient are

$\rho_{L.}$ kg/m³	ρ_G kg/m³	μ_L kg/ms	μ_G kg/ms	σ_0 N/m
998.3	1.205	$1e^{-3}$	$1.82e^{-5}$	0.0727

Table 6.1: Fluid properties of the gaseous and liquid phase

summarised in table 6.1, where subscripts L and G represent the liquid (water) and the gas bubble (air), respectively. The water is ”hyper clean” and the temperature during the experiment is 19.6 ± 0.2 °C.

According to the simulation setup displayed in figure 6.1 the liquid region is bounded by two boundaries: the outer boundary called ”space” and the interface boundary called ”interface”. The gas region is bounded by only one boundary called ”interfaceShadow”, which together with the ”interface” boundary forms the interface between liquid and gas. The boundary conditions for velocity, pressure and mesh deformation applied to the mentioned boundaries are displayed in table 6.2. Following Tuković and Jasak [81], a pressure Dirichlet and a velocity Neumann condition at the liquid side and a pressure Neumann and a velocity Dirichlet condition at the gas side of the interface are applied.

Boundary	U	p	pointMotionU
interfaceShadow	interfaceCoupledVelocityValue	interfaceCoupledPressureFlux	fixedValue
interface	interfaceCoupledVelocityFlux	interfaceCoupledPressureValue	fixedValue
space	inletOutlet	zeroGradient	fixedValue

Table 6.2: Boundary conditions of the numerical domain

As initial conditions, a zero velocity and pressure field was set in both regions. Table 6.3 reports the numerical schemes that are used for the equation discretisation. Their displayed names are in accordance to the equivalent terms used in OpenFOAM.

	Scheme	Setting
Time Scheme	ddtScheme	backward
Finite Volume Schemes	gradScheme (liquid)	Gauss linear
	gradScheme (gas)	extendedLeastSquares 0.6
	divScheme div(phi,U)	Gauss skewCorrected GammaVDC 1.0
	lapacianScheme	Gauss linear corrected
	interpolationScheme	linear
	snGradScheme	corrected
Finite Area Schemes	gradScheme	Gauss linear
	divScheme	Gauss linear
	interpolationScheme	linear

Table 6.3: Numerical schemes used for the simulation of a single rising bubble

For the above described case setup, experimental data from Duineveld [20] is available, which is used to validate the simulation results in section 6.1.3. However, before the flow field evaluation and validation is presented, the results of two additional studies are discussed. These are the parallel scaling of this setup in section 6.1.1 and the evaluation of the interface residuals with respect to the used DNA interface coupling algorithm in section 6.1.2.

6.1.1 Parallelisation study

The simulation of a rising gas bubble with the above explained setup requires high computational effort, due to small time step sizes obeying the stability criterion (6.1) and the need for a strongly refined interface. This can be illustrated by the fact that simulations of earlier studies, carried out with a similar setup but based on highly specialised code for bubbly flows, had runtimes of thirty to sixty days on 6 processors to reach a physical time of 1s [61]. This particular code, used by Pesci et al. [61], Weber et al. [86] and Steinhausen [76] had the limitations that the whole interface needed to be part of a sub-domain when decomposing the computational mesh. Because of the way in which parallelisation is ensured in the multiRegionFOAM framework (see section 5.1.2), this restriction does not apply here.

To show the parallel scaling of the single rising bubble simulation with multiRegionFOAM, five different simulations were performed. A bubble with radius $r_B = 0.5$mm was considered for all simulations. Thereby, the mesh of the liquid region consisted of $125\,545$ cells and the mesh of the gas region consisted of $63\,367$ cells (see figure 6.2). The interface patches were both discretised by $3\,587$ faces. All simulations were executed for 500 iterations, while a fixed time step of $\Delta t = 5e^{-6}$s was used. Furthermore, a maximal interface residual of $10e^{-3}$ was set as the DNA termination criteria for both the velocity transmission and the pressure jump condition. Thus, the simulations varied only in the number of processors to which the calculations were distributed. As a reference, the first simulation was performed on one processor only and thus without decomposition of the computational domain. The other four simulations were performed on $N = 2$, 4, 8 and 16 processors with both the liquid and gas regions decomposed using the *simple* decomposition method. The decomposition of the regions in each spatial direction for the respective number of processors can be found in table 6.4.

N	Sub-domains in (x, y, z) direction
2	$(2, 1, 1)$
4	$(2, 2, 1)$
8	$(2, 2, 2)$
16	$(4, 2, 2)$

Table 6.4: Number of sub-domains in each spatial direction for different number of processors N

For the evaluation of the parallel performance of multiRegionFOAM with respect to the setup considered here, the speedup S and the efficiency E were calculated on the basis of the five performed simulations. The speedup and the efficiency are defined by

$$S(N) = \frac{T(1)}{T(N)} \tag{6.2}$$

and

$$E(N) = \frac{S}{N}\,, \tag{6.3}$$

where $T(1)$ is the runtime of the simulation on one processor and $T(N)$ is the runtime of the simulation on N processors. Figure 6.3 shows both measures over the different number of processors in comparison to the best results of a parallelsisation study carried out by Steinhausen [76] for a similar setup. As a reference for the speedup, an ideal scaling with $S(N) = N$ is drawn by a grey dashed line. The trends of both measures show the expected behaviour for parallel applications. Increasing the number of processors and thus the number of subdomains reduces the runtime but increases the communication overhead between the processors and therefore decreases efficiency and reduces the speedup. Nevertheless, in comparison to the results of

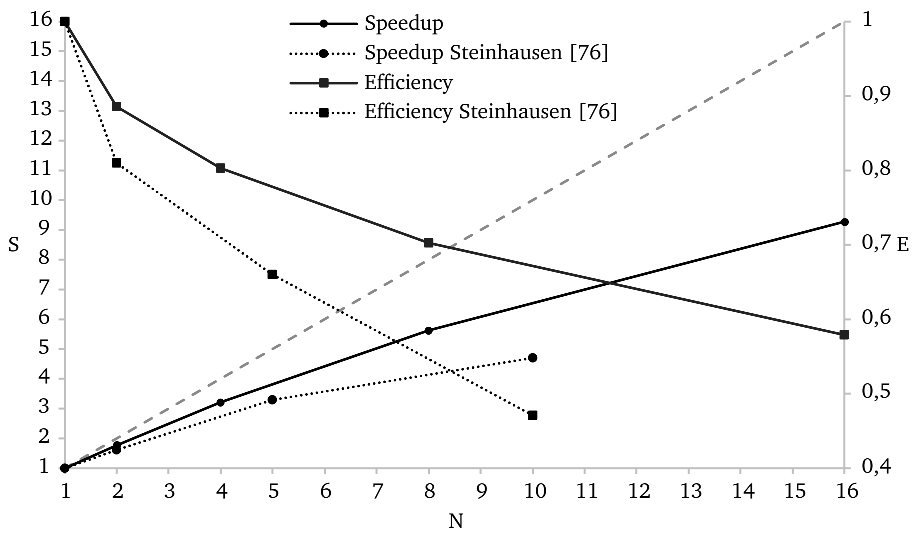


Figure 6.3: Parallel speedup S and Efficiency E of the single rising bubble simulation

Steinhausen [76], the setup on the basis of multiRegionFOAM shows much better parallel scaling. This demonstrates the good performance of the parallelisation approach used in multiRegionFOAM.

Due to the still good efficiency of $E > 0.5$ on 16 processors, all further results presented in this work have been computed on this number of processors.

6.1.2 Interface residual evaluation

To evaluate the performance of the DNA with respect to the interface coupling of two incompressible fluid regions, the interface residuals derived in chapter 4.4.2 are discussed as a measure for the coupling performance. For this purpose, a simulation of a bubble with $r_B = 0.5\,\mathrm{mm}$ is considered again. The setup was chosen in such a way that first the liquid side and then the gas side were solved in one DNA iteration. Due to the use of the `icoFluid` region type for both fluids, the momentum predictor for both regions was always solved first and then the pressure correction loop for both regions was solved thereafter. Furthermore, no termination criterion was specified for the DNA but a fixed number of 20 coupling iterations per time step was set.

Before evaluating the interface residuals, however, it must be clarified which interface residuals are relevant in the described setup. Due to the order in which the regions are treated in the DNA and the arrangement of the coupling boundary conditions, the pressure flux and velocity jump boundary conditions are enforced at the end of each DNA iteration. Vice versa, the pressure jump and velocity flux boundary conditions, which are enforced in the beginning of a DNA iteration, are not implicitly met at the end. Consequently, the pressure jump residual and the velocity flux residuals have to be considered. As already explained in section 4.4.2, it is important to note that due to the derivations of the two coupling boundary conditions, the pressure jump boundary condition represents the normal component and the velocity flux boundary condition represents the tangential component of the momentum transmission condition (3.30). For that reason, the respective interface residuals represent how well the different components of the momentum transmission condition are satisfied.

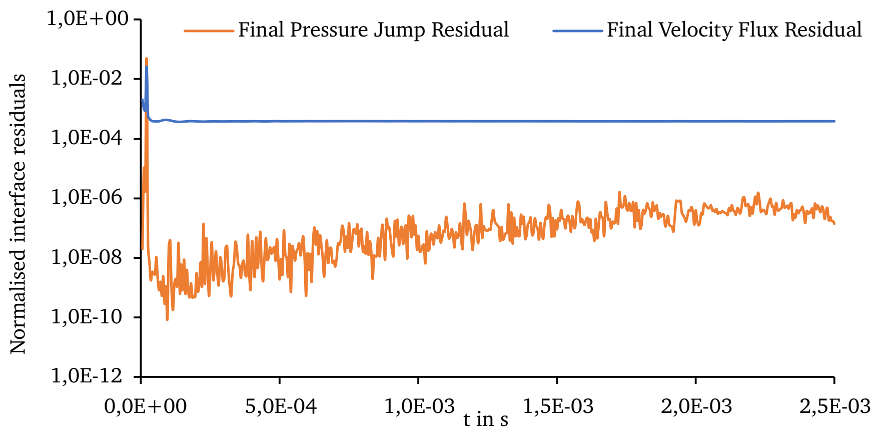


Figure 6.4: Final interface residuals at the end of each time step

Figure 6.4 shows the normalised pressure jump and interface flux residual at the end of each time step. Accordingly, these are the residuals after performing 20 coupling iterations in every time step. The pressure jump residual rises significantly at the very beginning of the simulation, is then reduced just as rapidly as the simulation progresses, and stabilises at about $1e^{-7}$. Also, the initially high fluctuations of the residual decrease over time. The velocity flux residual, as well, rises significantly at the very beginning of the simulation and is then reduced as the simulation progresses. In contrast to the pressure jump residual, the velocity flow residual stabilises very quickly at about 1e-3 without any fluctuation. The initial increase of the residuals can be explained by the fact that at the beginning of the simulation no physically meaningful pressure and velocity fields are present, which changes over the course of the first few time steps. In order to explain the further course of the final residuals, the reduction of the residuals over the 20 coupling iterations must first be considered. This residual reduction for both considered interface residuals over the course of 5 time steps starting at $1.5e^{-3}$s is displayed in figure 6.5. It shows how the DNA enforces the interface coupling of the two fluid regions in each time step by reducing the interface residuals throughout the coupling iterations. The velocity flux residual reaches a plateau after only 7 coupling iterations, whereas the pressure jump residual does not decrease any further after about 16 iterations, which in turn explains the results of the final residuals in figure 6.4. This implies that the error in the normal component of the momentum transmission condition (3.30) is reduced to a greater extent than in the tangential component.

