Imperial College London
Department of Mechanical Engineering
Exhibition Road
London, SW7 2AZ

Balanced-Force Two-Phase Flow Modelling on Unstructured and Adaptive Meshes

Fabian Denner
Dipl.-Ing.

November 2013

Submitted in part fulfilment of the requirements for the degree of
Doctor of Philosophy in Mechanical Engineering of Imperial College London
and the Diploma of Imperial College London

Declaration of Originality

I declare that the work presented in this thesis is my own and expressed in my own words except where declared otherwise. Any work used by any other author(s) in any form has been properly acknowledged where used. A list of all the references used has been included. This work has not already been submitted for any other degree and is not being submitted concurrently for any other degree.

London, November 2013
Fabian Denner

Copyright Declaration

London, November 2013
Fabian Denner

Abstract

Two-phase flows occur regularly in nature and industrial processes and their understanding is of significant interest in engineering research and development. Various numerical methods to predict two-phase phase flows have been developed as a result of extensive research efforts in past decades, however, most methods are limited to Cartesian meshes.

A fully-coupled implicit numerical framework for two-phase flows on unstructured meshes is presented, solving the momentum equations and a specifically constructed continuity constraint in a single equation system. The continuity constraint, derived using a momentum interpolation method, satisfies continuity, provides a strong pressure-velocity coupling and ensures a discrete balance between pressure gradient and body forces. The numerical framework is not limited to specific density ratios or a particular interface topology and includes several novelties.

A further step towards a more accurate prediction of two-phase flows on unstructured meshes is taken by proposing a new method to evaluate the interface curvature. The curvature estimates obtained with this new method are shown to be as good as or better than methods reported in literature, which are mostly limited to Cartesian meshes, and the accuracy on structured and unstructured meshes is shown to be comparable. Furthermore, lasting contributions are made towards the understanding of convolution methods for two-phase flow modelling and the underlying mechanisms of parasitic currents are studied in detailed.

The mesh resolution is of particular importance for two-phase flows due to the inherent first-order accuracy of the interface position using interface capturing methods. A mesh adaption algorithm for tetrahedral meshes with application to two-phase flows and its implementation are presented. The algorithm is applied to study mesh resolution requirements at interfaces and force-balancing for surface-tension-dominated two-phase flows on adaptive meshes.

Related Publications

Journal Papers

- Denner, F. and van Wachem, B.G.M.: On the convolution of fluid properties and surface force for interface capturing methods. International Journal of Multiphase Flow, 54 (2013), pp. 61-64.
- Denner, F. and van Wachem, B.G.M.: Fully-coupled balanced-force VOF framework for arbitrary meshes with least-squares curvature evaluation from volume fraction. Numerical Heat Transfer, Part B: Fundamentals, accepted for publication. DOI: 10.1080/10407790.2013.849996
- Denner, F., van der Heul, D., Oud, G.T., Martins Villar, M., da Silveira Neto, A. and van Wachem, B.G.M.: Comparative study of mass-conserving interface capturing frameworks for two-phase flows with surface tension. Submitted to International Journal of Multiphase Flows on 27 June 2013.

Conference Proceedings

- Denner, F. and van Wachem, B.G.M.: Two-Phase Flow Modelling on Arbitrary Meshes: Superior VOF Curvature Estimation and the Issue of Convolution. International Conference on Numerical Methods in Multiphase Flows, 12 - 14 June 2012, State College, PA, USA.
- Denner, F. and van Wachem, B.G.M.: Force-balancing at moving surface-tension-dominated interfaces on collocated unstructured meshes. 8th International Conference on Multiphase Flow (ICMF 2013), 26 - 31 May 2013, Jeju, Korea.

Acknowledgements

First and foremost, I would like to express my deep and sincere gratitude to my supervisor Dr. Berend van Wachem for his continuous support, his encouragement and his guidance throughout my PhD studies. He has been an invaluable source of knowledge and inspiration and it has been a great pleasure working with him.

I would like to thank Dr. Duncan van der Heul at Delft Technical University as well as Dr. Millena Martins Villar and Rodrigo Lisita at the Federal University Uberlandia for the excellent research cooperation. Furthermore, I would like to thank Prof. William P. Jones for his support and the interesting discussions over the years.

A special and wholehearted thanks goes to Dr. George Mallouppas, Dr. Stefan Wysocki, Fabio Di Salvo, Dr. Nakul Prasad, Mohamad Azlin Ali, Timothy Brauner and Martin Jurisch, who were not just colleagues but quickly became valued friends, for sharing the ups and downs of working towards a PhD, for the eye-watering laughter and the occasional beverage we have enjoyed together - you have made my time as a PhD student an unforgettable experience. I would also like to thank all my other colleagues whom I shared *Room 600* with throughout my PhD studies for interesting and entertaining discussions and their companionship in our office as well as Thomas Bangalter and Guy-Manuel de Homem-Christo, whose music carried me through the toughest and most frustrating hours of debugging.

I would also like to extend my gratitude to people who played an important role over the years in my development as a researcher and engineer. To Elmar Beeh and Roland Schöll at the German Aerospace Center (DLR) for their mentoring and for the opportunities they gave me as an undergraduate student. To Dr. Fabian Brännström at Porsche AG (now at Bombardier Transportation GmbH) for teaching me a sceptical view on CFD, which proves to be a valuable asset for my research. To Jürgen Unger, who sparked my interest for mathematics back when I was a teenager and who has been a close friend and valuable source of knowledge and enthusiasm for more than twenty years.

Last but by no means least, I would like to wholeheartedly thank my parents for their unconditional love and support and for being a seemingly endless source of encouragement and inspiration. They have been an invaluable part of my life and I will always be indebted to them.

The financial support of the Swedish Research Council (VR) and the Engineering and Physical Sciences Research Council (EPSRC) is gratefully acknowledged.

To my father - my greatest inspiration.

Dieter Denner
(1955-2008)

No problem can withstand the assault of sustained thinking.
- Voltaire

Contents

List of Figures

List of Tables

Nomenclature

Vectors and tensors are either denoted in **bold** or in tensor notation by subscripts i (vector) or ij (tensor), dependent on the context. All units are SI units.

Roman Letters

$\dot{m}$	Mass flux	$[kg\, s^{-1}]$
$\hat{d}$	Coefficient of the momentum interpolation method	$[m^3\, s\, kg^{-1}]$
$\tilde{r}$	Radial coordinate	$[m]$
$\tilde{u}$	Spatial momentum velocity	$[m\, s^{-1}]$
A	Area	$[m^2]$
A	Coefficient defining the type of boundary condition (only in Section 3.5)	
A	Constant that normalises the convolution kernel (only in Section 4.2.1)	
a	Combined implicit coefficient of the convective and viscous terms of the discretised momentum equation	$[kg\, s^{-1}]$
A_{ij}, $\boldsymbol{A}$	Coefficient matrix of an equation system	
a_i, $\boldsymbol{a}$	Co-variant vector	$[m]$
a_r	Length of a liquid inclusion along its radial axis	$[m]$
a_y	Length of a liquid inclusion along its symmetry axis	$[m]$
B	Coefficient defining the type of boundary condition (only in Section 3.5)	
B_{ij}, $\boldsymbol{B}$	Coefficient submatrix of an equation system	
b_i, $\boldsymbol{b}$	Right-hand side vector of an equation system	
C	Coefficient defining the magnitude of the boundary value (only in Section 3.5)	
C	Coefficient of the magnitude of parasitic currents	
C	General interface indicator function (only in Eq. 2.47)	
c	Coefficient of the momentum interpolation method	$[kg\, m^{-3}\, s^{-1}]$
C_γ	Coefficient to define the resolution of the interface thickness	
C_γ	Number of interface advection time-steps per fluid time-step	
C_κ	Coefficient to define the resolution of the interface curvature	
$C_{\nabla\gamma}$	Coefficient to define the width of the interface region	
C_{ij}, $\boldsymbol{C}$	Coefficient submatrix of an equation system	
C_l	Coefficient to define a general reference length scale	
d	Coefficient of the momentum interpolation method	$[m^3\, s\, kg^{-1}]$
d	Diameter	$[m]$
D_{ij}, $\boldsymbol{D}$	Coefficient submatrix of an equation system	
$E()$	Relative error of a given variable	
E^-, E^+	Unboundedness error of the CICSAM scheme	
E_s	Global spring energy equivalent	$[m^2]$
F	Volume flux	$[m^3\, s^{-1}]$

$f'_{s,i}$, $\boldsymbol{f}'_s$	Local volumetric surface force error	$[N\,m^{-3}]$
$F_{g,i}$, $\boldsymbol{F_g}$	Force due to gravity	$[N]$
$f_{g,i}$, $\boldsymbol{f}_g$	Volumetric force due to gravity	$[N\,m^{-3}]$
f_i, $\boldsymbol{f}$	Body force per unit volume	$[N\,m^{-3}]$
$F_{s,i}$, $\boldsymbol{F_s}$	Surface force	$[N]$
$f_{s,i}$, $\boldsymbol{f}_s$	Volumetric surface force	$[N\,m^{-3}]$
g	Gravitational acceleration	$[m\,s^{-2}]$
g_i, $\boldsymbol{g}$	Gravitational vector	$[m\,s^{-2}]$
h_{ij}	Second derivative of fluid height h	
h_i	First derivative of fluid height h	
k_ψ	Coefficient of the CICSAM scheme	
L	Section length (only in Section 3.8.2)	$[m]$
$L_2()$	L_2 error norm of a given variable	
l_γ	Reference length scale based on the colour function gradient	$[m]$
$L_\infty()$	L_∞ error norm of a given variable	
l_κ	Reference length scale based on the interface curvature	$[m]$
l_e	Length of a mesh edge	$[m]$
l_{max}	Maximum thickness of the interface (with respect to mesh adaption)	$[m]$
l_{min}	Minimum thickness of the interface (with respect to mesh adaption)	$[m]$
l_{ref}	General reference or target length scale	$[m]$
l_s	Edge length of a cubical stencil	$[m]$
M	Number of independent equations in an equation system	
m_i, $\boldsymbol{m}$	Unit normal vector of the interface	
N	Global number of mesh elements	
N_I	Number of cells in the interface region	
N_e	Global number of mesh edges	
N_G	Global number of a given mesh entity	
n_i, $\boldsymbol{n}$	Unit normal vector of a mesh face	
N_L	Local number of a given mesh entity	
N_{proc}	Number of processors used for a given simulation	
N_Q	Number of neighbour cells Q	
p	Pressure	$[Pa]$
R	Equivalent radius of an ellipsoid	$[m]$
r	Radius	$[m]$
r_1, r_2	Principal curvature radii	$[m]$
$r_{b,i}$, $\boldsymbol{r_b}$	Vector connecting cell P and boundary face b	$[m]$
$r_{f,i}$, $\boldsymbol{r}_f$	Vector connecting interpolation point f' and face centre f	$[m]$
R_{max}	Maximum residual value of a given variable	
$r_{P,i}$, $\boldsymbol{r}_P$	Vector connecting cell centre P and face centre f	$[m]$
$r_{Q,i}$, $\boldsymbol{r}_Q$	Vector connecting cell centre Q and face centre f	$[m]$
$r_{U,i}$, $\boldsymbol{r}_U$	Vector connecting upwind cell centre U and face centre f	$[m]$
$s_{f,i}$, $\boldsymbol{s}_f$	Unit vector pointing from cell centre P to cell centre Q	

$S_{g,i}$, $\boldsymbol{S_g}$	Gravity source term	$[N\,m^{-3}]$
S_i, $\boldsymbol{S}$	General source term	$[N\,m^{-3}]$
S_i, $\boldsymbol{S}$	Surface vector (only in Eqs. 2.14 and 2.15)	$[m^3]$
t	Time	$[s]$
U	Characteristic velocity for a spherical inclusion at $Re < 1$	$[m\,s^{-1}]$
u	Velocity component in x-direction	$[m\,s^{-1}]$
u^n	Advecting velocity (see Section 3.3)	$[m\,s^{-1}]$
u_i, $\boldsymbol{u}$	Velocity vector	$[m\,s^{-1}]$
u_t	Terminal rise velocity	$[m\,s^{-1}]$
u_t^∞	Terminal rise velocity in a domain of infinite extend	$[m\,s^{-1}]$
u_w	Wall velocity	$[m\,s^{-1}]$
u_x	Axial velocity	$[m\,s^{-1}]$
V	Volume	$[m^3]$
v	Velocity component in y-direction	$[m\,s^{-1}]$
V_I	Volume of the concave side of the interface	$[m^3]$
w	Velocity component in z-direction	$[m\,s^{-1}]$
w_β	Weighting factor of the advection term of the VOF transport equation	
w_γ	Weighting factor of the colour function	
W_k	Total kinetic energy	$[J]$
w_m	Weighting factor of the interface normal vector	
X	Data set	
x	Coordinate axis	$[m]$
x_i, $\boldsymbol{x}$	Position vector in the Cartesian coordinate system	$[m]$
Y	Data set	
y	Coordinate axis	$[m]$
z	Coordinate axis	$[m]$

Greek Letters

α	Scaling factor of the non-orthogonal correction	
α^*	Non-orthogonality angle	$[rad]$
β, β^{**}	Temporal weighting factor of the CICSAM scheme	
$\Delta\tau$	Sum of present time-step Δt_1 and previous time-step Δt_2	$[s]$
Δp	Pressure difference	$[Pa]$
Δp_{max}	Difference between maximum and minimum pressure in the domain	$[Pa]$
Δs	Distance between cell centres P and Q	$[m]$
Δt	Time-step	$[s]$
Δt_1	Present time-step	$[s]$
Δt_2	Previous time-step	$[s]$
Δt_c	Capillary time-step constraint	$[s]$
Δt_γ	Interface advection time-step	$[s]$
Δx	Mesh spacing	$[m]$
δ	Interpolation coefficient	
ϵ	Convolution length	$[m]$