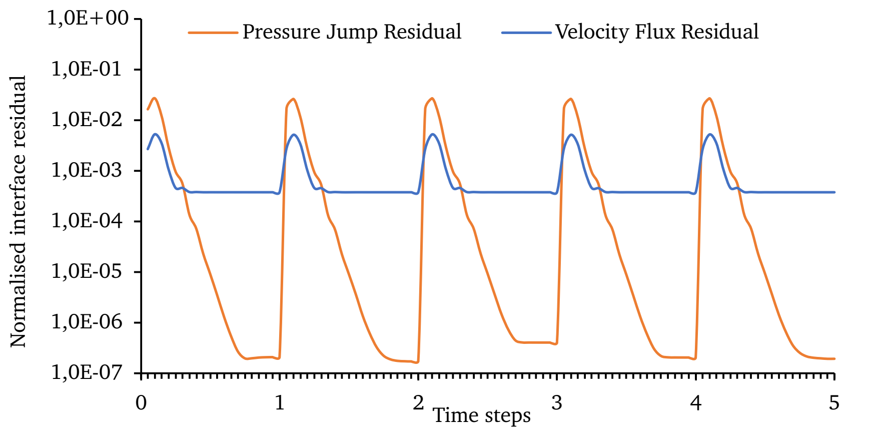
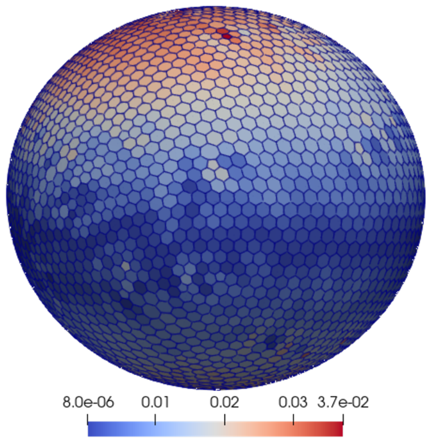
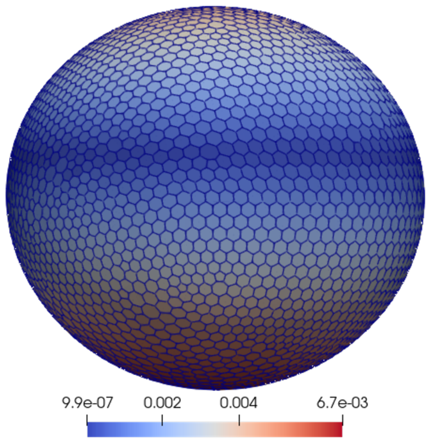


Figure 6.5: Interface residual reduction over 20 coupling iterations per time step

The initial and final unormalised residual fields for the pressure jump and the velocity flux boundary condition at $t = 2.5e^{-2}$s are displayed in figures 6.6 and 6.7, respectively. Comparing the initial pressure jump residual in figure 6.6a with the final pressure jump residual in figure 6.6b, it shows how the residual values decrease and the residual field gets smoothed throughout the coupling iterations. For both initial and final residual the maximum values occur at the top and bottom of the bubble because the highest pressure magnitudes occur at these positions during the rise. The residual fields of the velocity flux boundary condition show a different behaviour. Here, the residual fields are characterised by isolated peaks, which don't get significantly reduced throughout the coupling iterations. This reflects the previously discussed fact that a velocity flux residual plateau is reached after only a few coupling iterations (see figure 6.5). It is noticeable that the residual peaks occur around points of the interface mesh where the aspect ratios of the interface faces are not equal to one. This indicates either inaccuracies in the curvature calculation along the interface or faulty implementations of the FAM operators in the basic OpenFOAM code, which in turn leads to inaccuracies in the calculations of the interfacial gradients used in calculation of $\mathcal{F}_{\boldsymbol{u}}$ (see equation (4.45)). However, this could not be conclusively clarified in the scope of this work.

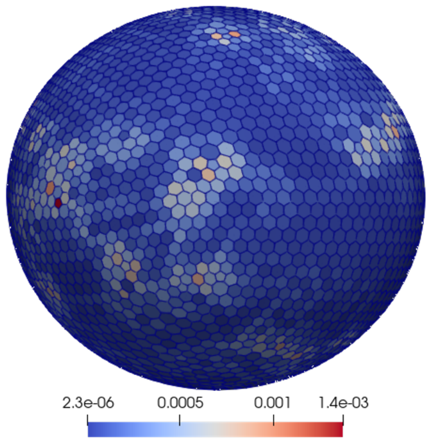


(a) Initial pressure jump residual field

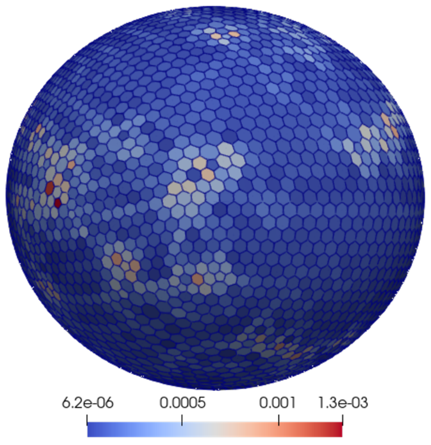


(b) Final pressure jump residual field

Figure 6.6: Pressure jump residual fields across the bubble interface at $t = 2.5e^{-3}$s



(a) Initial velocity flux residual field



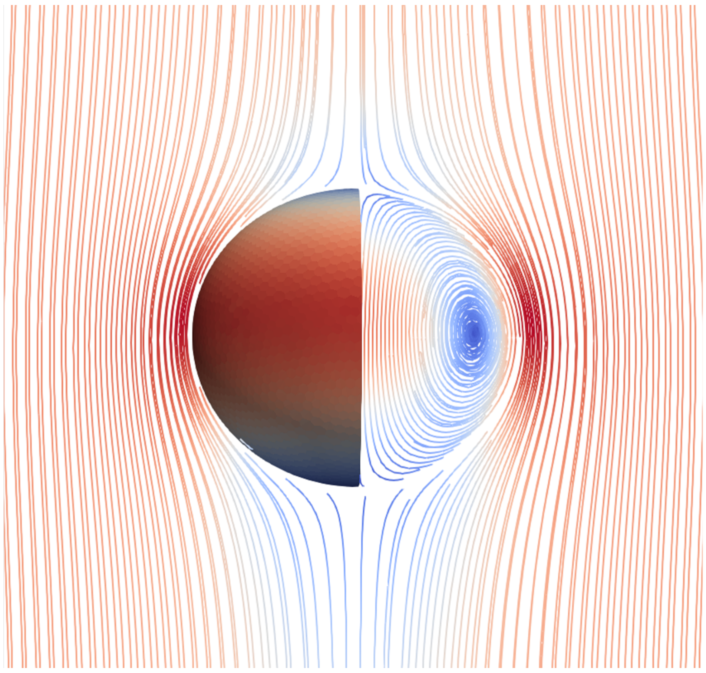
(b) Final velocity flux residual field

Figure 6.7: Velocity flux residual fields across the bubble interface at $t = 2.5e^{-3}$s

6.1.3 Flow field evaluation and comparison with experimental data

For the evaluation of the flow field and the comparison with experimental data by Duineveld [20], three rising bubbles with different sizes were simulated with multiRegionFOAM. To get a good comparison throughout the investigated range of bubble radii by Duineveld $r_B \in [0.3, 1.0]$ mm, the simulated bubble sizes were set to $r_B = 0.35$ mm, 0.5 mm and 0.8 mm. All three cases were simulated with the DNA termination criteria set to $1e^{-3}$ for both the velocity flux and the pressure jump interface residual, which resulted in a mean number of coupling iterations per time step of 5.

Since the simulated bubbles all undergo only moderate deformation during the rise, the corresponding flow fields are qualitatively very similar. Therefore, only the flow field of the bubble with radius $r_B = 0.5$ mm is presented as a representative for the flow fields of all simulated bubble sizes. Figure 6.8a shows the streamlines of the flow around and inside of the bubble, coloured according to the local velocity magnitude. Here, one can clearly see the circulating flow pattern inside the gas bubble and the flow field of the liquid phase, which strongly resembles a potential flow around a sphere. In figure 6.8b the pressure field on the liquid side of the interface is displayed. This corresponds to the expected state with the highest pressures at the top and bottom of the bubble. However, the otherwise very smooth pressure field is locally disturbed at certain points. These points coincide with the local spikes in the velocity flux residual field, displayed in figure 6.7. Therefore, these small disturbances of the pressure field can probably also be attributed to an inaccurate curvature calculation and/or a faulty implementation of the FAM operators in OpenFOAM.


(a) Streamlines coloured according to the velocity magnitude (blue is slow)

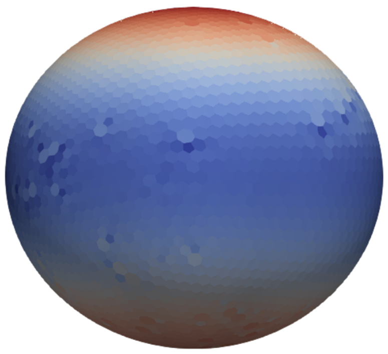

(b) Interface pressure field on the liquid side (blue is low)

Figure 6.8: Streamlines and interface pressure field of the single rising bubble

In figure 6.9 the terminal rise velocities of the three simulated bubbles with initial radii of $r_B = 0.35$ mm, 0.5 mm and 0.8 mm are compared to the experimental measurements by Duineveld [20]. The numerically determined terminal rise velocity of the smallest bubble matches the experimental values accurately, while the rise velocities of the middle sized bubble is slightly underestimated. For the biggest bubble the deviation

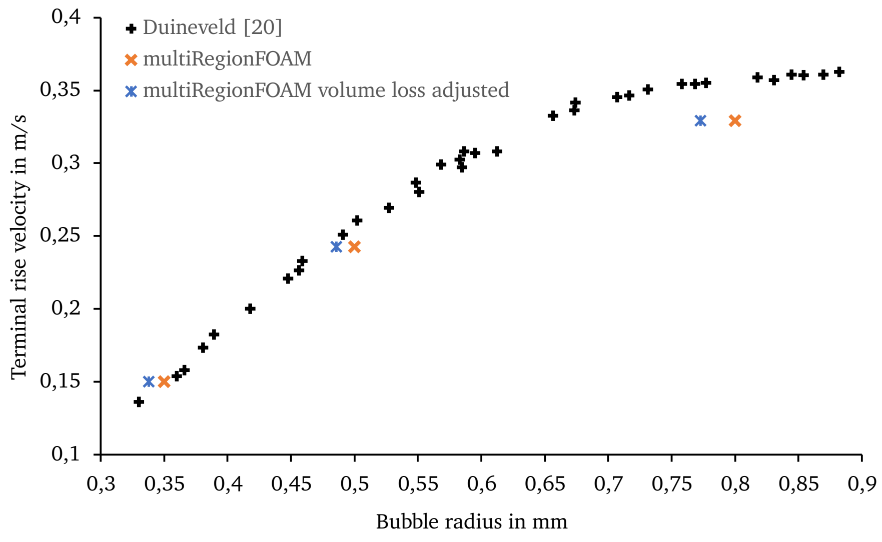


Figure 6.9: Comparison of numerical results to experimental data by Duineveld [20]

from the experimental data is the greatest, as the terminal rise velocity is significantly underestimated. If additionally the volume of each bubble is also evaluated during the simulation, it becomes apparent that all considered bubbles slightly shrink during the rise. Thereby, the bubble with $r_B = 0.35$ mm looses 10.0%, the bubble with $r_B = 0.5$ mm looses 8.2% and the bubble with $r_B = 0.8$ mm looses 9.7% of its volume until reaching the terminal velocity. Here, the terminal rise velocity was considered to be the maximum rise velocity reached, as it gradually decreases afterwards due to a continued volume loss over time (see figure 6.10). Thus, when the terminal rise velocities are reached, the three bubbles no longer correspond to the bubbles with their initial equivalent radius, but represent bubbles with an equivalent radius of $r_B = 0.338$ mm, 0.486 mm and 0.772 mm, respectively. If the terminal rise velocities of the three volume loss adjusted bubbles are again compared to the experimental data, a slightly different picture arises. While the rise velocity of the smallest bubble is now slightly overestimated, the rise velocity of the middle sized bubble accurately fits the experimental data. In the case of the biggest bubble, the volume loss adjustment improves the numerically determined rise velocity only marginally in comparison to the measurements of Duineveld.

This slight overestimation of the terminal rise velocities for bubbles of sizes around $r_B = 0.35$, a good match for bubble sizes around $r_B = 0.5$ and an underestimation for bubble sizes around $r_B = 0.8$ resembles the results of Deising [16], who conducted a similar study based on the VOF solver interFOAM. It should be noted, however, that the deviations using iterFOAM were much larger than those using multiRegionFOAM presented here. According to Deising [16], the main reason for these derivations was a deficient curvature calculation in the interFOAM solver. Since an inaccurate curvature calculation was also suspected in the case of the velocity flux residuals and the locally disturbed pressure field presented above, it is reasonable to assume that the curvature calculation is also the cause of the described deviation in the terminal rise velocity results.

In addition to the potentially deficient curvature calculation, the loss of volume represents an issue that needs to be addressed in future work, as it exceeds the scope of this thesis. In doing so, multiple factors need to be considered. Since ALE-IT is not an inherent volume conservative method, the use of a stronger refined interface should decrease the volume loss significantly. Furthermore, in the presented setup, the correction volume used for the interface displacement calculation is computed based on the volume flux over the interface on the liquid side. As the velocity flux boundary condition is applied on this side of the interface, the further reduction of the velocity flux interface residuals should be beneficial for the interface tracking accuracy.

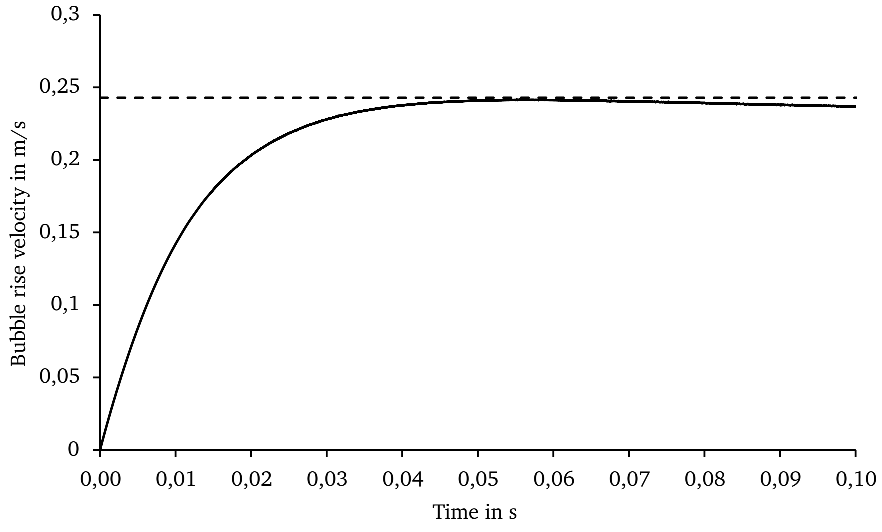


Figure 6.10: Bubble rise velocity over time for a bubble with initial radius of $r_b = 0.5mm$

Despite the discussed difficulties, it was clearly shown that the new unified multiphysics framework multiRegionFOAM, at this stage, provides all the foundations for accurately simulating the behaviour of bubbles with ALE-IT in the future.

6.2 Reactive single rising bubble

The architecture of multiRegionFOAM makes it possible to additionally account for species transport and reactions by superimposing another region type on the `icoFluid` region type, previously used for the inert gas bubble simulation. This additional region type called `transportSpeciesConcentration` sets up the linear equation systems which result from the species concentration transport equations. In this process, it employs the flux fields previously calculated through the utilisation of `icoFluid` and is therefore evoked every time step after the interface coupled hydrodynamic is solved. A distinction is made between species that only occur in one region and species that can transfer across the interface. Hence, a DNA coupling algorithm and corresponding interface coupled boundary conditions are only applied to those species that can transfer across the interface. The `chemistryModel` and `reactiveMixture` class from the OpenFOAM framework are used in `transportSpeciesTemperature` to account for chemical reactions, which can be specified by the user in form of reaction models. In the course of this work, only the use of constant diffusion coefficients for the individual species was implemented in `transportSpeciesTemperature`. Nevertheless, an extension by other diffusion models is easily possible in the future. Also no Subgrid-Scale model (SGS) has been implemented yet, to accurately compute the concentration boundary layer without the need for a highly refined mesh. However, all the foundations have been laid for the future extension of the species transport computations with Subgrid-Scale models such as that of Bothe and Fleckenstein [9] or Wiener and Bothe [88].

The same basic setup as for the chemically inert bubble is used for the reactive rising bubble case, which multiRegionFOAM is tested against. Therefore, the fluid properties of the liquid and gaseous phase and the boundary conditions for the velocity, pressure and mesh deformation field can be found in tables 6.1 and 6.2, respectively. Regarding the additional species transfer and reaction, the transfer of a generic species A from the gas to the liquid phase and the subsequent reaction with another species B is considered. An irreversible reaction of the two reactants

$$\mathrm{A} + \mathrm{B} \xrightarrow{k} \mathrm{P} \tag{6.4}$$

with a constant reaction rate coefficient k is assumed. This implies that three species are considered in this setup with the respective reaction rates being calculated by

$$\tilde{r}_B = \tilde{r}_A = -\tilde{r}_P = -kc_Ac_B. \tag{6.5}$$

The properties of the species under consideration can be found in table 6.5, where $D_{i,L}$ and $c^0_{i,L}$, and $D_{i,G}$ and $c^0_{i,G}$ are the diffusion coefficient and the the initial concentration of each species in the liquid and gas phase, respectively. Since no SGS and the same mesh as for the inert rising bubble is used, high diffusion coefficients for the species are assumed. Therefore, the Schmidt number for all species in both regions are close to one, which indicates that both the hydrodynamic and the concentration boundary layer are of equal size [7, Ch. 22.2]. The reaction rate coefficient is set to a constant value of $k = 45$ $\mathrm{m^3/mol{\cdot}s}$.

	M_i $\mathrm{kg/mol}$	$D_{i,L}$ $\mathrm{m^2/s}$	$D_{i,G}$ $\mathrm{m^2/s}$	$c^0_{i,L}$ $\mathrm{mol/m^3}$	$c^0_{i,G}$ $\mathrm{mol/m^3}$	$H_i = c_{i,G}/c_{i,L}$
A	40	$1e^{-6}$	$2e^{-5}$	0.0	1.0	30
B	20	$1e^{-6}$	–	1.0	–	–
P	60	$1e^{-6}$	–	0.0	–	–

Table 6.5: Species properties

The boundary conditions set for the three species in the flow domain can be found in Table 6.6. Here, the interface coupled boundary conditions for species transfer are defined on the two interface patches for species A. The coupled Dirichlet boundary condition is applied to the liquid side of the interface and the coupled Neumann boundary condition is applied to the gas side of the interface, which satisfies the stability criterion for DNA coupling of the species transport equations derived by Weber et al. [86]

$$H_A\sqrt{\frac{D_{A,G}}{D_{A,L}}} > \frac{2}{\pi}\,. \tag{6.6}$$

Boundary	A	B	P
interfaceShadow	interfaceCoupledSpeciesFlux	–	–
interface	interfaceCoupledSpeciesValue	zeroGradient	zeroGradient
space	inletOutlet	inletOutlet	inletOutlet

Table 6.6: Species concentration boundary conditions

Despite the described setup with low Schmidt numbers of about one in both phases and the compliance with stability condition 6.6, it was not possible to perform a stable reactive rising bubble simulation with multiRegionFOAM within the scope of this work. Both with and without the described reaction in the liquid phase, the DNA coupling of the species transport equation for species A diverged after a few time steps, which subsequently led to the abortion of the simulation. The cause of the failed interface coupling for species transport could not be clarified by the time this thesis was completed. However, in addition to the cause of an unresolved software bug that could lead to this error, future research should investigate the influence of a stronger refined grid or the initially unphysical pressure and velocity fields on the stability of the species transfer coupling.

7 Computation of Lagrangian Coherent Structures

Transport and mixing in fluids is advanced by advection. As can be seen in the derivation of the sharp interface model in chapter 3, advection is usually described mathematically in an Eulerian view by a time dependent velocity field $\boldsymbol{u}$. In this Eulerian description the mixing can in general be estimated by the Reynolds number, for which an increase in its value will go along with an overall better mixing. However, this correlation is not always true, as can be seen in a study of Kameke et al. [40]. In this study a coherent structure arises behind a bubble for intermediate Reynolds numbers, which causes material to move together and locally hinders mixing. This demonstrates that the evaluation of such coherent structures is necessary to evaluate the details of the material transport and mixing processes.

To compute such coherent structures the Lagrangian reference frame is utilised. In the Lagrangian view, trajectories of individual fluid parcels are easily computed and conclusions about the transport can be drawn from their evaluation. In particular, stable and unstable manifolds, to which the fluid parcels converge or diverge, can be derived from the fluid parcels trajectories. These stable and unstable manifolds divide the flow into different subdomains that move coherently [4] and are therefore called Lagrangian Coherent Structures.

Several different approaches to evaluate LCS have been developed during the last years [26]. In this thesis, an OpenFOAM function object is introduced and validated that calculates LCS candidates based on the three-dimensional FTLE fields on the fly. In doing so, the function object utilises the numerical library libcfd2lcs [39], whose theoretical background is explained in the following.