ϵ_{rel}	Relative tolerance of the numerical solver	
Γ	Diffusion coefficient	$[m^2\, s^{-1}]$
γ	VOF colour function	
γ^*	Average colour function value	
γ_{ref}	Reference colour function for the interface resolution	
κ	Interface curvature	$[m^{-1}]$
κ'	Local curvature error	$[m^{-1}]$
κ^*	Intermediate curvature value	$[m^{-1}]$
μ	Dynamic viscosity	$[Pa\, s]$
Ω	Support of the convolution kernel (only in Section 4.2.1)	
Ω	Surface area	$[m^2]$
ϕ	General fluid variable	$[m\, s^{-1}]$
ϕ_i, $\boldsymbol{\phi}$	Solution vector of an equation system	
ψ	Blending function of the CICSAM scheme (only in Section 3.4.2)	
ψ	Flux limiter of the convection scheme	
ρ	Density	$[kg\, m^{-3}]$
σ	Surface tension coefficient	$[N\, m^{-1}]$
σ_e	Tension of a fictitious massless spring	$[m]$
τ	Characteristic time scale (only in Section 6.3)	$[s]$
τ_{ij}, $\boldsymbol{\tau}$	Stress tensor	$[N\, m^{-2}]$
θ_f	Angle between interface normal vector and vector connecting adjacent cells	$[rad]$
$\tilde{\gamma}$	Normalised colour function value	
φ	Skewness ratio	
ϱ	Pearson product-moment correlation coefficient	
ς	Standard deviation	
ϑ	Dihedral angle between two mesh faces	$[rad]$
ξ	Relative local error	
ζ	Underrelaxation factor of the Laplacian smoothing	

Subscripts

0	Initial value
ϕ	Property of fluid variable ϕ
σ	Originating from surface tension
A	Acceptor cell of face f (with respect to CICSAM scheme)
A	General incompressible Newtonian fluid
B	General incompressible Newtonian fluid
b	Boundary face
D	Donor cell of face f (with respect to CICSAM scheme)
D	Downwind cell of face f
E	Neighbour element situated east of the considered element (cardinal directions)
e	Extrapolation boundary face (only in Section 3.6)
e	Face situated east of the considered element (cardinal directions)
e	Mesh edge under consideration (only Chapter 5)

EE	Element situated east of element E (cardinal directions)
f	Mesh face under consideration
f'	Interpolation point of face f
g	Originating from gravity
i	Fluid properties *inside* the interface (concave side)
i	Tensor component (for partial differential equations)
j	Tensor component (for partial differential equations)
k	Tensor component (for partial differential equations)
m	Neighbour mesh node of mesh node n
n	Mesh node under consideration
o	Fluid properties *outside* the interface (convex side)
P	Mesh cell under consideration
Q	Neighbour mesh cell of mesh cell P
U	Upwind cell of face f
u	Originating from velocity
W	Neighbour element situated west of the considered element (cardinal directions)
w	Face situated west of the considered element (cardinal directions)
$exact$	Analytically exact value
max	Maximum value in the domain
$mean$	Mean value in the domain
ref	Reference value

Superscripts

c	Convoluted variable
i	Element address on a structured mesh (only Eq. 4.13)
i	Present iteration
$i-1$	Previous iteration
j	Element address on a structured mesh (only Eq. 4.13)
n	New mesh cell (with respect to adaptive mesh refinement)
p	Parent mesh cell (with respect to adaptive mesh refinement)
t	Variable at the present time instant
$t-\Delta\tau$	Variable at the previous-previous time instant
$t-\Delta t$	Variable at the previous time instant
t_a	Variable at the time instant of mesh adaption on the new mesh
u	Unconvoluted variable

Non-Dimensional Numbers

Ca	Capillary number
Co	Courant number
Co_D	Courant number of fluxes leaving the donor cell D (CICSAM scheme)
Eo	Eötvös number
Fr	Froude number
La	Laplace number

Mo	Morton number
Pe	Peclet number
Re	Reynolds number
Re_d	Reynolds number based on the diameter
Re_L	Reynolds number based on the domain length
Re_r	Reynolds number based on the radius

Functions

δ_{ij}	Kronecker delta
$f()$	General function
K	General convolution kernel
K_3	Spline convolution kernel
K_6	Sixth-order convolution kernel
K_8	Eighth-order convolution kernel
K_{cos}	Cosine convolution kernel
K_{lin}	Linear convolution kernel

Mathematical Operators and Symbols

$-=$	Subtraction assignment operator
$*$	Convolution operator
$\int$	Integral
$\mathcal{O}()$	Order of the truncation error
∇	Nabla operator
$\oint$	Circular integral
∂	Partial differential operator
$\prod$	Product operator
$\propto$	Proportional to
$\sum$	Summation operator
$+=$	Addition assignment operator

Abbreviations

AMR	Adaptive Mesh Refinement
CD	Central Differencing
CDT	Constraint Delaunay Tetrahedralisation
CELESTE	Curvature Evaluation with Least-Squares fit of Taylor Expansion
CFD	Computational Fluid Dynamics
CICSAM	Compressive Interface Capturing Scheme for Arbitrary Meshes
CLSVOF	Coupled Level-Set and Volume of Fluid
CPU	Central Processing Unit
CSF	Continuum Surface Force
DAC	Direction Averaged Curvature
DNS	Direct Numerical Simulation
FV	Finite Volume
GPU	Graphics Processing Unit
HC	Hyper-C
HF	Height Function
HiRAC	Higher Resolution Artificial Compressive Formulation
HRIC	High Resolution Interface Capturing Scheme
LS	Level-Set
MCLS	Mass-Conserving Level-Set
NVD	Normalised Variable Diagram
PDE	Partial Differential Equation
PISO	Pressure Implicit with Split Operator
PLIC	Piecewise Linear Interface Construction
PROST	Parabolic Reconstruction of Surface Tension
STACS	Switching Technique for Advection and Capturing of Surfaces
SIMPLE	Semi-Implicit Method for Pressure-Linked Equations
SIMPLER	SIMPLE Revised
TVD	Total Variation Diminishing
UD	Upwind Differencing
UQ	Ultimate Quickest
VOF	Volume of Fluid

1. Introduction

The rapid development of computer hardware and numerical methods over the past decades have propelled a sharp rise in the use of *computational fluid dynamics* (CFD) in science and industry. Today CFD is used to predict flows for a variety of applications, from aerospace and automotive to medicine and oil recovery. Although the numerical simulation of single-phase flows is highly developed and predominantly constraint by the available computational resources, the prediction of two-phase flows remains a considerable challenge. The major difficulties in accurately modelling two-phase flows are the representation and tracking of the molecular fluid-fluid interface and the evaluation of the surface tension acting at fluid-fluid interfaces. The objectives of the research presented in this thesis are to advance the understanding of two-phase flow modelling, in particular with respect to unstructured meshes, and to develop numerical methods which provide more accurate results without being constraint to a specific mesh type or a specific application.

1.1. Two-Phase Flow Modelling

The occurrence of two-phase flows in nature and in industrial processes is versatile, making two-phase flows an interesting and important field of study. In the context of this thesis, the terms *two-phase flow* and, synonymously, *interfacial flow* refer to the flow of two immiscible, incompressible fluids. Typically, this represents the interaction of a gas and a liquid separated by an interface, such as an air bubble in water or an ocean surface. However, the term two-phase flow may also describe the interaction of two immiscible liquids. Although two immiscible liquids do not satisfy the physical definition of two different phases, it is common practice within the CFD community to label them as two-phase flows, too. In fact, with respect to the numerical modelling of incompressible, isothermal flows, as considered in this thesis, there is no technical difference between a gas and a liquid phase because only density and viscosity differ. Classical examples for two-phase flows in nature are rain drops, the ocean surface and lava. With respect to industrial applications, the most prominent example for two-phase flows are combustion processes, which are essential to most modes of transportation and energy production. Other examples for industrial applications are cooling processes, oil flow in pipelines, ink-jet printers and metal processing.

The numerical modelling of two-phase flows and the application of such models to predict natural and engineering processes is a very active field of research. Recent numerical studies investigated the catastrophic consequences of storm surges and tsunamis [109, 132] as well as the forces acting on dams [172] and weirs [142]. An oceanic phenomenon receiving notable research attention are rogue waves, also often referred to as freak waves, which are highly non-linear waves that do not correspond to the expected wave height due to the local sea conditions. In 1995, for instance, a high-quality measurement at the Draupner drilling platform in the North Sea recorded a rogue wave of 26 m height [62]. Experimental, theoretical and numerical research focuses on understanding the sea conditions creating rogue waves and on finding mathematical frameworks to predict the occurrence and magnitude of rogue waves [62, 114, 274]. Recent research efforts have seen the use of two-phase flow modelling tools to predict the retreat of glaciers in the Swiss alps [110]. Even though glaciers are made of ice, a glacier as a whole can be considered a highly viscous liquid. The numerical prediction of primary break-up of liquid jets has received considerable attention [94, 212, 251] due to their importance in combustion and spray processes. Other engineering applications of two-phase flow modelling include cooling processes, as for instance in nuclear reactors [104], micro- and nanofluidic applications, such as optimising the underfilling of flip-chip encapsulations for electronic packagings [254], metal processing [136, 262] and understanding the transport of gaseous emboli in human blood flow [52]. Alhendal *et al.* [5] performed numerical studies of the thermo-capillary migration of bubbles at zero gravity, a phenomenon which requires significant effort and resources to be studied experimentally on earth. A better understanding of this phenomenon is important for the design of thermofluid machinery operated in space and of experiments on-board the International Space Station.

Two-phase flow modelling is, however, not only used to predict flows in nature or in industrial applications. Prof. Fedkiw at Stanford University and his co-workers received an Academy Award, commonly known as Oscar, in 2008 for their technical achievements related to their work on simulation tools used to animate two-phase flows in motion pictures, such as Pirates of the Caribbean, Terminator 3 and Harry Potter. An interesting example of how everyday observations can be explained using two-phase modelling was presented by Benilov *et al.* [14], investigating why bubbles in drafted Guinness beer appear to sink rather than ascend. Through numerical experiments Benilov *et al.* demonstrated that the shape of the typical Guinness glass leads to an imbalanced distribution of gas bubbles in the glass. This imbalance initialises a circulation which results in an downward movement of the fluid at the outer radius of the glass and an upwards movement of the fluid in the centre of the glass. Because the bubbles are small in size, their trajectory is dominated by the circulation and, thus, sink at the outer radius of the glass. The results also show that the circulation in the glass reverses if the glass is turned upside down.

The accurate numerical modelling of two-phase flows presents considerable challenges, since the interface is infinitesimally thin with respect to continuum mechanics. Thus, the fluid properties experience a discontinuous change at the interface and a singular force is acting at the interface due to surface tension. Modelling the infinitesimally thin interface in a finite volume or finite element framework is not trivial because of the finite discrete resolution of space and time. Determining the spatial position of the interface on an Eulerian mesh is predominantly a resolution issue and the order of accuracy of the spatial position of the interface, numerically comparable with a shock wave, and of the forces acting at the interface on a Eulerian mesh using finite volume or finite element methods is only first-order [20, 61, 227]. The accurate evaluation of the interface curvature as well as the abrupt and large pressure gradient resulting from the force due to surface tension acting at the interface further complicate the numerical modelling of two-phase flows. Essentially, five particular issues with regards to two-phase flow modelling can be identified:

1. the definition of the force due to surface tension acting at the interface,
2. numerical instabilities as a result of the pressure jump across the interface,
3. the accurate evaluation of the interface curvature,
4. the advection of the sharp interface, and
5. the finite discrete resolution of the interface.

Various methods to address these issues exist, each with its individual advantages and disadvantages. Even though significant resources have been dedicated to the development of methods for two-phase flow modelling in recent decades, no gold-standard to simulate two-phase flows has evolved yet. Thus, it is of utmost importance to understand the available methods and the implications attached to them, in order to apply the best suited method to a given problem.

1.2. Mesh Type

The foundation of every finite difference, finite volume and finite element method is a discrete representation of all modelled dimensions. The spatial domain is, therefore, subdivided into a finite number of non-overlapping elements. The resulting computational meshes can be distinguished by orientation and implementation. The most frequently used classifications of mesh orientation are Cartesian and non-Cartesian meshes. In *Cartesian* meshes the mesh faces are oriented perpendicular to the Cartesian coordinate axes and, therefore, feature desirable numerical properties. Figure 1.1a shows an example of a Cartesian mesh where the computational nodes are arranged equidistant, a particularly beneficial arrangement. Studies by Juretić [111] demonstrated that the face-pairs of Cartesian meshes cancel out certain discretisation errors. Two mesh faces form a pair if their outward-pointing surface vectors sum up to zero. In *non-Cartesian* meshes, on the other hand, the mesh faces are not oriented in a particular fashion and, thus, non-Cartesian meshes can represent domains of arbitrary shape. A typical two-dimensional example of a non-Cartesian mesh is illustrated in Figure 1.1b. The numerical discretisation, however, is more complex on non-Cartesian meshes than it is on Cartesian meshes, since the discretisation has to account for the skewness and non-orthogonality of the mesh. Furthermore, the accuracy is adversely affected by the random orientation of mesh faces and the arbitrary arrangement of computational nodes. With respect to the implementation of the mesh, a structured and an unstructured implementation must be distinguished. A *structured* implementation of the mesh means that each mesh element can be uniquely addressed using a *i,j,k*-indexing system. In an *unstructured* mesh implementation the addressing of mesh elements does not possess an inherent structure and the connectivity must be established individually for each mesh element. The structured implementation is computationally more efficient than an unstructured implementation, whereas an unstructured implementation is independent of the mesh type and the arrangement of the mesh elements.