7.1 Finite-time Lyapunov exponent field calculation

From the underlying time resolved CFD simulations the time dependent velocity field $\boldsymbol{u}(\boldsymbol{x}, t)$ is known in space and time. From this information the fluid parcel trajectories

$$\boldsymbol{x}(\boldsymbol{x_0}, t) = \boldsymbol{x}_0 + \int_{t_0}^{t} \boldsymbol{u}(\boldsymbol{x}(\tau), \tau) d\tau \tag{7.1}$$

can be calculated, where $\boldsymbol{x_0}$ is the starting point of a trajectory in 3D space at a starting time t_0. Thereby, each trajectory is labelled by its initial location in space and time. If a set of initially close passive particles is released at the same time the distances between them change in the course of time due to the fluid motion. Passive particles initially forming a tiny sphere will undergo a linear deformation towards an ellipse for short times as would occur in a solid body under stress before it breaks. Certainly, in a fluid, the deformation will progress and non-linear, higher order terms will start to play a role causing the stretching and folding that is important for the mixing. However, as a first approximation and for short times these higher order terms are neglected for the analysis of the deformation. Therefore, if the fluid parcel trajectories are calculated for a short period $T = [t_0, t_1]$, regions where the stretching and thus particle separation is largest due to the flow can be calculated. This is done by first calculating the so called flow map $\Phi_{t_0}^{t_1}$, that maps all the particles from their initial positions onto their final positions at time t_1:

$$\Phi_{t_0}^{t_1} : \mathbb{R}^n \to \mathbb{R}^n; \quad \boldsymbol{x}_0 \mapsto \boldsymbol{x}_0 + \int_0^{t_1} \boldsymbol{u}(\boldsymbol{x}(\tau), \tau) d\tau \ . \tag{7.2}$$

With this flow map the right Cauchy-Green deformation tensor can be computed by

$$\boldsymbol{C}_{t_0}^{t_1}(\boldsymbol{x}_0, t_0) = \left[\boldsymbol{D}\Phi_{t_0}^{t_1}(\boldsymbol{x}_0, t_0)\right]^* \left[\boldsymbol{D}\Phi_{t_0}^{t_1}(\boldsymbol{x}_0, t_0)\right] \tag{7.3}$$

where $\mathbf{D\Phi}_{t_0}^{t_1}(\mathbf{x}_0, t_0)$ is the gradient (Jacobian) of the flow map with regard to the initial separation and $\boldsymbol{M}^*$ is the adjoint (transpose) of a matrix $\boldsymbol{M}$. The root of the largest eigenvalue of the right Cauchy-Green deformation tensor is the largest stretching that occurs [29]. The scaled logarithm of this stretching factor is then called Finite-Time Lyapunov Exponent and is defined by

$$\sigma_{t_0}^{t_1}(\mathbf{x}_0, t_0) = \frac{1}{|t_1 - t_0|} \log \sqrt{\lambda_{\max}\left(\mathbf{C}_{t_0}^{t_1}(\mathbf{x}_0, t_0)\right)} \ . \tag{7.4}$$

Connected manifolds of large FTLE values shape the fluid transport as these denote the areas along which deformation and thus particle separation is largest.

The same evaluation can be done if the fluid parcel trajectories are computed backwards in time. In this way the backward time flow map $\Phi_{t_1}^{t_0}$ maps all the particles from their final positions at time t_1 onto their initial positions at time t_0. Subsequently, the Jacobian and the right Cauchy-Green deformation tensor can be computed base on this flow map, which leads to the backward time FTLE

$$\sigma_{t_1}^{t_0}(\mathbf{x}_1, t_1) = \frac{1}{|t_1 - t_0|} \log \sqrt{\lambda_{\max}\left(\mathbf{C}_{t_1}^{t_0}(\mathbf{x}_1, t_1)\right)} \ . \tag{7.5}$$

Connected manifolds of large backward time FTLE values also shape the fluid transport as these denote the areas along which deformation and thus particle separation is largest in backward time. This implies that they denote the attracting material structures in forward time.

7.2 Function object for LCS computation

In general, function objects can be used to generate additional data at runtime of the simulation. In doing so, function objects can access data generated by the flow solver at runtime, which offers a great advantage over classical post-processing since it can only utilise the stored fields or logged information. The newly developed function object described in this thesis incorporates the functionalities of the Fortan library libcfd2lcs into OpenFOAM at runtime while acting as an interface between both. This is achieved by processing the data generated by OpenFOAM and the subsequent exchange of this data via the libcfd2lcs C++ API (see [39] for a detailed description of the libcfd2lcs API).

The calculation of the backward and forward time flow maps, the calculation of the resulting FTLE fields and the subsequent saving of these fields is completely handled by libcfd2lcs. Computational details on how libcfd2lcs computes these fields can be found in [39] and [24]. The basic task of the function object is therefore to pass the cell centre position vectors of the computational mesh as well as the velocity field calculated with OpenFOAM to libcfd2lcs. Due to the very strict data structure requirements of libcfd2lcs this is not a trivial task. libcfd2lcs can only use static rectlinear grids for the calculation of forward-time and backward-time flow maps and therefore needs the velocity fields on these grids. This means that the mesh and velocity data has to be globally organized in an i, j, k structured format [39, p. 6]. Since the LCS evaluation should also be available for simulations on moving meshes with general topology and adaptive grid refinement, the function object offers possibilities to deal with this problem.

In the simplest case, where the simulation mesh is already a static rectlinear mesh, the function object does not need to process the grid and velocity data, but can directly transfer it to libcfd2lcs as basic C++ arrays. This is the preferred method when the flow domain can be represented by a static rectlinear mesh. If a moving mesh, a mesh of general topology or adaptive mesh refinement is used for the simulation a different approach is needed in order to prepare the data for its use in libcfd2lcs. Here, an additional static rectlinear mesh needs to be constructed in the preprocessing step. This mesh has to contain the volume for which the LCS diagnostic should be performed, meaning that it can cover the whole simulation domain as well as only a part of it. During runtime, the velocity fields are then mapped from the simulation mesh of general topology to the static rectlinear LCS mesh from which the data can again be transferred to libcfd2lcs as basic C++ arrays. Although this implies that interpolation errors are made during the mapping process, the LCS evaluation is hardly affected by this, since Haller showed in [28] that LCS are very robust against errors in the velocity field. Also the additional computational overhead due to the mapping can be neglected compared to the overhead caused by the flow map computations in libcfd2lcs.

7.3 Validation

Before the newly developed function object can be used to evaluate the LCS of bubbly flows in the future, it must be validated against established benchmark test cases for which the LCS are known. Therefore, three test cases, namely the steady Arnold-Beltrami-Childress (ABC) flow, the time depended double gyre and the flow around a steady cylinder, are presented in this section which are designed to show the capabilities of the function object as well as validate the computed results.

7.3.1 Steady Arnold-Beltrami-Childress flow

The Arnold-Beltrami-Childress (ABC) flow is an exact periodic solution of the Euler equations and is often used in the literature to verify LCS calculation methods. Therefore this case is also being reviewed here. The velocity field $\boldsymbol{u}$ in the ABC flow can be described using two scalar potentials Ψ and Θ [79]

$$\boldsymbol{u} = \nabla \times [-\Psi \boldsymbol{k} + \nabla \times (\Theta \boldsymbol{k})] \tag{7.6}$$

which themselves are defined as

$$\begin{aligned} \Psi &= -[C \sin(y) + B \cos(x)] \\ \Theta &= A[-x \cos(z) + y \sin(z)] - \Psi \ . \end{aligned} \tag{7.7}$$

In equation (7.6) $\boldsymbol{k}$ can be any unit vector but is commonly chosen to be the vertical unit vector. This leads to the three expressions of the velocity components

$$\begin{aligned} u &= A \sin(z) + C \cos(y) \\ v &= B \sin(x) + A \cos(z) \\ w &= C \sin(y) + B \cos(x) \ . \end{aligned} \tag{7.8}$$

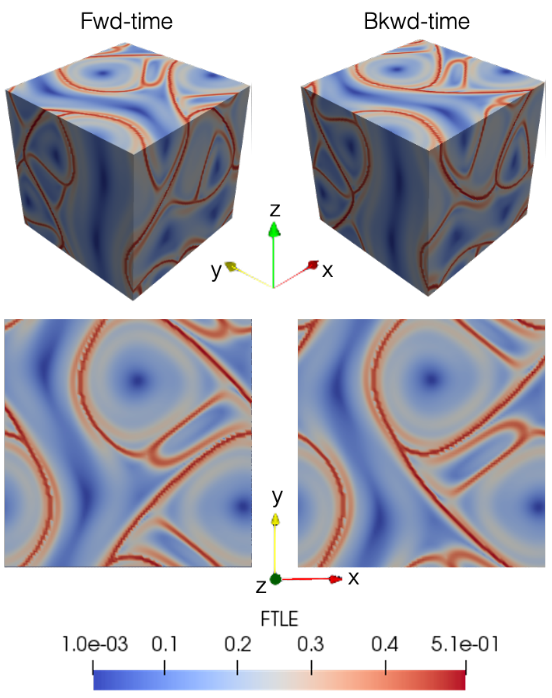


Figure 7.1: Forward-time and backward-time FTLE fields of a stationary ABC flow for the whole computational domain $x, y, z \in [0, 2\pi]$ at the top and for a cross-section at $z = 2\pi$ at the bottom.

The parameters A, B and C can be freely selected and influence the properties of the ABC flow. In order to create comparability with literature values, $A = 0.5$, $B = 0.8$, $C = 0.8$ is chosen. In order to test the newly developed function object on this flow configuration a dedicated ABC flow solver was written. This solver does not solve the Euler equations in the usual sense, but sets the velocity components on a given computational mesh according to (7.8). Due to the periodicity of the flow solution, the dimensions of the computational mesh used in this case setup are specified as $x, y, z \in [0, 2\pi]$ with a mesh size of $100 \times 100 \times 100$. Since the described mesh is rectlinear no additional LCS mesh is used. Again for reasons of comparability, a LCS integration time of $T = 10\,\mathrm{s}$ is selected for the LCS evaluation.

The results of the LCS evaluation, both in forward- and backward-time, can be seen in figure 7.1. In these results the FTLE ridges, which indicate the LCS in the ABC flow, can be seen very clearly. Furthermore, the results agree very well with the results from [79], both qualitatively and quantitatively, which suggests that the new function object calculates the FTLE ridges reliably.

7.3.2 Time dependet double gyre

Another frequently used flow for the verification of LCS computing algorithms is the time periodic Rayleigh-Bénard convection flow, or often called double gyre, proposed by Solomon and Gollub [74]. The velocity field of this flow can be describe by using a stream function ψ:

$$\begin{aligned} u &= -\frac{\partial \psi}{\partial y} \\ v &= \frac{\partial \psi}{\partial x} \,. \end{aligned} \tag{7.9}$$

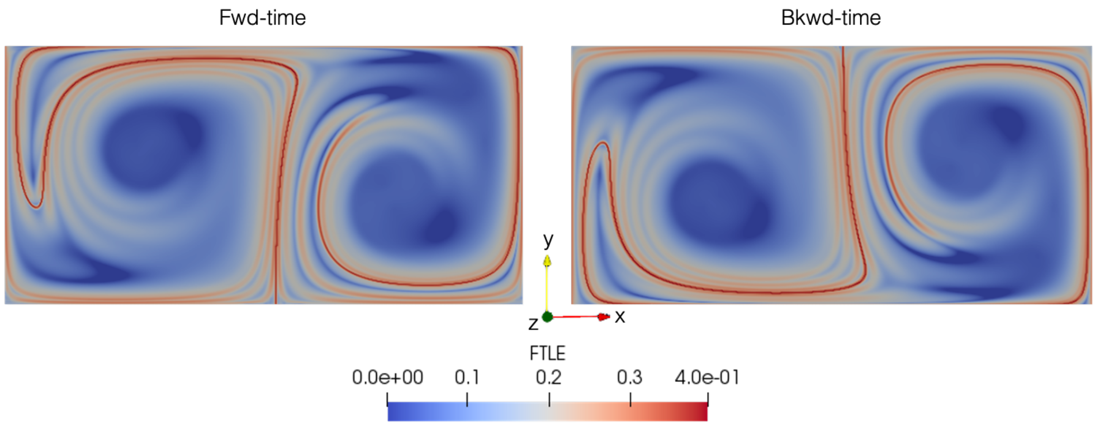


Figure 7.2: Forward-time and backward-time FTLE fields of a time-dependent double gyre at $t = 15s$.

Here ψ is defined by

$$\psi(x, y, t) = A\sin(\pi f(x, t))\sin(\pi y) \tag{7.10}$$

with

$$\begin{aligned} f(x, t) &= a(t)x^2 + b(t)x \ , \\ a(t) &= \epsilon\sin(\omega t) \ , \\ b(t) &= 1 - 2\epsilon\sin(\omega t) \ . \end{aligned} \tag{7.11}$$

This leads to the expressions for two-dimensional velocity components

$$\begin{aligned} u &= -\pi A\sin(\pi f(x))\cos(\pi y) \\ v &= \pi A\cos(\pi f(x))\sin(\pi y)\frac{\mathrm{d}f}{\mathrm{d}x} \ . \end{aligned} \tag{7.12}$$

As the name double gyre suggests, this model defines the flow of two two-dimensional gyres enclosed in a rectangle which expand and contract periodically along the x-axis. Therefore, the periodic motion is controlled by ϵ if $\epsilon \neq 0$. Then ϵ describes approximately how far the line separating the gyres moves to the left or right from its centre position [70]. Otherwise ($\epsilon = 0$), no periodic motion is happening. Furthermore, A specifies the magnitude of the velocity vectors and $\omega/2\pi$ determines the oscillation frequency of the gyres.

Similar to the ABC flow example, a dedicated OpenFOAM solver was written for this case, which sets the velocity field on a given computational mesh according to (7.12). For comparability a mesh with the same specifications as in [46],[45] and [70] was used. It has the dimensions $[0, 2] \times [0, 1] \times [0, 0.1]$ m and a resolution of $512 \times 256 \times 1$ cells. As this mesh is also static and rectlinear no additional LCS mesh was used. For the mathematical model of the flow the parameter values are chosen to be $\epsilon = 0.1$, $A = 0.1\,\mathrm{m/s}$ and $\omega = 2\pi/10\,\mathrm{s}$. The LCS integration time interval T was set to $T = 15\,\mathrm{s}$. Figure 7.2 shows the forward- and backward-time FTLE fields at $t = 15\,\mathrm{s}$ of the previously described double gyre flow. Again the results match very well with the results from [46],[45] and [70]. This confirms that the function object is able to calculate the correct FTLE fields from velocity fields generated by OpenFOAM solvers.

7.3.3 Flow around cylinder

It has already been shown in the previous two examples that the function object can calculate the correct FTLE fields from velocity fields provided by OpenFOAM solvers. This example will focus on how the function

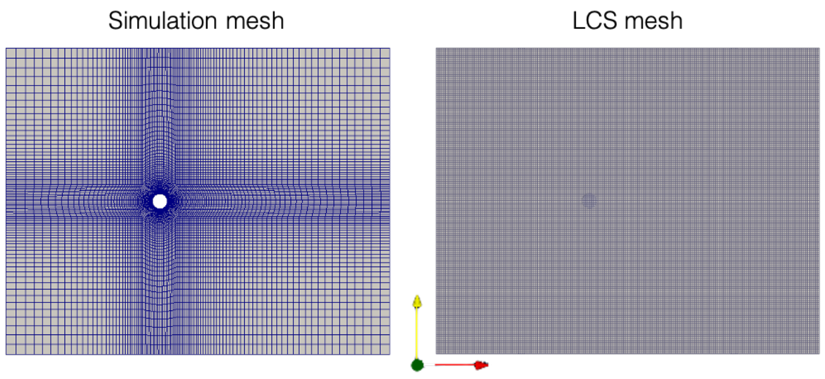


Figure 7.3: Separate meshes for flow field and LCS computation

object performs with an underlying non-rectlinear simulation mesh. For this purpose, a standard flow problem is selected that is very well suited for a LCS evaluation: the flow around an infinitely long cylinder.

The general case setup contains a fluid domain with size $[-20, 30] \times [-20, 20] \times [-0.5, 0.5]$ m that surrounds a cylinder with diameter $D = 2$ m and its centre axis at $x = y = 0$ m. The free-stream velocity and the fluids kinematic viscosity are set to $\boldsymbol{u}^\top = (1\ 0\ 0)\,\mathrm{m/s}$ and $\nu = 0.01\,\mathrm{m^2/s}$, respectively. This results in a Reynolds number of $Re = 200$ which indicates that vortex shedding behind the cylinder occurs in a barely laminar regime.

Because of the cylinder in the middle of the domain, a computational mesh that discretises this domain is no longer rectlinear. Therefore, an additional LCS mesh is used to which the velocity field is mapped and on which the FTLE field is computed subsequently (see figure 7.3). The additional LCS mesh that is used within this procedure encloses the whole flow domain. In order to minimise the loss of information during the mapping of the velocity fields between the two grids, the resolution of the LCS mesh is chosen in a way that it corresponds approximately to the finest resolution in the simulation mesh. This leads to a LCS mesh with $200 \times 160 \times 1$ hexahedra. The OpenFOAM flow solver that is used to simulate the previously described flow from $t = 0s$ to $t = 120s$ is pimpleFoam with the initial conditions being calculated by potentialFoam.

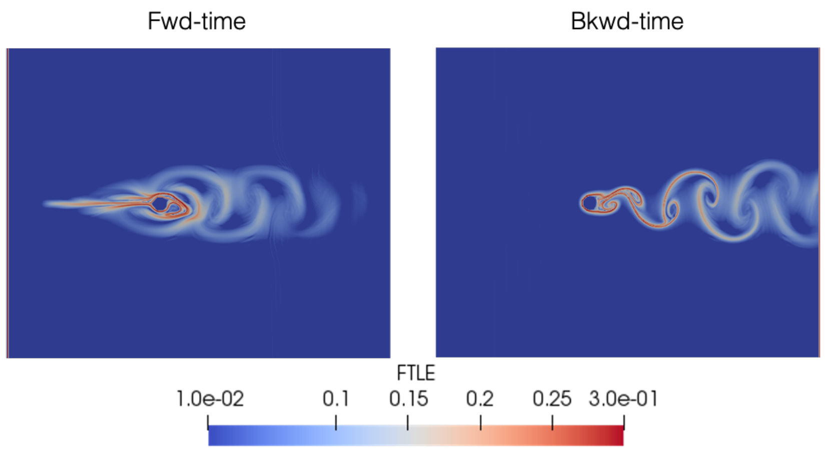


Figure 7.4: Forward-time and backward-time FTLE fields of a flow around a cylinder with $Re = 200$ at $t = 118s$.

In order to detect all LCS candidates, the LCS integration time T is usually chosen to be larger than dominant hydrodynamic timescale t_{ref} of the investigated flow [24]. This hydrodynamic time scale of the flow considered here is $t_{ref} = D/(St{\cdot}u) = 10s$, with an assumed Strouhal number of $St = 0.2$ at $Re = 200$. The LCS integration is therefore set to $T = 1.5 \cdot t_{ref} = 15s$.

The results of the forward- and backward-time FTLE fields can be seen in figure 7.4. They show very well how the vortices behind the cylinder form large coherent structures, where the FTLE ridges of the backward-time FTLE fields separate different fluid packages that do not mix in the vortex street.

8 Conclusion and Outlook

The first main goal of this thesis was to present the generic structure of interface coupled multiphysics systems to which reactive bubbly flows belong to. This was done in chapter 2 by first distinguishing between different types of multiphysics problems and then discussing the generic structure of the multiregion type. Consequently, the generic sharp interface model was derived in chapter 3, which is composed of the general transport equation in the bulk and the generic interface transmission and jump condition at the interface. Subsequently this generic model was specified for mass, momentum and reactive species transport, since reactive bubbly flows were considered in the following. In doing so, it became apparent that especially the generic formulation of the jump and transmission conditions applies to all these specialisations.

The numerics needed to simulate rising bubbles in a moving reference frame, using the FVM and ALE interface tracking, were presented thereafter. Thereby, special attention was paid to the numerical methods for both volumetric and interface coupling, as these play a major role in the simulation of all kinds of interface coupled multiphysics systems. Regarding the volumetric pressure-velocity coupling, the derivations of both the semi-implicit and the fully-implicit coupling methods were brought into line with each other. Regarding the interface coupling, the partitioned solution procedure called DNA was discussed. Here, the generic coupled Dirichlet and Neumann boundary conditions were formulated and again specialised for mass, momentum and reactive species transport. Since the DNA lacked a criteria to reliably evaluate the interface coupling, such a criteria needed to be derived. This represented the second main goal of this thesis. Therefore, so called interface residuals were formulated based on the generic coupled boundary conditions in section 4.4.2, for which an additional normalisation procedure was proposed.

The third main goal of this work was to introduce and test the novel, highly parallelisable multiphysics framework multiRegionFOAM for OpenFOAM, which exploits the generic structure of interface coupled multiphysics problems and implements the DNA coupling steered by the formulated interface residuals. The architecture and implementation details of multiRegionFOAM in general were explained in chapter 5. Also, the architecture and implementation details of the generic interface coupled boundary conditions and their specialisations, the implementation of the DNA solution control and the implementation of the interface tracking mesh were discussed in particular. As multiRegionFOAM is intended to be used to simulate reactive bubbly flows, among other multiphysics problems, in the future, the current status of the framework was tested against its application to both a chemically inert and a reactive rising bubble.

Regarding the single inert rising bubble, multiRegionFOAM proved to be very well applicable to this kind of simulation. However, the comparison of the numerical results to experimental measurements showed slight deviations over the range of different bubble sizes. Furthermore, a small volume loss of the bubbles could be observed in the course of the computations. Both of these issues could be associated with a possibly inaccurate curvature calculation at interface faces with an aspect ratio unequal to one or possible inconsistencies in the implementation of the FAM operators in OpenFOAM. In the course of this inert rising bubble study, the parallel scaling of multiRegionFOAM and the applicability of the interface residuals for the steering of the DNA have also been assessed. Here, multiRegionFOAM proved to provide much better parallel scaling than previously used codes for the simulation of bubbly flows with ALE-IT. The interface residuals also proved to be a good indicator of the convergence of the DNA and also provided detailed, spatially resolved information of the coupling accuracy when evaluating the interface residual fields.

Regarding the reactive rising bubble, no stable simulation could be conducted in the scope of this work. This was due to a diverging DNA for the reactive species, even though the stability criterion derived by Weber et al. [86] was met. Therefore, it could not be conclusively clarified what led to this divergence and the subsequent abortion of the simulation.

With respect to multiRegionFOAM it can be concluded that all foundations have been laid for it to be used for the simulation of reactive bubbly flows. However, further research must be carried out to reassess the interface curvature calculation and the implementation of the FAM operators in OpenFOAM. Also, the coupling of reactive species transport across interfaces needs to be further investigated in order to make multiRegionFOAM applicable to reactive bubbly flows in the future. Moreover, further work should focus on integrating surfactant transport and the use of Subgrid-Scale models in multiRegionFOAM. In addition, the effect of fully-implicit pressure-velocity coupling, as well as the use of monolithic interface coupling on the overall convergence of reactive bubbly flows could be investigated.

The ability to quantitatively assess the mixing processes in reactive flows in general and in reactive bubbly flows in particular is essential to optimise these processes concerning high yields and little undesired by-products. Since LCS can be used for that purpose, the fourth main goal of this thesis was to introduce a novel function object that computes LCS-candidates based on FTLE fields by utilising the Fortran library libcfd2lcs. In doing so, a brief overview on the theoretical background of the FTLE field computation was given. The results of the function object were validated against three established benchmark cases for FTLE field computation. Consequently, this function object enables, from now on, the on the fly geometrical assessment of the material transport in practically any flow and can therefore be used to evaluate the mixing processes in reactive bubbly flows in future work.