In the academic two-phase flow community, the use of non-Cartesian meshes is particularly unpopular and controversial. The inherent complexity of two-phase flows and their numerical discretisation complicates the application on non-Cartesian meshes considerably. As a result, most high-fidelity two-phase flow methods reported in the literature are limited to Cartesian meshes, which considerably constraints the applicability regarding complex applications. The advancement of two-phase flow modelling on arbitraryly oriented meshes with an unstructured implementation is, therefore, essential to make two-phase flow modelling applicable to a wider range of applications and industries.

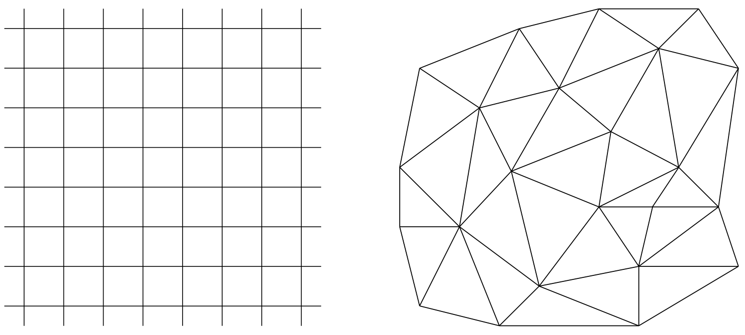

(a) Equidistant Cartesian mesh (b) Triangular mesh

Figure 1.1.: Example of a two-dimensional equidistant Cartesian mesh and a triangular mesh.

1.3. Present Contributions

The research presented in this thesis focuses on two-phase flow modelling on unstructured meshes. As part of the presented research a complete numerical framework for the modelling of two-phase flows on unstructured and adaptive tetrahedral meshes has been developed. A compressive VOF method, which is straightforward to implement on unstructured meshes and inherently conserves mass within the limit of the solver tolerance, is adopted to distinguish two incompressible, isothermal, immiscible fluids. This thesis discusses in detail the discretisation and implementation of the numerical framework and, in addition, elaborates on typical issues of two-phase flow modelling in general and of two-phase flow modelling on unstructured meshes in particular. The major contributions of this thesis to the field of computational fluid dynamics are:

- A fully-coupled balanced-force numerical framework for the simulation of two-phase flows on collocated unstructured meshes is presented. The numerical framework is based on a fully-coupled implicit approach, solving the momentum equations and a specifically constructed continuity constraint in a single linear equation system. The continuity constraint, based on the momentum interpolation method first proposed by Rhie and Chow [198], is derived in Section 3.3 for single-phase flows and subsequently extended to two-phase flows. The continuity constraint facilitates a strong pressure-velocity coupling and preserves continuity. The numerical framework provides and maintains an accurate balance between pressure gradient and body forces, eliminating a major source of errors with respect to large body forces, such as the forces due to surface force or gravity. In particular the presented implementation of the force due to surface tension, the proposed density weighting and the extension to adaptive meshes represent novel contributions. Furthermore, the necessity of a symmetric implementation of the pressure term, and if applicable the body forces, regardless of the mesh type is explained as it is a general source of confusion in the literature. As demonstrated in this thesis, the numerical framework is capable of accurately simulating interfacial flows with large density ratios and arbitrary interface topology on structured, unstructured and adaptive meshes. The successful application of a balanced-force framework, such as the one presented in this thesis, to moving interfaces and on adaptive meshes has not been previously reported in the literature and is demonstrated for the first time.

- A new method for the evaluation of the interface curvature directly from volume fractions is proposed. The evaluation of the interface curvature represents a major challenge for two-phase flow modelling with interface capturing methods, as a result of the implicit interface representation and the finite numerical resolution. Inaccurate curvature estimates directly affect the accuracy of the results and, in severe cases, parasitic currents resulting from inaccurate interface curvature estimates may even destroy the interface. The new method, presented in Section 4.3, is based on a least-squares fit of a second-order Taylor series expansion of the volume fraction field and is applicable to arbitrary meshes. The presented method yields similar or better results than existing methods, which are typically limited to Cartesian meshes, and the results on structured and unstructured meshes obtained with the new method are shown to be comparable.

- The findings presented in this work contribute substantially to the understanding of the application of convolution methods to two-phase flow simulations. Convolution of the interface indicator function (*e.g.* the volume fraction) in interface capturing/tracking methods is a common way to smooth the momentum discontinuity at fluid-fluid interfaces. With respect to VOF methods, convolution is also applied to improve curvature estimates calculated from the volume fraction field. However, the success of applying convolution methods depends on the use of appropriate convolution stencils and on the application to the correct variables. Different convolution strategies with respect to fluid properties and the force due to surface tension are examined in Section 4.4 and the implications of the convolution stencil size is assessed in Section 4.5. The conducted studies highlight inherent problems of convolution, misconceptions about the correct application of convolution and miscorrelations between common academic test cases and realistic applications. Although the aspects of convolution are discussed and assessed using a VOF method, the findings equally apply to level-set methods and front-tracking methods.

- The origin of parasitic currents in the vicinity of interfaces is examined. Most importantly, the presented results clearly show that parasitic currents and an inaccurate pressure jump across the interface have different origins and are independent of each other. It is also demonstrated that commonly used indicators for the applicability of respective two-phase flow methods are only applicable in some cases but cannot be used in all instances. Furthermore, examples where scaling of a test case can reduce numerical errors but equally maintain the defining flow features are identified.
- The fundamentals of the application of tetrahedral mesh adaption algorithms to two-phase flow simulations are studied. In this context, an implementation concept for unstructured mesh adaption algorithms is presented in Section 5.4, applicable to single-processor and multi-processor computer architectures. Furthermore, the mesh resolution at interfaces with respect to suitable parameters for the control of mesh adaption algorithms is investigated in Section 5.5 and reference length scales are derived to determine the required mesh resolution. The conducted study contributes to the understanding of mesh resolution requirements at interfaces and demonstrates the applicability of the proposed numerical framework to adaptive meshes, satisfying conservation laws and maintaining a discrete balance between pressure gradient and body forces.

1.4. Thesis Outline

The remainder of this thesis is structured as follows:

In **Chapter 2**, the governing equations are presented and the fundamentals of the used numerical methods are explained. The focus of this chapter is to lay the groundwork for the research presented in this thesis. In particular, the discretisation errors related to unstructured meshes and the problems associated with two-phase flow modelling are discussed.

The fully-coupled balanced-force numerical framework for unstructured meshes developed as part of the research presented in this thesis is devised and discussed in **Chapter 3**. The discretisation of the governing equations is examined in detail and a continuity constraint, specifically designed for the simulation of flows with large body forces on arbitrary meshes, is derived. This continuity constraint conserves continuity, couples pressure and velocity and maintains an accurate balance between body forces and pressure gradient.

Chapter 4 is concerned with the numerical representation of fluid-fluid interfaces. This chapter focuses on the correct application of convolution to two-phase flows and the accurate evaluation of the interface curvature. A new method to evaluate the interface curvature on arbitrary meshes is presented and validated.

In **Chapter 5**, the application of adaptive tetrahedral meshes at interfaces is studied. An adaption algorithm for multi-processor computer architectures is presented and suitable reference length scales for the resolution of the interface are derived. Furthermore, the used numerical framework is extended to maintain an accurate balance between body forces and the pressure gradient on adaptive meshes.

The results of additional test cases to evaluate the proposed methods are presented in **Chapter 6**. The test cases assess the accurate description and interaction of viscous stresses, the force due to surface tension and gravity. Additionally, the influence of interface properties on parasitic currents is assessed and the origin of parasitic currents is examined.

In **Chapter 7**, the thesis is concluded and suggestions for future work are discussed.

The **Appendix** briefly discusses common topics concerning the implementation of the numerical framework developed as part of the presented research. Furthermore, the damping of spurious pressure oscillations arising from a collocated variable arrangement is explained with a short example.

2. Fundamentals

The numerical methods presented in this study are designed for incompressible, isothermal, immiscible Newtonian fluids[1] and are based on continuum mechanics principles, assuming that the macroscopic physical properties of the fluid can be described as continuous functions in space and time. This assumes that relevant length scales of the flow are significantly larger than the discrete structures of the simulated materials. Hence, the fluid in a domain of finite size can be described by a set of differential equations and boundary conditions. The spatial domain is divided into a finite number of discrete control volumes by means of a computational mesh and the temporal domain is represented by finite time-steps. The differential equations describing the flow are discretised in space and time on the applied computational mesh and time-steps, using numerical differencing schemes which are founded on the *finite volume* (FV) method. The resulting set of algebraic equations is solved utilising preconditioning and iterative solving methods.

In this chapter the governing equations describing incompressible, isothermal, immiscible Newtonian fluids are presented in Section 2.1 and the basic discretisation in space and time is devised in Section 2.2, using the example of convective-diffusive transport of a passive scalar. Subsequently, the fundamentals of the numerical treatment of two-phase flows are discussed in Section 2.3.

2.1. Governing Equations

The flow of a fluid is governed by two conservation laws: mass conservation and momentum conservation. In what follows, the equations constituting these conservation laws are briefly presented. The interested reader may refer to the textbook of Versteeg and Malalasekera [249] for a detailed derivation of the governing equations.

2.1.1. Conservation of Mass

Conservation of mass is a fundamental concept in fluid mechanics, described by the continuity equation. Observing the fluid in an infinitesimally small control volume, the change of mass is equal to the sum of mass flux over the bounding faces. Two assumptions about the fluid characteristics have to be distinguished concerning the conservation of mass: compressible and incompressible fluids. In *compressible* fluids the fluid density is dependent on the surrounding pressure. An *incompressible* fluid, on the other hand, shows no or negligible pressure dependency of its density. Hence, the density is taken to be dependent

[1] A *Newtonian fluid* is characterised by a linear relationship between velocity and viscous stresses.

on the temperature only. A common example for an incompressible fluid is water at practical velocity. Most gases may be assumed to be incompressible for low Mach numbers, as for instance air for Mach numbers smaller than 0.3. For a compressible fluid the continuity equation is defined as

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho u_i}{\partial x_i} = 0 \ , \tag{2.1}$$

where ρ represents the fluid density, t stands for time and $\boldsymbol{u}$ is the velocity. Due to the negligible density change, continuity for an incompressible fluid is satisfied if the divergence of the velocity field is zero and, thus, the continuity equation for an incompressible fluid becomes

$$\frac{\partial u_i}{\partial x_i} = 0 \ . \tag{2.2}$$

This definition also holds for two-phase flows with two fluids of different density. Reformulating the definition for the conservation of mass with variable density in Eq. 2.1 follows as

$$\frac{\partial \rho}{\partial t} + \rho \frac{\partial u_i}{\partial x_i} + u_i \frac{\partial \rho}{\partial x_i} = 0 \ . \tag{2.3}$$

Inserting the material derivative of the density, given as

$$\frac{D\rho}{Dt} = \frac{\partial \rho}{\partial t} + u_i \frac{\partial \rho}{\partial x_i} = 0 \tag{2.4}$$

in Eq. 2.3 follows as

$$\rho \frac{\partial u_i}{\partial x_i} = 0 \ , \tag{2.5}$$

therefore, proving that mass is conserved applying Eq. 2.2.

2.1.2. Transport of Momentum

The transport of momentum is described by the momentum equation and originates from Newton's *second law*. The momentum equation for a Newtonian fluid is

$$\rho \left(\frac{\partial u_i}{\partial t} + u_j \frac{\partial u_i}{\partial x_j} \right) = \frac{\partial \tau_{ij}}{\partial x_j} + \sum f_i \ , \tag{2.6}$$

where $\boldsymbol{\tau}$ is the tensor representing stresses in the fluid and $\boldsymbol{f}$ are external forces per unit volume[2] acting on the fluid, such as the force due to gravity. The stresses in a general Newtonian fluid can be described as

$$\tau_{ij} = \mu \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) - \frac{2}{3} \mu \frac{\partial u_k}{\partial x_k} - p \ \delta_{ij} \ , \tag{2.7}$$

where μ is the viscosity of the fluid, p represents pressure and δ_{ij} is the Kronecker delta. For an incompressible fluid the stress tensor simplifies to

$$\tau_{ij} = \mu \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) - p \ \delta_{ij} \ . \tag{2.8}$$

[2]Alternative terms for *force per unit volume* are *volumetric force* or *body force*.

Inserting the stress tensor of Eq. 2.8 in Eq. 2.6, the momentum equation for an incompressible Newtonian fluid, without external body forces, becomes

$$\rho \left(\frac{\partial u_i}{\partial t} + u_j \frac{\partial u_i}{\partial x_j} \right) = -\frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j} \left[\mu \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) \right] \ . \tag{2.9}$$

External body forces, such as gravity, may be included in the momentum equation as a source term on the right-hand side. For instance, including the volumetric force due to gravity $\boldsymbol{f}_g = \rho \boldsymbol{g}$ in the momentum equation follows as

$$\rho \left(\frac{\partial u_i}{\partial t} + u_j \frac{\partial u_i}{\partial x_j} \right) = -\frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j} \left[\mu \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) \right] + f_{g,i} \ . \tag{2.10}$$

2.1.3. Transport of Passive Scalars

The transport of a passive scalar property, such as thermal energy, in a fluid is described by a convection-diffusion equation, typically referred to as *transport equation*. The transport equation for the general fluid variable ϕ of an incompressible fluid is given as

$$\rho \left(\frac{\partial \phi}{\partial t} + u_j \frac{\partial \phi}{\partial x_j} \right) - \frac{\partial}{\partial x_j} \left(\Gamma_\phi \frac{\partial \phi}{\partial x_j} \right) = S_i \ , \tag{2.11}$$

where Γ_ϕ is the diffusion coefficient of fluid variable ϕ. The first term on the left-hand side describes the temporal derivative of ϕ followed by the convective and diffusive transport. The right-hand side contains the non-linear and linear source terms of the transport variable, combined in source term $\boldsymbol{S}$.