Acronyms

ABC Arnold-Beltrami-Childress. 6, 68, 69

ALE Arbitrary Lagrangian-Eulerian. 28, 29, 33, 36, 73

ALE-IT Arbitrary Lagrangian-Eulerian Interface Tracking method. 13, 14, 28, 55, 63, 64, 74

BDS Backward Differencing Scheme. 30, 32

CFD Computational Fluid Dynamics. 13, 66

CFL Courant-Friedrichs-Lewy. 56

CHT Conjugate Heat Transfer. 13

CPU Central Processing Unit. 51

CV Control Volume. 6, 28, 29, 30, 32

DDM Domain Decomposition Method. 37

DNA Dirichlet–Neumann Algorithm. 14, 28, 37, 38, 40, 41, 42, 43, 44, 48, 53, 57, 59, 60, 62, 64, 65, 73, 74

FAM Finite Area Method. 28, 61, 62, 73, 74

FSI Fluid-Structure Interaction. 13, 14

FT Front Tracking method. 13

FTLE Finite-Time Lyapunov Exponent. 6, 14, 66, 67, 69, 70, 71, 72, 74

FVM Finite Volume Method. 28, 30, 73

LCS Lagrangian Coherent Structures. 6, 14, 66, 68, 69, 70, 71, 74

LS Level-Set method. 13

NDA Neumann-Dirichlet Algorithm. 37, 42, 43

OOP Object-Oriented Programming. 47

PISO Pressure-Implicit with Splitting of Operators. 35

SCL	Space (Geometric) Conservation law. 29, 30, 32
SGS	Subgrid-Scale model. 64, 65, 74
SIMPLE	Semi-Implicit Method for Pressure Linked Equations. 35
surfactant	surface-active agent. 7, 18, 25, 26
UML	Unified Modeling Language. 6, 47, 49, 50, 52, 53, 54
VOF	Volume-Of-Fluid method. 13, 63

Bibliography

[1] A. Alke and D. Bothe. “3D Numerical Modeling of Soluble Surfactant at Fluidic Interfaces Based on the Volume-of-Fluid Method”. In: *Fluid Dynamics and Materials Processing* 5 (Jan. 2009).

[2] Holm Altenbach. *Kontinuumsmechanik: Einführung in die materialunabhängigen und materialabhängigen Gleichungen*. ger. 3., überarbeitete Auflage. Berlin Heidelberg: Springer Vieweg, 2015. ISBN: 978-3-662-47069-5. DOI: 10.1007/978-3-662-47070-1.

[3] *ANSYS FLUENT 12.0 User’s Guide - 26.13.1 Monitoring Residuals*. URL: https://www.afs.enea.it/project/neptunius/docs/fluent/html/ug/node812.htm#uns-def-resid-eq4 (visited on 08/31/2022).

[4] Sanjeeva Balasuriya, Nicholas T. Ouellette, and Irina I. Rypina. “Generalized Lagrangian coherent structures”. en. In: *Physica D: Nonlinear Phenomena* 372 (June 2018), pp. 31–51. ISSN: 01672789. DOI: 10.1016/j.physd.2018.01.011. URL: https://linkinghub.elsevier.com/retrieve/pii/S0167278917302750 (visited on 08/03/2022).

[5] Martin Beaudoin and Hrvoje Jasak. “Development of a Generalized Grid Interface for Turbomachinery simulations with OpenFOAM”. In: (Jan. 2008).

[6] M. Becker et al. “Mehrphasenreaktoren: Zusammenspiel von Prozessentwicklung und Hydrodynamik”. en. In: *Chemie Ingenieur Technik* 84.8 (Aug. 2012), pp. 1223–1223. ISSN: 0009286X. DOI: 10.1002/cite.201250497. URL: https://onlinelibrary.wiley.com/doi/10.1002/cite.201250497 (visited on 06/13/2022).

[7] Robert Byron Bird, Warren E. Stewart, and Edwin N. Lightfoot. *Transport phenomena*. eng. Revised ed. New York: Wiley, 2007. ISBN: 978-0-470-11539-8.

[8] D. Bothe, J. Prüss, and G. Simonett. “Well-posedness of a Two-phase Flow with Soluble Surfactant”. en. In: *Nonlinear Elliptic and Parabolic Problems*. Ed. by Haim Brezis, Michel Chipot, and Joachim Escher. Vol. 64. Series Title: Progress in Nonlinear Differential Equations and Their Applications. Basel: Birkhäuser-Verlag, 2005, pp. 37–61. ISBN: 978-3-7643-7266-8. DOI: 10.1007/3-7643-7385-7_3. URL: http://link.springer.com/10.1007/3-7643-7385-7_3 (visited on 07/12/2022).

[9] Dieter Bothe and Stefan Fleckenstein. “A Volume-of-Fluid-based method for mass transfer processes at fluid particles”. en. In: *Chemical Engineering Science* 101 (Sept. 2013), pp. 283–302. ISSN: 00092509. DOI: 10.1016/j.ces.2013.05.029. URL: https://linkinghub.elsevier.com/retrieve/pii/S0009250913003606 (visited on 07/14/2022).

[10] H.-J. Bungartz and M. Schäfer, eds. *Fluid-structure interaction: modelling, simulation, optimisation*. Lecture notes in computational science and engineering 53. OCLC: ocm71798417. Berlin ; New York: Springer-Verlag, 2006. ISBN: 978-3-540-34595-4.

[11] P. Cardiff et al. “An open-source finite volume toolbox for solid mechanics and fluid-solid interaction simulations”. In: (2018). Publisher: arXiv Version Number: 2. DOI: 10.48550/ARXIV.1808.10736. URL: https://arxiv.org/abs/1808.10736 (visited on 05/23/2022).

[12] Paolo Cermelli, Eliot Fried, and Morton E. Gurtin. “Transport relations for surface integrals arising in the formulation of balance laws for evolving fluid interfaces”. en. In: *Journal of Fluid Mechanics* 544.-1 (Nov. 2005), p. 339. ISSN: 0022-1120, 1469-7645. DOI: 10.1017/S0022112005006695. URL: http://www.journals.cambridge.org/abstract_S0022112005006695 (visited on 09/05/2022).

[13] Kang Ping Chen, William Saric, and Howard A. Stone. “On the deviatoric normal stress on a slip surface”. en. In: *Physics of Fluids* 12.12 (Dec. 2000), pp. 3280–3281. ISSN: 1070-6631, 1089-7666. DOI: 10.1063/1.1321259. URL: http://aip.scitation.org/doi/10.1063/1.1321259 (visited on 05/23/2022).

[14] M. Darwish, I. Sraj, and F. Moukalled. “A coupled finite volume solver for the solution of incompressible flows on unstructured grids”. en. In: *Journal of Computational Physics* 228.1 (Jan. 2009), pp. 180–201. ISSN: 00219991. DOI: 10.1016/j.jcp.2008.08.027. URL: https://linkinghub.elsevier.com/retrieve/pii/S0021999108004725 (visited on 05/23/2022).

[15] Michael A. Day. “The no-slip condition of fluid dynamics”. en. In: *Erkenntnis* 33.3 (Nov. 1990), pp. 285–296. ISSN: 0165-0106, 1572-8420. DOI: 10.1007/BF00717588. URL: http://link.springer.com/10.1007/BF00717588 (visited on 07/17/2022).

[16] Daniel Deising. “Modelling and Numerical Simulation of Species Transfer in Bubbly Flows using OpenFOAM”. en. PhD Thesis. Darmstadt: Technische Universität, Feb. 2019. URL: http://tuprints.ulb.tu-darmstadt.de/8522/.

[17] I. Demirdžić and M. Perić. “Space conservation law in finite volume calculations of fluid flow”. en. In: *International Journal for Numerical Methods in Fluids* 8.9 (Sept. 1988), pp. 1037–1050. ISSN: 0271-2091, 1097-0363. DOI: 10.1002/fld.1650080906. URL: https://onlinelibrary.wiley.com/doi/10.1002/fld.1650080906 (visited on 07/12/2022).

[18] Kathrin Dieter-Kissling, Holger Marschall, and Dieter Bothe. “Direct Numerical Simulation of droplet formation processes under the influence of soluble surfactant mixtures”. en. In: *Computers & Fluids* 113 (May 2015), pp. 93–105. ISSN: 00457930. DOI: 10.1016/j.compfluid.2015.01.017. URL: https://linkinghub.elsevier.com/retrieve/pii/S0045793015000262 (visited on 07/14/2022).

[19] Victorita Dolean, Pierre Jolivet, and Frédéric Nataf. *An introduction to domain decomposition methods: algorithms, theory, and parallel implementation*. Philadelphia: Society for Industrial and Applied Mathematics, 2015. ISBN: 978-1-61197-405-8.

[20] P. C. Duineveld. “The rise velocity and shape of bubbles in pure water at high Reynolds number”. en. In: *Journal of Fluid Mechanics* 292 (June 1995), pp. 325–332. ISSN: 0022-1120, 1469-7645. DOI: 10.1017/S0022112095001546. URL: https://www.cambridge.org/core/product/identifier/S0022112095001546/type/journal_article (visited on 07/06/2022).

[21] Gabriel G.S. Ferreira et al. “Implementation of an implicit pressure–velocity coupling for the Eulerian multi-fluid model”. en. In: *Computers & Fluids* 181 (Mar. 2019), pp. 188–207. ISSN: 00457930. DOI: 10.1016/j.compfluid.2019.01.018. URL: https://linkinghub.elsevier.com/retrieve/pii/S0045793019300143 (visited on 05/23/2022).

[22] Joel H. Ferziger, Milovan Perić, and Robert L. Street. *Computational Methods for Fluid Dynamics*. en. Cham: Springer International Publishing, 2020. ISBN: 978-3-319-99691-2 978-3-319-99693-6. DOI: 10.1007/978-3-319-99693-6. URL: http://link.springer.com/10.1007/978-3-319-99693-6 (visited on 05/23/2022).

[23] Adolph Fick. "V. *On liquid diffusion*". en. In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 10.63 (July 1855), pp. 30–39. ISSN: 1941-5982, 1941-5990. DOI: 10.1080/14786445508641925. URL: https://www.tandfonline.com/doi/full/10.1080/14786445508641925 (visited on 07/12/2022).

[24] Justin Finn and Sourabh V. Apte. "Integrated computation of finite-time Lyapunov exponent fields during direct numerical simulation of unsteady flows". en. In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 23.1 (Mar. 2013), p. 013145. ISSN: 1054-1500, 1089-7682. DOI: 10.1063/1.4795749. URL: http://aip.scitation.org/doi/10.1063/1.4795749 (visited on 06/29/2022).

[25] William Gropp et al. "A high-performance, portable implementation of the MPI message passing interface standard". en. In: *Parallel Computing* 22.6 (Sept. 1996), pp. 789–828. ISSN: 01678191. DOI: 10.1016/0167-8191(96)00024-5. URL: https://linkinghub.elsevier.com/retrieve/pii/0167819196000245 (visited on 09/12/2022).

[26] Alireza Hadjighasem et al. "A critical comparison of Lagrangian methods for coherent structure detection". en. In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 27.5 (May 2017), p. 053104. ISSN: 1054-1500, 1089-7682. DOI: 10.1063/1.4982720. URL: http://aip.scitation.org/doi/10.1063/1.4982720 (visited on 06/29/2022).

[27] Hidajet Hadzić. "Development and Application of Finite Volume Method for the Computation of Flows Around Moving Bodies on Unstructured, Overlapping Grids". PhD thesis. 2006.

[28] G. Haller. "Lagrangian coherent structures from approximate velocity data". en. In: *Physics of Fluids* 14.6 (June 2002), pp. 1851–1861. ISSN: 1070-6631, 1089-7666. DOI: 10.1063/1.1477449. URL: http://aip.scitation.org/doi/10.1063/1.1477449 (visited on 07/02/2022).

[29] George Haller. "Lagrangian Coherent Structures". en. In: *Annual Review of Fluid Mechanics* 47.1 (Jan. 2015), pp. 137–162. ISSN: 0066-4189, 1545-4479. DOI: 10.1146/annurev-fluid-010313-141322. URL: https://www.annualreviews.org/doi/10.1146/annurev-fluid-010313-141322 (visited on 07/02/2022).

[30] Charles Hirsch. *Numerical computation of internal and external flows: fundamentals of computational fluid dynamics*. 2nd ed. OCLC: ocn148277909. Oxford ; Burlington, MA: Elsevier/Butterworth-Heinemann, 2007. ISBN: 978-0-7506-6594-0.

[31] C. W. Hirt, A. A. Amsden, and J. L. Cook. "An arbitrary Lagrangian-Eulerian computing method for all flow speeds". en. In: *Journal of Computational Physics* 14.3 (Mar. 1974), pp. 227–253. ISSN: 00219991. DOI: 10.1016/0021-9991(74)90051-5. URL: https://linkinghub.elsevier.com/retrieve/pii/0021999174900515 (visited on 07/15/2022).

[32] Hrvoje Jasak. "Abstract for the 5th OpenFOAM User Conference 2017 - Porting of the Finite Area Method to OpenFOAM Plus". In: Wiesbaden, 2017. URL: https://www.esi-group.com/sites/default/files/resource/other/2113/abstract-jasak-wikki-porting-of-the-finite-are-method-to-openfoam-plus.pdf (visited on 07/28/2022).

[33] M. Ishii and T. Hibiki. *Thermo-fluid dynamics of two-phase flow*. OCLC: ocm63789873. New York, N.Y: Springer Science+Business Media, 2006. ISBN: 978-0-387-28321-0 978-0-387-29187-1.

[34] R. I. Issa. "Solution of the implicitly discretised fluid flow equations by operator-splitting". en. In: *Journal of Computational Physics* 62.1 (Jan. 1986), pp. 40–65. ISSN: 00219991. DOI: 10.1016/0021-9991(86)90099-9. URL: https://linkinghub.elsevier.com/retrieve/pii/0021999186900999 (visited on 08/02/2022).

[35] James Casey. "On the derivation of jump conditions in continuum mechanics". en. In: *International Journal of Structural Changes in Solids – Mechanics and Applications* 3.2 (2011), pp. 61–84. URL: https://ijscs-ojs-tamu.tdl.org/ijscs/article/view/2420 (visited on 09/05/2022).

[36] Hrvoje Jasak. "Error Analysis and Estimation for the Finite Volume Method With Applications to Fluid Flows". PhD thesis. Jan. 1996.

[37] Hrvoje Jasak and Zeljko Tukovic. "Automatic mesh motion for the unstructured Finite Volume Method". In: *Transactions of FAMENA* (Nov. 2013).

[38] F. Juretic and A. D. Gosman. "Error Analysis of the Finite-Volume Method with Respect to Mesh Type". In: *Numerical Heat Transfer Part B-fundamentals* 57 (June 2010), pp. 414–439. DOI: 10.1080/10407791003685155.

[39] Justin Finn, Romain Watteaux, and Andrew Lawrie. *libcfd2lcs: A general purpose library for computation of Lagrangian coherent structures during CFD simulation*. English. Feb. 2017. URL: https://www.archer.ac.uk/training/virtual/2016-06-22-libcfd2lcs/ArcherVT.pdf (visited on 06/29/2022).

[40] A.v. Kameke et al. "How coherent structures dominate the residence time in a bubble wake: An experimental example". en. In: *Chemical Engineering Science* 207 (Nov. 2019), pp. 317–326. ISSN: 00092509. DOI: 10.1016/j.ces.2019.06.033. URL: https://linkinghub.elsevier.com/retrieve/pii/S0009250919305366 (visited on 06/29/2022).

[41] David E. Keyes et al. "Multiphysics simulations: Challenges and opportunities". en. In: *The International Journal of High Performance Computing Applications* 27.1 (Feb. 2013), pp. 4–83. ISSN: 1094-3420, 1741-2846. DOI: 10.1177/1094342012468181. URL: http://journals.sagepub.com/doi/10.1177/1094342012468181 (visited on 05/23/2022).

[42] Nikolay Ivanov Kolev. *Multiphase flow dynamics*. 3rd. ed. Berlin ; New York: Springer, 2007. ISBN: 978-3-540-69832-6 978-3-540-69834-0.

[43] Konrad Königsberger. *Analysis 2*. Springer-Lehrbuch. Berlin, Heidelberg: Springer Berlin Heidelberg, 1997. ISBN: 978-3-540-62871-2 978-3-662-05700-1. DOI: 10.1007/978-3-662-05700-1. URL: http://link.springer.com/10.1007/978-3-662-05700-1 (visited on 06/28/2022).

[44] Ulrich Küttler, Christiane Förster, and Wolfgang A. Wall. "A Solution for the Incompressibility Dilemma in Partitioned Fluid–Structure Interaction with Pure Dirichlet Fluid Domains". en. In: *Computational Mechanics* 38.4-5 (Sept. 2006), pp. 417–429. ISSN: 0178-7675, 1432-0924. DOI: 10.1007/s00466-006-0066-5. URL: https://link.springer.com/10.1007/s00466-006-0066-5 (visited on 05/23/2022).

[45] Shingyu Leung. "An Eulerian approach for computing the finite time Lyapunov exponent". en. In: *Journal of Computational Physics* 230.9 (May 2011), pp. 3500–3524. ISSN: 00219991. DOI: 10.1016/j.jcp.2011.01.046. URL: https://linkinghub.elsevier.com/retrieve/pii/S0021999111000799 (visited on 06/29/2022).

[46] Doug Lipinski and Kamran Mohseni. "A ridge tracking algorithm and error estimate for efficient computation of Lagrangian coherent structures". en. In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 20.1 (Mar. 2010), p. 017504. ISSN: 1054-1500, 1089-7682. DOI: 10.1063/1.3270049. URL: http://aip.scitation.org/doi/10.1063/1.3270049 (visited on 06/29/2022).

[47] R.D. Lonsdale. "An algebraic multigrid solver for the Navier-Stokes equations on unstructured meshes". en. In: *International Journal of Numerical Methods for Heat & Fluid Flow* 3.1 (Jan. 1993), pp. 3–14. ISSN: 0961-5539. DOI: 10.1108/eb017512. URL: https://www.emerald.com/insight/content/doi/10.1108/eb017512/full/html (visited on 08/03/2022).

[48] L. Mangani, M. Buchmayr, and M. Darwish. “Development of a Novel Fully Coupled Solver in OpenFOAM: Steady-State Incompressible Turbulent Flows”. en. In: *Numerical Heat Transfer, Part B: Fundamentals* 66.1 (July 2014), pp. 1–20. ISSN: 1040-7790, 1521-0626. DOI: 10.1080/10407790.2014.894448. URL: http://www.tandfonline.com/doi/abs/10.1080/10407790.2014.894448 (visited on 08/03/2022).

[49] Holger Marschall. *Arbitrary Lagrangian Eularian (ALE) Interface Tracking Method for Unstrucktured Meshes of General Topology*. Lecture Notes. Darmstadt, July 2020.

[50] John G. Michopoulos, Charbel Farhat, and Jacob Fish. “Modeling and Simulation of Multiphysics Systems”. en. In: *Journal of Computing and Information Science in Engineering* 5.3 (2005), p. 198. ISSN: 15309827. DOI: 10.1115/1.2031269. URL: http://ComputingEngineering.asmedigitalcollection.asme.org/article.aspx?articleid=1400266 (visited on 05/23/2022).

[51] Dimitri Mihalas and Barbara Weibel-Mihalas. *Foundations of radiation hydrodynamics*. Mineola, N.Y: Dover, 1999. ISBN: 978-0-486-40925-2.

[52] Reinhard Miller and Valentin B. Fainerman. “Surfactant adsorption layers at liquid-fluid interfaces”. en. In: *Handbook of Surfaces and Interfaces of Materials*. Elsevier, 2001, pp. 383–421. ISBN: 978-0-12-513910-6. DOI: 10.1016/B978-012513910-6/50013-X. URL: https://linkinghub.elsevier.com/retrieve/pii/B978012513910650013X (visited on 07/21/2022).

[53] Azahar Monge and Philipp Birken. “On the convergence rate of the Dirichlet–Neumann iteration for unsteady thermal fluid–structure interaction”. en. In: *Computational Mechanics* 62.3 (Sept. 2018), pp. 525–541. ISSN: 0178-7675, 1432-0924. DOI: 10.1007/s00466-017-1511-3. URL: http://link.springer.com/10.1007/s00466-017-1511-3 (visited on 05/23/2022).

[54] F. Moukalled, L. Mangani, and M. Darwish. *The Finite Volume Method in Computational Fluid Dynamics: An Advanced Introduction with OpenFOAM® and Matlab*. en. Vol. 113. Fluid Mechanics and Its Applications. Cham: Springer International Publishing, 2016. ISBN: 978-3-319-16873-9 978-3-319-16874-6. DOI: 10.1007/978-3-319-16874-6. URL: http://link.springer.com/10.1007/978-3-319-16874-6 (visited on 05/23/2022).

[55] S. Muzaferija and M. Peri´c. “Computation of free-surface flows using the finite- volume method and moving grids”. en. In: *Numerical Heat Transfer, Part B: Fundamentals* 32.4 (Dec. 1997), pp. 369–384. ISSN: 1040-7790, 1521-0626. DOI: 10.1080/10407799708915014. URL: http://www.tandfonline.com/doi/abs/10.1080/10407799708915014 (visited on 07/12/2022).

[56] Samir Muzaferija and M Peri‘c. *Computation of free-surface flows using interface-tracking and interface-capturing methods.* Jan. 1999.

[57] Witold Nowacki. *Thermoelasticity*. 2nd ed., rev. and enl. Oxford ; New York : Warszawa: Pergamon Press ; PWN-Polish Scientific Publishers, 1986. ISBN: 978-0-08-024767-0.

[58] “Reactive Flows”. en. In: *Prandtl’s Essentials of Fluid Mechanics*. Ed. by Herbert Oertel. Vol. 158. Series Title: Applied Mathematical Sciences. New York: Springer-Verlag, 2004, pp. 503–570. ISBN: 978-0-387-40437-0. DOI: 10.1007/0-387-21803-3_11. URL: http://link.springer.com/10.1007/0-387-21803-3_11 (visited on 06/14/2022).

[59] *OpenFOAM: User Guide: Residuals*. URL: https://www.openfoam.com/documentation/guides/latest/doc/guide-solvers-residuals.html (visited on 08/31/2022).