2.2. Finite Volume Method

The finite volume method is the most frequently used discretisation framework in CFD. It can be applied to every common, non-overlapping mesh and all approximations are based on physical conservation principles. In the finite volume method, control volumes are defined for each element of the mesh and each element is bounded by a finite number of faces. The integral of each equation has to be fulfilled for each control volume. Thus, the discrete value of a general, continuous fluid variable $\phi(\boldsymbol{x}, t)$ at control volume P has to fulfil the condition

$$\int_{V_P} \left(\phi(\boldsymbol{x}, t) - \phi_P \right) dV = 0 \ , \tag{2.12}$$

where V represents volume and subscript P denotes discrete values at the centre of control volume P. The discretisation accuracy of fluid variable ϕ is directly dependent on the variation of $\phi = \phi(\boldsymbol{x}, t)$. In order to obtain a second-order accurate approximation, it is assumed that the variation in space and time is linear. Therefore, applying the *Gauss theorem*[3], the value at element centre P is calculated based on the value at the volume

[3]The *Gauss theorem* is also frequently called *Green-Gauss theorem* or *divergence theorem.*

surface as

$$\int_{V_P} \nabla\phi \; dV = \oint_{\Omega} \phi \boldsymbol{n} \; d\Omega \; , \tag{2.13}$$

where Ω denotes the surface area and $\boldsymbol{n}$ is the outward-pointing surface normal vector. Since the surface of the volume is constituted by a finite number of flat faces f, the integration over the surface can be expressed as

$$\oint_{\Omega} \phi \boldsymbol{n} \; d\Omega = \sum_f \left(\int_f \phi \; d\boldsymbol{S} \right) \; , \tag{2.14}$$

with $\boldsymbol{S}$ being the outward-pointing surface vector. The flow variable ϕ, however, is generally not known everywhere on a given surface but rather at element centres or face centres only, dependent on the variable arrangement. Therefore, every face is represented by its centre point, the so-called *midpoint rule*, assuming a linear variation of ϕ in space. This leads to the second-order approximation at face centres

$$\sum_f \left(\int_f \phi \; d\boldsymbol{S} \right) \approx \sum_f \phi_f \, \boldsymbol{n}_f \, A_f \; , \tag{2.15}$$

where subscript f denotes values at face centre f and surface vector $\boldsymbol{S}$ is represented by its individual components, the outward-pointing unit normal vector $\boldsymbol{n}_f$ of face f and the face area A_f. As a result, the volume integral of Equation 2.13 is approximated as

$$\int_{V_P} \nabla\phi \; dV \approx \sum_f \phi_f \, \boldsymbol{n}_f \, A_f \; . \tag{2.16}$$

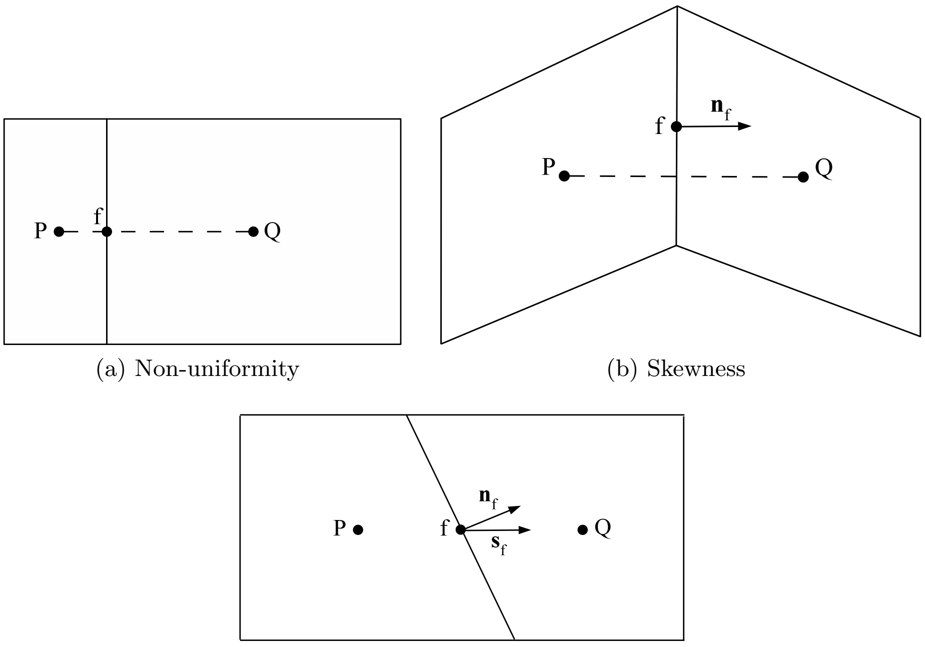
For an in-depth explanation and discussion of the finite volume method the interested reader may refer to the textbook of Versteeg and Malalasekera [249] or the PhD thesis of Jasak [106].

2.2.1. Mesh Errors

The distribution and quality of the computational mesh is very important for the accuracy of the simulation results and the stability of the numerical solver. Using arbitrary meshes, three potential errors with respect to the orientation and position of the mesh faces have to be considered [111], as illustrated in Figure 2.1:

1. non-uniformity,
2. skewness, and
3. non-orthogonality.

A mesh is called *uniform* if the shared face of two adjacent computational nodes is situated equidistant with respect to both nodes. For comparison, Figure 2.1a shows a face that is situated *non-uniform.* In the context of an interpolation from adjacent element centres to the shared face centre, a face is called *skewed* if the geometric face centre does not coincide with the point where the vector connecting the adjacent element centres intersects the face,



(a) Non-uniformity (b) Skewness

(c) Non-orthogonality

Figure 2.1.: Mesh cell P with neighbour cell Q and shared face f of a non-uniform, a skewed and a non-orthogonal quadrilateral two-dimensional mesh.

as illustrated in Figure 2.1b. Uniformity and skewness are of significance when values stored at the element centres are to be interpolated to face centres, or *vice versa*. Non-orthogonality of mesh faces represents another possible source of error. Again, with respect to an interpolation from adjacent element centres to the shared face centre, a mesh face is called *non-orthogonal* if the vector connecting the two adjacent element centres is not parallel to the normal vector of the face, which is shown in Figure 2.1c. Non-orthogonality presents a particular problem in cases where gradients are to be evaluated at face centres, for instance in the viscosity term of the momentum equation (Eq. 2.10) or in the diffusion term of the transport equation (Eq. 2.11).

2.2.2. Spatial Interpolation

In case of a uniform mesh, as illustrated in Figure 2.2a, a second-order interpolation from element centres to face centres can be performed by a linear interpolation, defined as

$$\phi_f = \frac{\phi_P + \phi_Q}{2} \, , \tag{2.17}$$

where P and Q denote the elements adjacent to face f. Using unstructured meshes, however, a uniform constellation of two elements is the exception rather than the norm. If the mesh is non-uniform, illustrated for example in Figure 2.1a, an appropriate interpolation coefficient for the element-centred values has to be defined to improve the accuracy of the interpolation. Thus, the face-centred value is defined as

$$\phi_f = (1 - \delta)\, \phi_P + \delta\, \phi_Q \tag{2.18}$$

with δ being the interpolation coefficient. The most intuitive approach is weighting the element-centred values by their inverse distance to face centre f [174], with the interpolation coefficient following as

$$\delta = \frac{|\boldsymbol{r}_P|}{|\boldsymbol{r}_P| + |\boldsymbol{r}_Q|} \ , \tag{2.19}$$

where $\boldsymbol{r}_P$ and $\boldsymbol{r}_Q$ represent the vectors from element centres P and Q to face centre f, respectively. The linear interpolation with inverse distance weighting is second-order accurate on Cartesian meshes and generally if $\boldsymbol{r}_P = \boldsymbol{r}_Q$ [260]. However, distance weighting is not the only possibility and Dalal *et al.* [36] used a volume-weighted interpolation approach, given as

$$\delta = \frac{V_P}{V_P + V_Q} \ , \tag{2.20}$$

with V representing the volume of the respective element.

Interpolating element-centred values to face centres by means of Eq. 2.18 leads to significant errors on meshes with considerable skewness, illustrated in Figure 2.1b. In order to correct the interpolation for mesh skewness, the element-centred values are interpolated to interpolation point f', which is defined as the intersection point between the face and the vector connecting the element centres, as depicted in Figure 2.2b. The interpolated value at point f' is then corrected to face centre f using the first derivative at face centre f. For face f with its adjacent elements P and Q, following Figure 2.2b, the interpolation is defined as

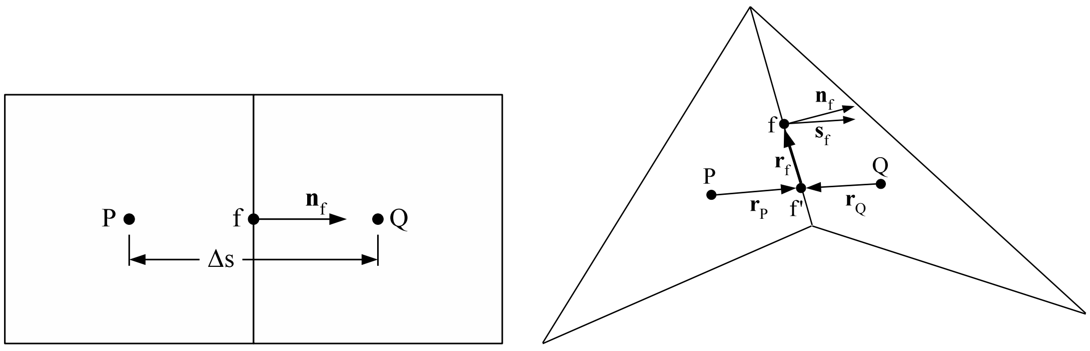
$$\phi_f = (1-\delta)\ \phi_P + \delta\ \phi_Q + \Big[(1-\delta)\ \ \nabla\phi|_P + \delta\ \ \nabla\phi|_Q\Big]\, \boldsymbol{r}_f \ , \tag{2.21}$$

where $\boldsymbol{r}_f$ is the vector from interpolation point f' to face centre f. The most common way to weight the element-centred values in Eq. 2.21 is to apply the inverse distance from element centres P and Q to interpolation point f', following Eq. 2.19. Perez-Segarra *et al.* [177] proposed an alternative weighting, where interpolation point f' is the point situated on the line connecting elements P and Q closest to face centre f. The interpolation coefficient δ is specified as

$$\delta = \frac{\boldsymbol{r}_f \cdot \boldsymbol{s}_f}{\Delta s} \ , \tag{2.22}$$

with $\boldsymbol{s}_f$ representing the unit vector pointing from cell centre P to cell centre Q and Δs being the distance between cell centres P and Q, as depicted in Figure 2.2a. Farre *et al.* [65] did not find significant differences in accuracy between the interpolation coefficients presented in Eq. 2.19 and Eq. 2.22, but their study also suggests that weighting the element values using Eq. 2.22 improves convergence on severely non-orthogonal meshes. Assuming a perfectly orthogonal mesh both weighting methods result in the same interpolation coefficients. Karimian and Straatman [113] used $\delta = 0.5$ to weight the element-centred values, arguing that it reduces the likeliness of extreme values at face centres because of the symmetric weighting.

For the remainder of this thesis, if not explicitly stated otherwise, the gradient-corrected interpolation presented in Eq. 2.21 with inverse distance weighting as defined by Eq. 2.19 is applied for spatial interpolation.



(a) Equidistant Cartesian mesh (b) Triangular mesh

Figure 2.2.: Mesh element P with neighbour element Q and shared face f of an equidistant Cartesian and a triangular two-dimensional mesh including interpolation entities.

2.2.3. Convection Term

The discretisation of convection terms is presented using the transport equation defined in Eq. 2.11 and is equally applicable to other convection-diffusion equations, such as the momentum equation. The convection term of Eq. 2.11 is discretised by integrating over the element volume using the Gauss theorem and the midpoint rule, following as

$$\int_{V_P} \rho\, u_j \frac{\partial \phi}{\partial x_j}\, dV \approx \sum_f \phi_f\, \rho_f\, (\boldsymbol{n}_f \cdot \boldsymbol{u}_f)\, A_f \tag{2.23}$$

$$\approx \sum_f \dot{m}_f\, \phi_f \ , \tag{2.24}$$

where ϕ_f is the fluid variable at the centre of face f and $\dot{m}_f = \rho_f\, (\boldsymbol{u}_f \cdot \boldsymbol{n}_f)\, A_f$ is the mass flux through the centre of face f. The transport variable ϕ_f at face f is evaluated using an appropriate differencing scheme. In what follows, two common convection differencing schemes are discussed, upwind differencing and central differencing. *Total variation diminishing* (TVD) schemes represent an alternative for the discretisation of convection terms but are not considered in this work. For a summary of TVD schemes for unstructured meshes the interested reader may consult the paper of Darwish and Moukalled [38].

The *upwind differencing* (UD) scheme is a first-order accurate differencing scheme used in convection-dominated flows. The face value of fluid variable ϕ is determined based on the flow direction and follows as

$$\phi_f = \begin{cases} \phi_U & \text{for } \dot{m}_f \geq 0 \\ \phi_D & \text{for } \dot{m}_f < 0 \ , \end{cases} \tag{2.25}$$

where U and D denote the upwind and downwind elements, respectively. Advantages of the UD scheme are the guarantee of a bounded solution [174] and the simplicity of implementation. On the other hand, the UD scheme introduces a substantial amount of numerical diffusion which adversely affects the quality of the solution.

The *central differencing* (CD) scheme assumes a linear variation of fluid variable ϕ. Using the values at both adjacent elements, the CD scheme in its general form is defined as

$$\phi_f = (1-\delta)\ \phi_P + \delta\ \phi_Q\ , \tag{2.26}$$

where δ is the interpolation coefficient as defined in Section 2.2.2. The CD scheme may be corrected as demonstrated in Eq. 2.21 if meshes with high skewness are used. An alternative CD-type scheme follows a deferred correction approach. In a deferred correction approach a low-order scheme is implemented implicitly to reduce the stencil size and to increase the diagonal dominance of the matrix of the linear equation system, and a high-order correction is implemented explicitly to increase the accuracy of the discretisation scheme. The CD-scheme implemented using a deferred correction approach is based on an implicit UD scheme implementation and an explicit high-order correction by means of a gradient correction. The face value is defined as

$$\phi_f = \phi_U + \nabla\phi|_U \cdot \boldsymbol{r}_U\ , \tag{2.27}$$

where $\boldsymbol{r}_U$ is the vector connecting upwind element centre U and face centre f. This implementation provides an increased numerical stability compared to the traditional CD scheme implementation (Eq. 2.26) as a result of the unconditionally stable UD scheme representing the implicit term. The CD scheme is second-order accurate [68] but may result in unphysical oscillations since the scheme is not bounded for convection-dominated flows [106, 174]. Typically, the CD scheme becomes unstable for Peclet numbers $Pe = \Delta x\,|\boldsymbol{u}|/\Gamma \geq 2$ [49], where Δx is the mesh spacing and Γ is the diffusion coefficient. Studies of Farre *et al.* [65] indicate that the CD scheme leads to divergent results if $Pe \gg 10$.

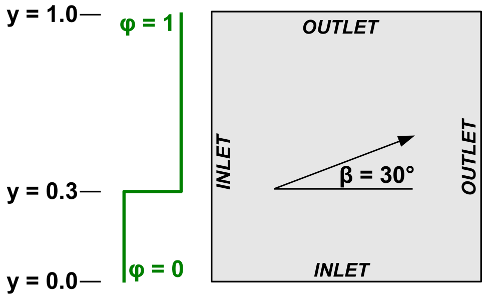
The results of a simple test case, following a test case reported by Jasak [106, chap. 3.7.1], are shown below to demonstrate the application of UD and CD schemes. The step profile of a passive scalar is convected at an angle of 30° on an equidistant Cartesian mesh with a Courant number of $Co = |\boldsymbol{u}|\,\Delta t/\Delta x = 0.1$. The applied boundary conditions are illustrated in Figure 2.3a. As expected, the result for upwind differencing, depicted in Figure 2.3b, shows significant numerical diffusion. Central differencing, on the other hand, results in considerably less numerical diffusion but leads to notable oscillations of the scalar field, as shown in Figure 2.3c. The results presented in Figure 2.3 are in excellent agreement with the results of Jasak [106] for this test case.

2.2.4. Diffusion Term

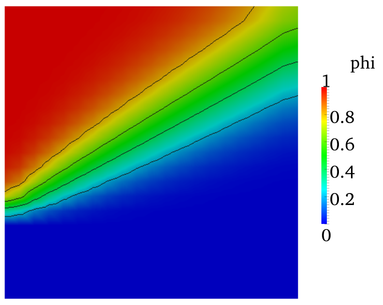
Applying the Gauss theorem and the midpoint rule to the integral form of the diffusion term of the transport equation presented in Eq. 2.11 is given as

$$\int_{V_P} \frac{\partial}{\partial x_j}\left(\Gamma_\phi \frac{\partial \phi}{\partial x_j}\right)\ dV \approx \sum_f (\Gamma_\phi \nabla\phi)_f\, \boldsymbol{n}_f\, A_f\ . \tag{2.28}$$

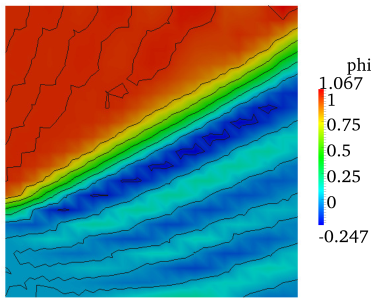
Considering a mesh with orthogonally oriented faces and a linear spatial variation of ϕ, the gradient at face centre f can be calculated with second-order accuracy using central



(a) Boundary conditions



(b) Upwind differencing



(c) Central differencing

Figure 2.3.: Convection of a step profile at a constant oblique velocity using upwind differencing and central differencing.

differencing, defined as

$$\nabla\phi|_f = \frac{\phi_Q - \phi_P}{\Delta s} \ . \tag{2.29}$$

However, on unstructured meshes orthogonality is a rare exception and non-orthogonality has to be corrected to maintain the stability of the solving algorithm.

A deferred correction approach is typically deployed to correct the non-orthogonality of the mesh [162]. The gradient calculation is decomposed into an orthogonal and a non-orthogonal contribution, following as

$$\nabla\phi|_f \cdot \boldsymbol{n}_f = \nabla\phi|_f \cdot (\alpha_f\, \boldsymbol{s}_f) + \overline{\nabla\phi}\big|_f \cdot (\boldsymbol{n}_f - \alpha_f\, \boldsymbol{s}_f) \ , \tag{2.30}$$

where $\boldsymbol{s}_f$ is the normalised vector connecting the element centres adjacent to face f and α_f is the scaling factor of the decomposition. The first term on the right-hand side of Eq. 2.30, representing the orthogonal contribution, is calculated using the nearest neighbours as

$$\nabla\phi|_f \cdot (\alpha_f\, \boldsymbol{s}_f) = \alpha_f \frac{\phi_Q - \phi_P}{\Delta s} \tag{2.31}$$

and is implemented implicitly. The face-centred gradient in the second term on the right-hand side of Eq. 2.30, which is the non-orthogonal contribution, is taken as the average of the gradients at the adjacent element centres of the previous iteration, defined as

$$\overline{\nabla\phi}\big|_f = (1-\delta)\ \nabla\phi|_P + \delta\ \nabla\phi|_Q \ \ , \tag{2.32}$$

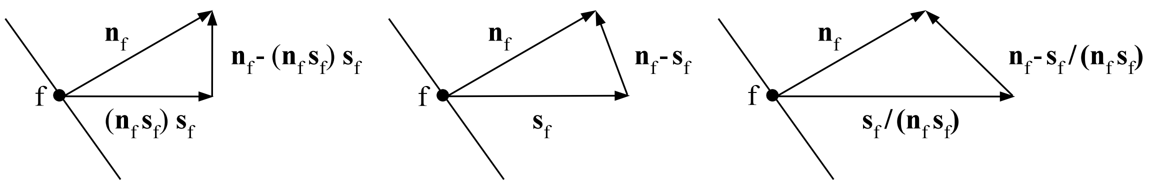
and is implemented explicitly. Thus, Eq. 2.30 becomes

$$\nabla\phi|_f \cdot \boldsymbol{n}_f = \alpha_f \frac{\phi_Q - \phi_P}{\Delta s} + \left[(1-\delta)\ \nabla\phi|_P + \delta\ \nabla\phi|_Q\right] \cdot (\boldsymbol{n}_f - \alpha_f\, \boldsymbol{s}_f)\ . \tag{2.33}$$

and the complete diffusion term follows as

$$(\Gamma_\phi \nabla\phi)_f\, \boldsymbol{n}_f\, A_f = \left[\alpha_f \frac{\phi_Q - \phi_P}{\Delta s} + \left[(1-\delta)\ \nabla\phi|_P + \delta\ \nabla\phi|_Q\right] \cdot (\boldsymbol{n}_f - \alpha_f\, \boldsymbol{s}_f)\right] \Gamma_{\phi,f}\, A_f\ . \tag{2.34}$$

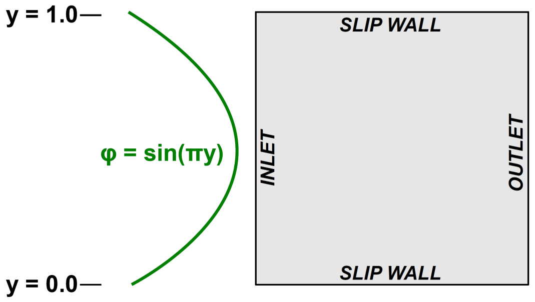
Three basic decompositions for the non-orthogonal correction are available [106, 177], determined by the choice of the scaling factor α_f, as illustrated in Figure 2.4. The *minimal correction* approach, $\alpha_f = \boldsymbol{n}_f \cdot \boldsymbol{s}_f$, minimises the non-orthogonal correction [277]. The *orthogonal correction* approach [50, 162], $\alpha_f = 1$, leaves the contribution of the implicit term unchanged regardless of the angle between $\boldsymbol{n}_f$ and $\boldsymbol{s}_f$. The contribution of the implicit part increases with increasing non-orthogonality if the *overrelaxed correction* [111, 152], $\alpha_f = (\boldsymbol{n}_f \cdot \boldsymbol{s}_f)^{-1}$, is used. According to Zwart [277], the minimal correction approach provides theoretically the highest accuracy of the three approaches, as the length of the correction vector $\boldsymbol{n} - \alpha \boldsymbol{s}$ is minimised. However, as studies of Jasak [106] demonstrate, the minimal correction approach is unstable for large non-orthogonalities, because the contribution of the implicit term diminishes as the angle between $\boldsymbol{n}$ and $\boldsymbol{s}$ increases. The orthogonal correction is more robust than the minimal correction approach at the cost of a lower accuracy. Ahipo and co-workers [2, 234] showed that the orthogonal correction approach, $\alpha_f = 1$, diverges if the non-orthogonality exceeds an angle of 38°. The overrelaxed approach provides the highest numerical stability of the three approaches on meshes with considerable non-orthogonality [65, 106, 236], since the coefficient of the implicit term is unconditionally positive [49, chap. 6.5], increasing the diagonal dominance of the coefficient matrix. Because this thesis focuses on unstructured meshes with potentially significant local non-orthogonality, the overrelaxed correction following Mathur and Murthy [152] with $\alpha_f = (\boldsymbol{n}_f \cdot \boldsymbol{s}_f)^{-1}$ is used throughout this work to assure a stable convergence.



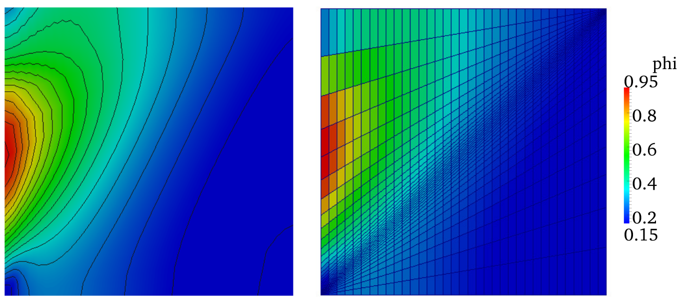
(a) Minimal correction (b) Orthogonal correction (c) Overrelaxed correction

Figure 2.4.: Non-orthogonal correction methods at face f of a non-orthogonal two-dimensional mesh.

The diffusion of a general fluid variable is simulated on a non-orthogonal hexahedral mesh to demonstrate the necessity and capabilities of the applied correction method. A sinusoidal profile of the general fluid variable ϕ is transported within a cubical domain by diffusion only, as illustrated in Figure 2.5a. The results given in Figures 2.5b and 2.5c, calculated without non-orthogonal correction, show significant errors which are clearly induced by the underlying mesh. Performing the same test case using the overrelaxed non-orthogonal correction, the result obtained on this non-orthogonal mesh, shown in Figure 2.6a, is in very good agreement with the result obtained on the equidistant Cartesian mesh, shown in Figure 2.6b, despite the considerable non-orthogonality of the mesh.

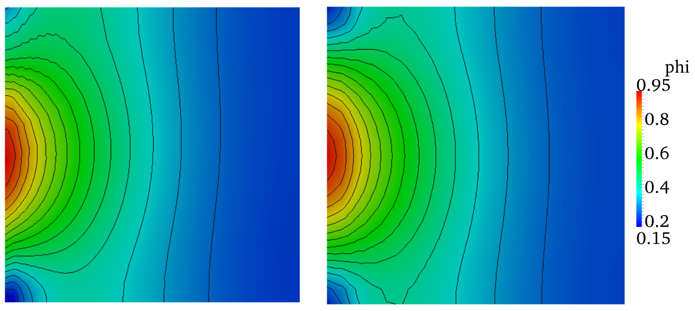


(a) Boundary conditions



(b) Smoothed contours (c) Cell-centred values

Figure 2.5.: Diffusion of a sinusoidal profile on a non-orthogonal hexahedral mesh without non-orthogonal correction.



(a) Non-orthogonal hexahedral mesh (b) Cartesian mesh

Figure 2.6.: Diffusion of a sinusoidal profile on a non-orthogonal hexahedral mesh with non-orthogonal correction and on an equidistant Cartesian mesh.

2.2.5. Transient Term

The discretisation of time as the fourth dimension is required to perform unsteady simulations. The major difference between transient terms and the spatial terms discussed in previous sections is the direction of influence. Spatial terms describe an elliptical problem whereas transient terms are parabolic, as there is no backward influence in time.

A solution at any given time instant is calculated from a given distribution at the preceding time instant. Therefore, discretising the variation in time of a given initial value problem

$$\frac{\partial \phi}{\partial t} = f(t, \phi(t)) \tag{2.35}$$

follows as

$$\int_{t-\Delta t}^{t} \frac{\partial \phi}{\partial t} dt = \phi^{t} - \phi^{t-\Delta t} = \int_{t-\Delta t}^{t} f(t, \phi(t)) dt \ , \tag{2.36}$$

where the superscripts $t - \Delta t$ and t denote the value at the old and the new time instant, respectively. Evaluating the integral on the right-hand side requires an approximation according to the chosen discretisation scheme. Approximating the right-hand side using the values at the old time instant results in

$$\phi^{t} = \phi^{t-\Delta t} + f(t^{t-\Delta t}, \phi^{t-\Delta t})\Delta t + \mathcal{O}(\Delta t) \ , \tag{2.37}$$

and is called *Explicit Euler* or *Forward Euler* scheme. The Explicit Euler scheme becomes unstable for large time-steps and, thus, suffers from strict Courant number limitations, $Co \leq 1$, because of its explicit implementation. Approximating the right-hand side instead with the values at the new time instant, Eq. 2.36 becomes

$$\phi^{t} = \phi^{t-\Delta t} + f(t^{t}, \phi^{t})\Delta t + \mathcal{O}(\Delta t) \ , \tag{2.38}$$

which is called *Implicit Euler* or *Backward Euler* scheme. The Implicit Euler scheme is unconditionally stable and guarantees boundedness of the solution [174]. Both Euler schemes, explicit and implicit, are first-order accurate.