[60] Suhas Patankar. *Numerical Heat Transfer and Fluid Flow*. English. OCLC: 1140143078. 2018. ISBN: 978-1-4822-3421-3 978-1-315-27513-0 978-1-351-99151-3. URL: http://search.ebscohost.com/login.aspx?direct=true&scope=site&db=nlebk&db=nlabk&AN=2026360 (visited on 07/28/2022).

[61] Chiara Pesci et al. “Computational analysis of single rising bubbles influenced by soluble surfactant”. en. In: *Journal of Fluid Mechanics* 856 (Dec. 2018), pp. 709–763. ISSN: 0022-1120, 1469-7645. DOI: 10.1017/jfm.2018.723. URL: https://www.cambridge.org/core/product/identifier/S0022112018007231/type/journal_article (visited on 07/12/2022).

[62] Chiara Pesci et al. “Finite Volume/Finite Area Interface-Tracking Method for Two-Phase Flows with Fluid Interfaces Influenced by Surfactant”. In: Dec. 2015, pp. 373–409. ISBN: 978-1-4987-2208-7. DOI: 10.1201/b19337-22.

[63] C. M. Rhie and W. L. Chow. “Numerical study of the turbulent flow past an airfoil with trailing edge separation”. en. In: *AIAA Journal* 21.11 (Nov. 1983), pp. 1525–1532. ISSN: 0001-1452, 1533-385X. DOI: 10.2514/3.8284. URL: https://arc.aiaa.org/doi/10.2514/3.8284 (visited on 08/01/2022).

[64] Akshai K. Runchal. “Emergence of Computational Fluid Dynamics at Imperial College (1965–1975): A Personal Recollection”. en. In: *Journal of Heat Transfer* 135.1 (Jan. 2013), p. 011009. ISSN: 0022-1481, 1528-8943. DOI: 10.1115/1.4007655. URL: https://asmedigitalcollection.asme.org/heattransfer/article/doi/10.1115/1.4007655/366475/Emergence-of-Computational-Fluid-Dynamics-at (visited on 05/23/2022).

[65] Y. Saad. *Iterative methods for sparse linear systems*. 2nd ed. Philadelphia: SIAM, 2003. ISBN: 978-0-89871-534-7.

[66] H. Schlichting and K. Gersten. *Grenzschicht-Theorie*. English. OCLC: 1315800127. 2006. ISBN: 978-3-540-23004-5.

[67] Michael Schlüter et al. “Small-Scale Phenomena in Reactive Bubbly Flows: Experiments, Numerical Modeling, and Applications”. en. In: *Annual Review of Chemical and Biomolecular Engineering* 12.1 (June 2021), pp. 625–643. ISSN: 1947-5438, 1947-5446. DOI: 10.1146/annurev-chembioeng-092220-100517. URL: https://www.annualreviews.org/doi/10.1146/annurev-chembioeng-092220-100517 (visited on 06/13/2022).

[68] David P. Schmidt et al. “Direct interface tracking of droplet deformation”. en. In: *Atomization and Sprays* 12.5-6 (2002), pp. 721–736. ISSN: 1045-5110. DOI: 10.1615/AtomizSpr.v12.i56.110. URL: http://www.dl.begellhouse.com/journals/6a7c7e10642258cc,05c0147b0b2aa305,7d4d29673be61d90.html (visited on 09/08/2022).

[69] *See the MRF development - OpenFOAMWiki*. URL: https://openfoamwiki.net/index.php/See_the_MRF_development#Navier-Stokes_equations_in_the_relative_frame_with_absolute_velocity (visited on 09/01/2022).

[70] Shawn C. Shadden, Francois Lekien, and Jerrold E. Marsden. “Definition and properties of Lagrangian coherent structures from finite-time Lyapunov exponents in two-dimensional aperiodic flows”. en. In: *Physica D: Nonlinear Phenomena* 212.3-4 (Dec. 2005), pp. 271–304. ISSN: 01672789. DOI: 10.1016/j.physd.2005.10.007. URL: https://linkinghub.elsevier.com/retrieve/pii/S0167278905004446 (visited on 07/02/2022).

[71] Sheng Xu. “Derivation of principal jump conditions for the immersed interface method in two-fluid flow simulation”. In: *Conference Publications 2009*. AIMS Press, 2009. ISBN: 978-1-60133-007-9. DOI: 10.3934/proc.2009.2009.838. URL: http://aimsciences.org/article/doi/10.3934/proc.2009.2009.838 (visited on 05/23/2022).

[72] John C. Slattery. *Advanced Transport Phenomena*. 1st ed. Cambridge University Press, June 1999. ISBN: 978-0-521-63203-4 978-0-521-63565-3 978-0-511-80023-8. DOI: 10.1017/CBO9780511800238. URL: https://www.cambridge.org/core/product/identifier/9780511800238/type/book (visited on 06/28/2022).

[73] John C. Slattery, Leonard M. Sagis, and Eun-Suok Oh. *Interfacial transport phenomena*. 2nd ed. New York: Springer, 2007. ISBN: 978-0-387-38438-2 978-0-387-38442-9.

[74] Tar Solomon and J. Gollub. "Chaotic Transport in Time-Dependent Rayleigh-Bénard Convection". In: *Physical review. A* 38 (Jan. 1989), pp. 6280–6286. DOI: 10.1103/PhysRevA.38.6280.

[75] Joseph H. Spurk and Nuri Aksel. *Strömungslehre Einführung in die Theorie der Strömungen*. English. OCLC: 1181186590. 2006. ISBN: 978-3-540-29712-3.

[76] Matthias Steinhausen. *Numerical simulation of single rising bubbles influenced by soluble surfactant in the spherical and ellipsoidal regime*. Darmstadt, 2018. URL: http://tuprints.ulb.tu-darmstadt.de/8296/.

[77] H. A. Stone. "A simple derivation of the time-dependent convective-diffusion equation for surfactant transport along a deforming interface". en. In: *Physics of Fluids A: Fluid Dynamics* 2.1 (Jan. 1990), pp. 111–112. ISSN: 0899-8213. DOI: 10.1063/1.857686. URL: http://aip.scitation.org/doi/10.1063/1.857686 (visited on 07/14/2022).

[78] Bjarne Stroustrup. *The C++ programming language*. 3rd ed. Reading, Mass: Addison-Wesley, 1997. ISBN: 978-0-201-88954-3.

[79] Mohamed H. M. Sulman et al. "Leaving flatland: Diagnostics for Lagrangian coherent structures in three-dimensional flows". en. In: *Physica D: Nonlinear Phenomena* 258 (Sept. 2013), pp. 77–92. ISSN: 01672789. DOI: 10.1016/j.physd.2013.05.005. URL: https://linkinghub.elsevier.com/retrieve/pii/S0167278913001450 (visited on 06/29/2022).

[80] Andrea Toselli and Olof B. Widlund. "Substructuring Methods: Introduction". In: *Domain Decomposition Methods — Algorithms and Theory*. Vol. 34. Series Title: Springer Series in Computational Mathematics. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 87–112. ISBN: 978-3-540-20696-5 978-3-540-26662-4. DOI: 10.1007/3-540-26662-3_4. URL: http://link.springer.com/10.1007/3-540-26662-3_4 (visited on 08/15/2022).

[81] Ž. Tuković and H. Jasak. "A moving mesh finite volume interface tracking method for surface tension dominated interfacial fluid flow". en. In: *Computers & Fluids* 55 (Feb. 2012), pp. 70–84. ISSN: 00457930. DOI: 10.1016/j.compfluid.2011.11.003. URL: https://linkinghub.elsevier.com/retrieve/pii/S0045793011003380 (visited on 05/23/2022).

[82] Željko Tuković et al. "OpenFOAM Finite Volume Solver for Fluid-Solid Interaction". In: *Transactions of FAMENA* 42.3 (Oct. 2018), pp. 1–31. ISSN: 13331124, 18491391. DOI: 10.21278/TOF.42301. URL: http://hrcak.srce.hr/206941 (visited on 05/23/2022).

[83] H. K. Versteeg and W. Malalasekera. *An introduction to computational fluid dynamics: the finite volume method*. 2nd ed. OCLC: ocm76821177. Harlow, England ; New York: Pearson Education Ltd, 2007. ISBN: 978-0-13-127498-3.

[84] Vuko Vukcevic et al. "Development of a CFD Solver for Primary Diesel Jet Atomization in FOAM-Extend". In: Sept. 2019, pp. 2019–24–0128. DOI: 10.4271/2019-24-0128. URL: https://www.sae.org/content/2019-24-0128/ (visited on 05/23/2022).

[85] Paul S. Weber. "Modeling and numerical simulation of multi-component single- and two-phase fluid systems". en. PhD Thesis. Darmstadt: TU Darmstadt, May 2017. URL: http://tubiblio.ulb.tu-darmstadt.de/87245/.

[86] Paul S. Weber, Holger Marschall, and Dieter Bothe. “Highly accurate two-phase species transfer based on ALE Interface Tracking”. en. In: *International Journal of Heat and Mass Transfer* 104 (Jan. 2017), pp. 759–773. ISSN: 00179310. DOI: 10.1016/j.ijheatmasstransfer.2016.08.072. URL: https://linkinghub.elsevier.com/retrieve/pii/S0017931015316744 (visited on 05/23/2022).

[87] R. Webster. “An algebraic multigrid solver for Navier-Stokes problems in discete second-order approximation”. en. In: *International Journal for Numerical Methods in Fluids* 22.11 (June 1996), pp. 1103–1123. ISSN: 0271-2091, 1097-0363. DOI: 10.1002/(SICI)1097-0363(19960615)22:11<1103::AID-FLD406>3.0.CO;2-K. URL: https://onlinelibrary.wiley.com/doi/10.1002/(SICI)1097-0363(19960615)22:11%3C1103::AID-FLD406%3E3.0.CO;2-K (visited on 08/03/2022).

[88] Andre Weiner and Dieter Bothe. “Advanced subgrid-scale modeling for convection-dominated species transport at fluid interfaces with application to mass transfer from rising bubbles”. en. In: *Journal of Computational Physics* 347 (Oct. 2017), pp. 261–289. ISSN: 00219991. DOI: 10.1016/j.jcp.2017.06.040. URL: https://linkinghub.elsevier.com/retrieve/pii/S0021999117304941 (visited on 09/27/2022).

[89] H. G. Weller et al. “A tensorial approach to computational continuum mechanics using object-oriented techniques”. en. In: *Computers in Physics* 12.6 (1998), p. 620. ISSN: 08941866. DOI: 10.1063/1.168744. URL: http://scitation.aip.org/content/aip/journal/cip/12/6/10.1063/1.168744 (visited on 05/23/2022).

[90] Yuri B. Zudin. *Theory of periodic conjugate heat transfer*. OCLC: ocn124038247. Berlin ; New York: Springer, 2007. ISBN: 978-3-540-70723-3.

Acknowledgements

First of all I would like to thank my supervising Professor Prof. Dr. Dieter Bothe for the opportunity to do my master's thesis at Mathematical Modelling and Analysis institute at TU Darmstadt. A special thanks goes to my supervisor Dr. Holger Marschall who gave me excellent support throughout the entire process and helped to drive this work forward through a multitude of great scientific discussions and ideas. Also I, would like to thank all members of the Computational Multiphase Flow Group. They made me feel very welcome in their team and I was able to reach out to them at all times when I had any questions.

Furthermore, I would like to thank my family and my girlfriend Marie, for their everlasting support and understanding.

Additionally, I gratefully acknowledge the computing time provided to me on the high-performance computer Lichtenberg at the NHR Centers NHR4CES at TU Darmstadt.