Applying trapezoidal integration to the right-hand side of Eq. 2.36 results in the second-order accurate *Crank-Nicolson* scheme, following as

$$\phi^{t} = \phi^{t-\Delta t} + \frac{1}{2}\left[f(t^{t-\Delta t}, \phi^{t-\Delta t}) + f(t^{t}, \phi^{t})\right]\Delta t + \mathcal{O}(\Delta t^{2}) \ . \tag{2.39}$$

The Crank-Nicolson scheme is unconditionally stable but is known to potentially cause unboundedness of the solution [175].

Another second-order accurate temporal discretisation scheme is the *Second-Order Backward Euler* scheme, also called *3-level Implicit Euler* scheme, which uses two previous time instants in addition to the present time instant for discretisation. In case time-step Δt is constant, Eq. 2.36 discretised using the Second-Order Backward Euler scheme becomes

$$\frac{3\phi^t - 4\phi^{t-\Delta t} + \phi^{t-2\Delta t}}{2\Delta t} = f(t^t, \phi^t) + \mathcal{O}(\Delta t^2) \tag{2.40}$$

$$\phi^t = \frac{4}{3}\phi^{t-\Delta t} - \frac{1}{3}\phi^{t-2\Delta t} + \frac{2}{3}f(t^t, \phi^t)\Delta t \; . \tag{2.41}$$

Compared to the Crank-Nicolson scheme, the Second-Order Backward Euler scheme is easier to implement, since the spatial discretisation does not require special attention and the additional transient term is merely an addition to the right-hand side of the equation system. Studies by Jasak [106] demonstrated that the Second-Order Backward Euler scheme and the Crank-Nicolson scheme yield similar results, however, the latter potentially becomes unbounded.

2.3. Two-Phase Flows

The simulation of two-phase flows requires a numerical representation of the interface separating the two fluids. The work presented in this thesis utilises a *one-fluid formulation*, where the involved fluids are distinguished by their properties, *i.e.* density and viscosity, and the force due to surface tension is included in the governing equations as a source term. In what follows, a brief introduction of important interface tracking and interface capturing methods is given, followed by a detailed presentation of the *Volume of Fluid* method, which is adopted to capture the interface in the presented study. Subsequently, the source term representing the volumetric surface force is discussed and numerical conservation issues in interfacial flows are examined.

2.3.1. Overview of Interface Tracking and Capturing Methods

Numerical methods to track or capture interfaces can be classified into two fundamental categories: interface tracking methods[4] and interface capturing methods[5]. *Interface tracking* methods represent the interface between two immiscible fluids explicitly, either adapting the Eulerian fluid mesh in a way to resemble the interface, typically called *Moving-Mesh* methods [187, 237], or introducing an immersed boundary representing the interface, *e.g. Front-Tracking* methods [184, 235, 240]. Moving-mesh methods are able to reproduce the jump condition of fluid properties at the interface accurately and are able to precisely impose the force acting at the interface due to surface tension. The required mesh movement and remeshing in cases of large interface movements makes the implementation complex, particularly with parallelised computer systems. Consequently, problems with very high interface curvature and moderate interface movement, such as oscillating microbubbles [261], are particularly suited for moving-mesh methods. Front-tracking methods do not require to alter the Eulerian fluid mesh but the adaption of the surface mesh representing the interface makes the tracking of significant interface deformation difficult. Front-tracking methods require additional numerical models to simulate interface

[4]Also called *surface methods.*

[5]Also called *volume methods* or *volume tracking methods.*

topology changes, such as breakup or coalescence.

Interface capturing methods rely on an implicit interface representation, which means the explicit position and shape of the interface is not known. The most important interface capturing methods are *Volume of Fluid* (VOF) methods [97, 269] and *Level-Set* (LS) methods [169, 225]. VOF methods compute the evolution of an interface indicator function, typically the volume fraction, which is advected based on the underlying flow field. VOF methods inherently conserve mass but suffer from the absence of an explicit interface representation and the related inaccuracies of calculating interface curvatures with the available data. Level-set methods use a distance function from the interface, assigning the zero level-set to the interface, and advect the distance function with the local fluid velocity. LS methods provide accurate results when the interface is advected parallel to one of the coordinate axes but suffer from mass loss if the interface is strongly deformed or in flows with considerable vorticity. VOF and LS methods are in principle capable of capturing interface breakup and coalescence without additional models, although very fine meshes are required to diminish numerical artifacts and in case of coalescence it must be assured that coalescence is physically plausible.

2.3.2. The Volume of Fluid Method

The Volume of Fluid (VOF) method [97] is adopted for the presented work to distinguish two incompressible, immiscible fluids. The VOF method is among the most widely used methods for two-phase flows, because it is easy to implement, applicable to arbitrary meshes and mass conservative. The VOF method assigns a volume fraction γ, typically called *colour function*, to every mesh element, representing the local volume fraction, defined as

$$\gamma(\boldsymbol{x}, t) = \begin{cases} 0 & \text{fluid } A \\ 1 & \text{fluid } B \ . \end{cases} \tag{2.42}$$

Thus, a mesh element holding a colour function value of $0 < \gamma < 1$ contains an interface. The colour function γ is advected based on the underlying flow field by the transport equation

$$\frac{\partial \gamma}{\partial t} + u_i \frac{\partial \gamma}{\partial x_i} = 0 \ . \tag{2.43}$$

The fluid properties, *i.e.* density and viscosity in isothermal flows, are discontinuous at the interface and are defined based on the colour function as

$$\rho = \rho_A (1-\gamma) + \rho_B \, \gamma \tag{2.44}$$

$$\mu = \mu_A (1-\gamma) + \mu_B \, \gamma \ . \tag{2.45}$$

The accurate advection of the colour function based on the underlying flow is essential for the outcome of VOF simulations. Two fundamental types of VOF methods may be distinguished: compressive methods and geometric methods. *Compressive VOF* methods discretise the VOF transport equation (Eq. 2.43) with standard numerical differencing schemes, as for instance the schemes presented in Section 2.2. However, low-order advection schemes are not suitable as they lead to significant smearing of the interface, whereas high-order schemes typically result in numerical oscillations and wrinkling of the interface. Compressive VOF methods, therefore, use specifically designed spatial advection schemes based on a donor-acceptor approach [39, 108, 123, 163, 239]. A donor (upwind) and an acceptor (downwind) cell are assigned for every mesh face with respect to the underlying flow field. Based on the angle between the local interface normal vector and the normal vector of the cell face, donor-acceptor schemes blend between compressive downwind and diffusive upwind schemes. The temporal advection is commonly discretised with a second-order accurate scheme, such as the Crank-Nicolson scheme or the Second-Order Backward Euler scheme [39, 161, 238], as first-order schemes are too diffusive to maintain the sharpness of the interface. Alternatively, the interface is transported using a *geometric method*, advecting an explicit representation of the interface which is reconstructed from the colour function field. The most notable state-of-the-art reconstruction methods are *piecewise linear interface construction* (PLIC) methods [9, 133, 180, 199, 205, 247, 269] and parabolic reconstruction methods [197]. The explicit interface is geometrically fitted to the VOF volume fraction field and advected in an Eulerian, Lagrangian or mixed Eulerian-Lagrangian fashion [150].

The major advantage of compressive VOF methods compared to geometric methods are the straightforward implementation, the computational efficiency and, crucially, the applicability to arbitrary meshes. However, even compressive VOF methods based on very sophisticated donor-acceptor schemes suffer from considerable numerical diffusion on unstructured meshes. Geometric methods advect the interface generally very accurately but are, apart from a few exceptions [101, 149, 150, 155], exclusively available on Cartesian meshes. Moreover, the implementation of geometric methods is more complex than compressive methods and the required computational effort is higher.

For the research presented in this thesis, given the applicability on unstructured meshes and the computational efficiency, a compressive VOF method is adopted to capture the interface. The numerical discretisation of the VOF transport equation (Eq. 2.43) is discussed in Section 3.4. The numerical diffusion induced by the interface advection, an important issue of compressive VOF methods on unstructured meshes, is further investigated as part of the case studies presented in Chapter 6.

2.3.3. Surface Tension

Surface tension is a property of liquid interfaces caused by the cohesion of the molecules of the liquid, which enables a liquid interface to resist external forces. A liquid naturally strives towards the state of minimum potential energy. As a result of the missing neighbour molecule of the same kind, molecules at the liquid interface have a higher energy than molecules inside the liquid which are surrounded by molecules of the same kind. Thus, the liquid minimises the number of molecules at the interface and, therefore, the interface area, creating an internal pressure which is known as surface tension. The force due to surface tension restores the balance between repulsive and attractive forces, which is broken as a result of the missing neighbour molecule [146]. From a thermodynamic point of view, surface tension is the work necessary to increase the surface area of a liquid interface by a given amount, *i.e.* the work done per unit area. Every fluid pair separated by an interface, for instance air and water, is characterised by a surface tension coefficient σ, which represents the factor defining the magnitude of the surface tension acting at their interface. Considering an interface without external forces such as gravity, surface tension is defined by the Young-Laplace equation [13]

$$\Delta p = p_i - p_o = \sigma \left(\frac{1}{r_1} + \frac{1}{r_2} \right) = \sigma \, \kappa \ , \tag{2.46}$$

where κ denotes the mean curvature of the interface, r_1 and r_2 are the principal curvature radii of the three-dimensional interface and p_i and p_o are the pressure inside and outside the interface, respectively. *Inside* the interface refers to the concave side of the interface, which yields the higher pressure, and *outside* refers to the convex side of the interface, holding the lower pressure. The article of Navascues [166] provides a detailed review of the derivation of surface tension from a macroscopic, *i.e.* mechanical and thermodynamic, and a microscopic viewpoint.

In a landmark paper, Brackbill *et al.* [19] proposed the *Continuum Surface Force* (CSF) model to numerically describe the effect of surface tension. The CSF model essentially transforms the molecular surface tension into a volumetric source term, spreading the force acting at the interface due to surface tension, henceforth simply referred to as *surface force*[6], over a transition region of finite thickness. The model is constructed in order to fulfil the Young-Laplace equation presented in Eq. 2.46. The volumetric surface force acting at the interface for a general interface indicator function C according to the CSF model is [19]

$$f_{s,i} = \sigma \, \kappa \, \frac{1}{\Delta C} \frac{\partial C}{\partial x_i} \ , \tag{2.47}$$

where $\Delta C = |C_A - C_B|$ is the jump of the indicator function distinguishing fluids A and B. With respect to the VOF method presented in Section 2.3.2 and its colour function γ, which ranges from 0 to 1, the surface force per unit volume given in Eq. 2.47 simplifies to

$$f_{s,i} = \sigma \, \kappa \, \frac{\partial \gamma}{\partial x_i} \ . \tag{2.48}$$

A similar definition is used for level-set methods, where the indicator function C in Eq. 2.47 represents the level-set distance function. The interface curvature κ is defined as

$$\kappa = -\frac{\partial m_i}{\partial x_i} \ , \tag{2.49}$$

where $\boldsymbol{m}$ is the unit normal vector of the interface, which is the normalised first derivative of the colour function, given as

$$\boldsymbol{m} = \left| \frac{\partial x_i}{\partial \gamma} \right| \frac{\partial \gamma}{\partial x_i} \ . \tag{2.50}$$

The volumetric surface force $\boldsymbol{f}_s$ defined in Eq. 2.47 is included as a source term in the

[6]The terms *surface tension force* and *interface force* are also frequently used in the relevant literature.

momentum equation (Eq. 2.10), which then becomes

$$\rho \left(\frac{\partial u_i}{\partial t} + u_j \frac{\partial u_i}{\partial x_j} \right) = - \frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j} \left[\mu \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) \right] + f_{g,i} + f_{s,i} \ . \tag{2.51}$$

A key problem related to the CSF method is the occurrence of unphysical currents around the interface, so-called *parasitic currents*. In general, parasitic currents are caused by the discretisation of a molecular surface force on a macroscopic scale. Lafaurie *et al.* [123] found the magnitude of parasitic currents for a spherical fluid particle in a stationary fluid to satisfy the relationship $|\boldsymbol{u}| = C\,\sigma/\mu$, where C is a coefficient dependent on the numerical method. Parasitic currents represent a substantial problem for surface-tension-dominated flows, where in extreme cases parasitic currents can be large enough to breakup the interface. Evidently, parasitic currents develop either due to a local imbalance between the pressure gradient at the interface and the surface force or due to an inaccurate estimation of the interface curvature. An imbalance between pressure gradient and surface force can be avoided by a careful implementation, as Francois *et al.* [71] and Mencinger and Žun [154] demonstrated successfully. Both studies proved that if a constant curvature value is imposed at the interface of a circular stationary two-dimensional fluid particle and the implementation of surface force and pressure gradient match each other, parasitic currents effectively vanish as their magnitude reduces to solver tolerance or machine precision, respectively. The discretisation of a fully-coupled balanced-force numerical framework for arbitrary meshes is the topic of Chapter 3.

The second key issue when simulating interfacial flows, particularly when using a VOF method, is the evaluation of the interface normal vector and the interface curvature. In order to be able to use classical finite volume methods to determine the interface normal vector, the colour function γ, or any other interface indicator function for this purpose, has to be continuous and differentiable [240]. To determine the interface curvature, the colour function has to be twice differentiable. This requires the colour function γ to change smoothly over a finite distance rather than suddenly. The colour function, however, is abruptly varying in space, which leads to substantial aliasing errors in the evaluation of gradients [34]. The calculation of the interface curvature is investigated in detail in Chapter 4.

2.3.4. Conservation Issue in Two-Phase Flows

The conservation laws for mass and momentum which build the mathematical foundation for the prediction of fluid flows, discussed in Section 2.1, require special attention with respect to two-phase flows. Considering two incompressible, immiscible fluids without evaporation or condensation, the fluid velocity at either side of the interface is equal [27, 185, 189, 202, 205, 206, 262, 264]. For example, the volume flux upstream and downstream of the interface must be identical for a liquid fluid front travelling with a constant velocity through a straight pipe. Thus, the momentum $\rho\boldsymbol{u}$ of the lighter fluid is smaller than the momentum of the heavier fluid. This leads to a momentum defect which represents a viable problem for the numerical modelling of interfacial flows with large density ratios. Figure 2.7 schematically illustrates the momentum exchange in the interface region. In order to locally conserve momentum, $(\rho \boldsymbol{u})_A = (\rho \boldsymbol{u})_B$, the lighter fluid must accelerate and the denser fluid must decelerate. As a result, a non-zero density gradient, *i.e.* $\rho_A/\rho_B \neq 1$, would change the interface thickness with respect to the CSF model. An interface thickness which is dependent on the density ratio, however, is physically implausible and violates the CSF method [19]. The conservation of mass is not straightforward either. Although mass is conserved for a closed domain as well as for an infinitesimally small control volume, conservation of mass for every individual mesh element is numerically not feasible with respect to interfacial flows. Examining the example in Figure 2.7, the element containing the interface has a higher mass entering the element through face e than mass leaving the element through face f, as a result of the constant volume flux. Enforcing mass conservation for mesh elements containing the interface would accelerate the flow as a result of the density jump, which is unphysical.

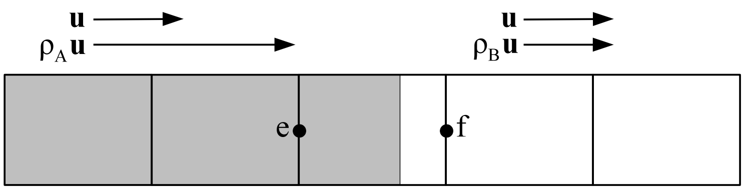


Figure 2.7.: Schematical illustration of the momentum transport across a liquid front separating fluids A (grey) and B (white), travelling through a two-dimensional pipe at constant velocity.

Implementing the momentum equation in its non-conservative form can assure continuity and a constant interface thickness. The non-conservative form of the momentum equation (Eq. 2.51) is mathematically identical to its conservative counterpart but has different numerical implications. The non-conservative form of the momentum equation is given as

$$\frac{\partial u_i}{\partial t} + u_j \frac{\partial u_i}{\partial x_j} = -\frac{1}{\rho}\frac{\partial p}{\partial x_i} + \frac{1}{\rho}\frac{\partial}{\partial x_j}\left[\mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)\right] + g_i + \frac{f_{s,i}}{\rho} \ . \tag{2.52}$$

Applying the non-conservative momentum equation, only the element-centred density ρ is used to calculate the mass flow through faces bounding the element. Therefore, the mass flow at face f bounding element P is defined as

$$\dot{m}_f = (\boldsymbol{u}_f \cdot \boldsymbol{n}_f)\, A_f\, \rho_P \ . \tag{2.53}$$

As a result, the fluid in element P is not accelerated by a non-zero density gradient. Similarly, only the density of the present time instant is used for the discretisation of the transient term of the non-conservative momentum equation.

3. Numerical Framework

This chapter presents and discusses the numerical framework developed to solve incompressible, isothermal single-phase and two-phase flows on unstructured meshes. In what follows, the relevant discretisation schemes and techniques for the governing equations are devised and discussed, based on the mathematical and numerical foundation presented in Chapter 2. Firstly, the fundamental architecture of the numerical framework is discussed in Section 3.1 and the discretisation of the momentum equations is presented in Section 3.2. Subsequently, Section 3.3 presents the derivation and discretisation of the continuity constraint and the advecting velocity, using a momentum interpolation method. The devised advecting velocity includes a revised implementation of the surface force as well as a novel density weighting, both presented in Section 3.3.4. Moreover, the accurate application of the momentum interpolation method on non-Cartesian meshes is explained, particularly emphasising the necessity of a symmetrical interpolation of the pressure gradient and the body forces to satisfy the filtering properties of the momentum interpolation method. In Section 3.4 the used compressive VOF methodology is described, followed by a short description of the implementation of the boundary conditions in Section 3.5 and the discretisation and implementation of the gradient evaluation in Section 3.6. The solution procedure of the flow equation system and the interface advection is outlined in Section 3.7. The numerical framework is validated in Section 3.8. This validation also includes a test case to demonstrate that the presented balanced-force numerical framework is applicable to moving interfaces, a numerical capability and a test case not previously reported in the literature. The chapter is concluded with a short summary in Section 3.9.

3.1. Basis of the Numerical Framework

This section presents and examines the basic architecture of the developed numerical framework. The numerical framework is predicated on four cornerstones:

1. applicability to unstructured meshes,
2. a collocated variable arrangement,
3. an implicit discretisation, and
4. a fully-coupled system of equations to describe the flow.

The variable arrangement on the computational mesh plays an important role for the discretisation and implementation of the numerical framework. The variables can be arranged in two ways: staggered or collocated. The reason why the type of variable arrangement is important can be demonstrated by a finite difference example, considering a one-dimensional equidistant mesh with mesh spacing Δx. On a *staggered* mesh, pressure is typically stored at cell centres whereas velocity is stored at face centres, meaning that pressure and velocity have different control volumes. Discretising the momentum equation at the positions where velocity is stored, for instance at face w with its adjacent cells W and P, the discretised pressure gradient term of the momentum equation (Eq. 2.9) is

$$\left.\frac{\partial p}{\partial x}\right|_w \approx \frac{p_P - p_W}{\Delta x} + \mathcal{O}(\Delta x^2) \; . \tag{3.1}$$

No interpolation of pressure or velocity is required, all values are taken as-is. However, the implementation of a staggered variable arrangement becomes particularly cumbersome for unstructured meshes [259]. The complexity of a staggered variable arrangement for an unstructured mesh can be bypassed with a *collocated* variable arrangement. Using a collocated mesh, pressure and velocity are stored at cell centres. Thus, the pressure gradient in the discretised momentum equation becomes

$$\left.\frac{\partial p}{\partial x}\right|_P \approx \frac{p_e - p_w}{\Delta x} = \frac{p_P + p_E}{2\Delta x} - \frac{p_P + p_W}{2\Delta x} = \frac{p_E - p_W}{2\Delta x} + \mathcal{O}(\Delta x^2) \; . \tag{3.2}$$

The pressure gradient at cell centre P becomes independent of the pressure at P on a collocated mesh and is only determined by the neighbours of cell P. This decoupling of pressure and its gradient typically leads to unphysical pressure oscillations [174]. An efficient and the most widely used method to prevent such pressure oscillations is the interpolation method pioneered by Rhie and Chow [198], which is presented and discussed in more detail in Section 3.3.

For a three-dimensional flow, the three momentum equations contain four unknown variables; the three velocity components and pressure. Thus, the equation system is underdefined and an additional relationship between velocity and pressure is required. Two approaches are available to tackle the problem of the underdefined equation system: segregated algorithms and coupled algorithms. In *segregated* algorithms the velocity and pressure calculation are separated, calculating the pressure based on the velocity distribution using a pressure-correction method in an iterative procedure. The velocity field is approximated by solving the momentum equations using the pressure gradient of the previous iteration or an initial guess. Subsequently, the new pressure distribution is explicitly calculated based on the approximated velocity field. The fluxes and pressure gradients are updated thereafter and the iterative loop starts again until a sufficiently small residual variation is reached. The most widely used pressure-correction methods are the SIMPLE [174, 176], SIMPLER [174] and PISO [103] algorithms. In the context of two-phase flows with appreciable surface tension and density ratio, the pressure Poisson equation

$$\nabla \cdot \frac{1}{\rho} \nabla p = \frac{1}{\Delta t} \nabla \cdot \boldsymbol{u} \; , \tag{3.3}$$

typically used in the aforementioned pressure correction procedures, becomes ill-conditioned due to the discontinuous pressure and density fields in conjunction with the continuous velocity field [58, 86], and as a result the pressure-correction step becomes computationally very expensive [194] and the numerical stability suffers [248].

An alternative to segregated flow solvers using pressure-correction methods is to construct and solve a *fully-coupled* implicit equation system, as for instance in [22, 31, 32, 40, 51, 246, 248]. Considering a fully-coupled equation system, the equations describing the flow by means of its primitive variables are solved simultaneously. This requires a fourth equation to describe the relationship between pressure and velocity to close the equation system. Since the continuity equation (Eq. 2.2), which is naturally the fourth equation of the set of equations describing incompressible, isothermal flows in three dimensions, does not contain pressure but is merely an additional constraint on the velocity field, the major issue is to define a fourth equation which is independent of the momentum equations and contains pressure as well as velocity. As shown in Section 3.3, an equation which fulfils the continuity equation and provides a strong pressure-velocity coupling can be derived using the interpolation method first proposed by Rhie and Chow [198]. This equation, from hereon simply referred to as *continuity constraint*, provides an additional relationship between pressure and velocity as a result of the implicitly included pressure-velocity coupling. Therefore, the equation system describing the flow has the same number of independent equations as it has unknown variables and can be written in its generic form as

$$\begin{pmatrix} \text{x-momentum equation} \\ \text{y-momentum equation} \\ \text{z-momentum equation} \\ \text{continuity constraint} \end{pmatrix} \cdot \begin{pmatrix} u \\ v \\ w \\ p \end{pmatrix} = \boldsymbol{b} \ , \tag{3.4}$$

where u, v and w are the velocity components, p is pressure and $\boldsymbol{b}$ is the right-hand side vector of the equation system. The solution of a fully-coupled implicit equation system requires more computational memory than a segregated solution approach and the solution of the fully-coupled system of equations also suffers from similarly ill-conditioned equations as the solution of the pressure-correction equation of segregated methods and the related increase in required computational resources. However, the fully-coupled implicit approach is more robust than a segregated solution methodology as a result of the strong, implicit pressure-velocity coupling [51]. The strong pressure-velocity coupling is particularly desirable for the simulation of two-phase flows, because of the large, quasi-discontinuous pressure jump resulting from surface tension and the potentially large density ratio between the interacting fluids. Additionally, continuity is inherently satisfied because the same velocities are used for the momentum equations and the continuity constraint.

3.2. Momentum Equations

An individual momentum equation is solved for each of the three coordinate axes of the spatial domain. In order to prohibit the acceleration of the flow due to non-zero density gradients at interfaces in two-phase flows, the momentum equations are implemented in their non-conservative form, as defined in Eq. 2.52. In the following sections, the discretisation of each term of Eq. 2.52 is presented and, where applicable, the differences between single-phase and two-phase flows are discussed. It is worth mentioning that the presented discretisation methods equally apply to all three momentum equations.

3.2.1. Transient Term

The transient term of the momentum equation is discretised using the Second-Order Backward Euler scheme, presented in Section 2.2.5. For a variable time-step Δt, the Second-Order Backward Euler scheme for fluid variable ϕ can be derived by developing the Taylor series of ϕ at the previous time instant $t - \Delta t_1$

$$\phi^{t-\Delta t_1} = \phi^t - \Delta t_1 \frac{\partial \phi^t}{\partial t} + \frac{\Delta t_1^2}{2}\frac{\partial^2 \phi^t}{\partial t^2} + \mathcal{O}(\Delta t^3) \tag{3.5}$$

and at the previous-previous time instant $t - \Delta\tau$

$$\phi^{t-\Delta \tau} = \phi^t - \Delta \tau \frac{\partial \phi^t}{\partial t} + \frac{\Delta \tau^2}{2}\frac{\partial^2 \phi^t}{\partial t^2} + \mathcal{O}(\Delta t^3) \ , \tag{3.6}$$

where Δt_1 is the present time-step, Δt_2 is the previous time-step and $\Delta\tau = \Delta t_1 + \Delta t_2$. After multiplication by $\Delta t_1^2/\Delta\tau^2$, Eq. 3.6 becomes

$$\frac{\Delta x_1^2}{\Delta \tau^2}\phi^{t-\Delta \tau} = \frac{\Delta x_1^2}{\Delta \tau^2}\phi^t - \frac{\Delta x_1^2}{\Delta \tau}\frac{\partial \phi^t}{\partial t} + \frac{\Delta t_1^2}{2}\frac{\partial^2 \phi^t}{\partial t^2} + \mathcal{O}(\Delta t^3) \ . \tag{3.7}$$

Subtracting Eq. 3.7 from Eq. 3.5 then gives

$$\phi^{t-\Delta t_1} - \frac{\Delta t_1^2}{\Delta \tau^2}\phi^{t-\Delta \tau} = \phi^t \left(1 - \frac{\Delta t_1^2}{\Delta \tau^2}\right) + \frac{\partial \phi^t}{\partial t}\left(\frac{\Delta t_1^2}{\Delta \tau} - \Delta t_1\right) + \mathcal{O}(\Delta t^2) \tag{3.8}$$

and after rearranging, the temporal derivative of ϕ follows as [144]

$$\frac{\partial \phi^t}{\partial t} = \left(\frac{1}{\Delta t_1} - \frac{\Delta \tau}{\Delta t_1^2}\right)\left[\left(1 - \frac{\Delta t_1^2}{\Delta \tau^2}\right)\phi^t - \phi^{t-\Delta t_1} + \frac{\Delta t_1^2}{\Delta \tau^2}\phi^{t-\Delta \tau}\right] + \mathcal{O}(\Delta t^2) \ . \tag{3.9}$$

Eq. 3.9 is identical to Eq. 2.41 if $\Delta t_1 = \Delta t_2$. As a result, the discretisation of the transient term of the non-conservative momentum equation applying the Second-Order Backward Euler scheme is given as

$$\int_{V_P} \rho \frac{\partial u_i}{\partial t}\, dV \approx \rho_P^t \left(\frac{1}{\Delta t_1} - \frac{\Delta \tau}{\Delta t_1^2}\right)\left[\left(1 - \frac{\Delta t_1^2}{\Delta \tau^2}\right) u_{i,P}^t - u_{i,P}^{t-\Delta t_1} + \frac{\Delta t_1^2}{\Delta \tau^2} u_{i,P}^{t-\Delta \tau}\right] V_P \ . \tag{3.10}$$

3.2.2. Convection Term

Following the discretisation of fluid variable ϕ, discussed in Section 2.2.3, the convection term of the momentum equation is discretised as

$$\int_{V_P} \rho\, u_j \frac{\partial u_i}{\partial x_j}\, dV \approx \sum_f \dot{m}_f \ u_{i,f} \ , \tag{3.11}$$

where $u_{i,f}$ is the face velocity and $\dot{m}_f$ is the mass flux through face f. Because of the non-conservative implementation of the momentum equations, the mass flux is implemented as defined in Eq. 2.53. Using an upwind-based formulation, the face velocity is defined as

$$u_{i,f} = u_{i,U} + \psi\, \delta_D \left(u_{i,D} - u_{i,U}\right) . \tag{3.12}$$

The upwind cell U is the cell located upstream and adjacent to face f, and, accordingly, the downwind cell D is the cell located downstream and adjacent to face f. The flux limiter ψ defines the applied differencing scheme, *i.e.* $\psi = 0$ for first-order upwind differencing and $\psi = 1$ for central differencing, and δ_D is the inverse distance interpolation coefficient, defined as

$$\delta_D = \frac{|\boldsymbol{r}_U|}{|\boldsymbol{r}_U| + |\boldsymbol{r}_D|} , \tag{3.13}$$

where $\boldsymbol{r}_U$ is the vector connecting the upwind cell centre U with face centre f and $\boldsymbol{r}_D$ denotes the vector connecting the downwind cell centre D with face centre f. The velocity at face centre f is implemented implicitly, whereas the mass flow of the previous iteration is used. For the work presented in this thesis the central differencing (CD) scheme is used, as it diminishes numerical diffusion and is second-order accurate.

3.2.3. Viscous Term

Following the discretisation and the non-orthogonal correction presented for the generic diffusion term in Section 2.2.4, the viscosity term of the incompressible momentum equation is discretised as

$$\int_{V_P} \frac{\partial}{\partial x_j}\left[\mu_f \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)\right] dV \approx \sum_f \mu_f \left[\left(\underbrace{\alpha_f \frac{u_{i,Q} - u_{i,P}}{\Delta s}}_{\text{orthogonal}} + \underbrace{\left.\overline{\frac{\partial u_i}{\partial x_j}}\right|_f (\boldsymbol{n}_f - \alpha_f\, \boldsymbol{s}_f)}_{\text{non-orthogonal}}\right) + \left.\overline{\frac{\partial u_j}{\partial x_i}}\right|_f \boldsymbol{n}_f\right] A_f , \tag{3.14}$$

including a non-orthogonal correction which decomposes the first term of the left-hand side of Eq. 3.14 into an orthogonal and a non-orthogonal part, as previously presented in Eq. 2.34. The second term of the discretised viscous term represents the average of the cell-centred gradients at the face, defined as

$$\left.\overline{\frac{\partial u_i}{\partial x_j}}\right|_f (\boldsymbol{n}_f - \alpha_f\, \boldsymbol{s}_f) = \left[(1-\delta) \left.\frac{\partial u_i}{\partial x_j}\right|_P + \delta \left.\frac{\partial u_i}{\partial x_j}\right|_Q\right] (\boldsymbol{n}_f - \alpha_f\, \boldsymbol{s}_f) . \tag{3.15}$$

The interpolation to the face centre is weighted by the inverse distance to face f following the work of Demirdžić [49], with δ being the interpolation coefficient as discussed in Section 2.2.2. The third term on the ride-hand side of Eq. 3.14 is averaged in the same way. As defined in Section 2.2.4, the scaling factor α_f is defined as $\alpha_f = (\boldsymbol{n}_f \cdot \boldsymbol{s}_f)^{-1}$.

The implementation of the viscous term as defined in Eq. 3.14 is identical for single-phase flows and two-phase flows. However, if two interacting fluids hold different viscosities, the interpolation of the viscosity at face centres μ_f requires special consideration in two-phase flows. Two basic interpolation methods can be distinguished [119], the arithmetic mean

$$\mu_f = \frac{\mu_P + \mu_Q}{2} \tag{3.16}$$

and the harmonic mean

$$\mu_f = \frac{2}{\mu_P^{-1} + \mu_Q^{-1}} \ . \tag{3.17}$$

Since the arithmetic mean gives equal weight to the fluids on either side of the face, the fluid with smaller viscosity is accelerated. The harmonic mean, on the other hand, gives more weight to the contribution of the less viscous fluid, avoiding a considerable acceleration of the less viscous fluid [67, 190, 200]. The harmonic mean is used for all simulations presented in this thesis as it provides a meaningful interpolation of the face viscosity and reduces the impact of large viscosity ratios.

3.2.4. Pressure Term

Similar to the other spatial terms, the pressure term of the momentum equation is discretised using the Gauss theorem and the midpoint rule. The accuracy of the interpolation and the stability of the numerical implementation is increased by means of a gradient-based skewness correction as presented in Eq. 2.21, using explicitly calculated gradients. Therefore, the pressure discretisation follows as

$$\int_{V_P} \frac{\partial p}{\partial x_i} \, dV \approx \sum_f \left[(1-\delta)\, p_P + \delta\, p_Q + \overline{\nabla p}\big|_f \, \boldsymbol{r}_f \right] n_{i,f} \, A_f \ , \tag{3.18}$$

where $n_{i,f}$ is the i-th component of the outward-pointing unit normal vector of face f and $\boldsymbol{r}_f$ is the vector from the interpolation point to the actual face centre. The interpolated pressure gradient at the face is defined as

$$\overline{\nabla p}\big|_f = (1-\delta)\, \nabla p|_P + \delta\, \nabla p|_Q \ . \tag{3.19}$$

The interpolation coefficient δ is determined using the inverse distance weighting defined in Eq. 2.19.

3.2.5. Gravity

The volumetric gravity force $\boldsymbol{f}_g = \rho \boldsymbol{g}$ is implemented by means of a source term $\boldsymbol{S}_g$, following the work of van Wachem *et al.* [248]. For single-phase flows on Cartesian meshes the gravity force can be implemented in a straightforward manner, integrating the volumetric gravity force over the control volume

$$F_{g,i} = \int_{V_P} f_{g,i} \, dV = \int_{V_P} \rho \, g_i \, dV \approx \rho \, g_i \, V_P \ . \tag{3.20}$$

On the other hand, because gravity results in a pressure gradient, the gravity force should be discretised using a similar computational stencil as for the pressure gradient. Therefore, the gravity force is represented by source term $\boldsymbol{S}_g$, defined at cell centre P as

$$f_{g,i} \approx S_{gi,P} = \frac{1}{V_P} \sum_f (\boldsymbol{g} \cdot \boldsymbol{a}_f)\, \rho_f \, n_{f,i} \, A_f \ , \tag{3.21}$$

where subscript f denotes all faces bounding cell P, A_f is the area of face f and $\boldsymbol{a}_f = \boldsymbol{x}_f - \boldsymbol{x}_P$ is the co-variant cell face vector, with $\boldsymbol{x}_f$ and $\boldsymbol{x}_P$ being the position vector of face centre f and cell centre P, respectively.

The source term describing gravity requires special consideration if non-zero density gradients occur, as it is typically the case in two-phase flows because the two fluids have different densities. Similar to viscous stresses, the gravity force is not constrained by conservation laws. Thus, following the same reasoning as for the interpolation of the viscosity to face centres, density is interpolated to face centres using the harmonic mean [67, 159], defined as

$$\rho_f = \frac{2}{\rho_P^{-1} + \rho_Q^{-1}} \ , \tag{3.22}$$

in order to avoid a strong acceleration of the lighter fluid near the interface.

3.2.6. Surface Tension

In the presented finite volume framework, surface force is modelled using the CSF model (see Eq. 2.48) and discretised as

$$F_{s,i} = \int_{V_P} f_{s,i} \, dV \approx \sigma \, \kappa_P \left. \frac{\partial \gamma}{\partial x_i} \right|_P V_P \ . \tag{3.23}$$

Eq. 3.23 is included explicitly in the flow equation system. The surface force, which only acts in the interface region, results in a pressure jump across the interface. Thus, as mentioned previously in Section 2.3.3, the pressure gradient integrated over the volume has to precisely replicate the surface force on a discrete level to avoid numerical artifacts. The gradient of the colour function is, therefore, discretised in the same way as the pressure gradient term in Eq. 3.18 and follows as

$$\left. \frac{\partial \gamma}{\partial x_i} \right|_P = \frac{1}{V_P} \sum_f \Big[(1-\delta)\, \gamma_P + \delta \, \gamma_Q + \Big((1-\delta) \, \nabla\gamma|_P + \delta \, \nabla\gamma|_Q \Big) \, \boldsymbol{r}_f \Big] \, n_{i,f} \, A_f \ . \tag{3.24}$$

The interpolation coefficient δ is determined by Eq. 2.19 and the colour function gradient is evaluated explicitly applying the iterative procedure described in Section 3.6. The evaluation of interface curvature κ is examined in Sections 4.2.2 and 4.3.

3.3. Continuity Constraint

The content of this section has in parts been published in:

[54] Denner, F. and van Wachem, B.G.M.: Force-balancing at moving surface-tension-dominated interfaces on collocated unstructured meshes. 8th International Conference on Multiphase Flow (ICMF 2013), 26 - 31 May 2013, Jeju, Korea.

[55] Denner, F. and van Wachem, B.G.M.: Fully-coupled balanced-force VOF framework for arbitrary meshes with least-squares curvature evaluation from volume fraction. Numerical Heat Transfer, Part B: Fundamentals, accepted for publication. DOI: 10.1080/10407790.2013.849996

As briefly discussed in Section 3.1, the equation system of the discretised momentum equations describing the fluid is underdefined, with three momentum equations containing four unknown variables: the three Cartesian velocity components and pressure. Using a fully coupled approach, a fourth equation containing the four unknown variables is required. This problem can be overcome by deriving a fourth equation using the interpolation approach first proposed by Rhie and Chow [198], which was initially introduced to prevent pressure-velocity decoupling. This approach introduces a third-order pressure gradient term to the interpolation of face velocities. This third-order pressure gradient term effectively dampens out spurious oscillations in the pressure field, arising from a collocated variable arrangement. Because the interpolation approach of Rhie and Chow can be derived from the discretised momentum equation, it is generally referred to as *momentum interpolation method* or *momentum weighted interpolation method*. In the proposed numerical framework the face velocities calculated with the momentum interpolation method are then used to formulate a continuity constraint for every mesh cell which preserves continuity and assures a strong pressure-velocity coupling.

Several numerical frameworks successfully applied a fully-coupled approach based on the momentum interpolation method [22, 31, 32, 40, 248]. Cubero and Fueyo [31, 32], Darwish *et al.* [40, 42] and Chen and Przekwas [22] published fully-coupled numerical frameworks for unstructured meshes, devising a continuity constraint using the original formulation of the momentum interpolation method as introduced by Rhie and Chow [198]. The original formulation of the momentum interpolation method, however, only considers pressure gradient terms but neither body forces, which significantly affect the pressure gradient, nor transient terms. Further developments of the momentum interpolation method also include body forces [26, 154, 246, 248], ensuring a balance between the pressure gradient and body forces, as well as transient terms [25, 31, 154, 173, 246], diminishing the time-step dependency of the pressure-velocity coupling.

In what follows, an advecting velocity u_f^n at face centres f is derived using the momentum interpolation method. The advecting velocity is then used to formulate the continuity constraint

$$\frac{\partial u_i}{\partial x_i} \approx \frac{1}{V_P} \sum_f \left(\boldsymbol{u}_f \cdot \boldsymbol{n}_f\right) A_f \approx \frac{1}{V_P} \sum_f u_f^n \, A_f = 0 \ , \tag{3.25}$$

where f denotes all bounding faces of a given mesh cell P, $\boldsymbol{n}_f$ is the outward-pointing unit normal vector of face f and A_f is the area of face f. The presented continuity constraint has five distinct characteristics:

1. it provides a strong pressure-velocity coupling,
2. it assures an exact balance between pressure gradient and body forces,
3. it fulfils the continuity equation (Eq. 2.2),
4. it can handle arbitrarily large density ratios, and
5. it is applicable to unstructured meshes.

The advecting velocity u_f^n constructed for the continuity constraint is also used to define the mass flux, thus, assuring a consistently defined convection term of the momentum equations and further strengthen the coupling of velocity and pressure. The mass flux, previously defined in Eq. 2.53, is then given as

$$\dot{m}_f = (\boldsymbol{u}_f \cdot \boldsymbol{n}_f)\, A_f\, \rho_P \approx u_f^n\, A_f\, \rho_P \ . \tag{3.26}$$

In the following section the general form of the advecting velocity u_f^n is derived. Subsequently, the modifications for arbitrary meshes, the treatment of gravity and the implications for two-phase flows are explained.

3.3.1. Derivation of the Advecting Velocity

The general form of the advecting velocity u_f^n is derived for an equidistant Cartesian mesh using a first-order temporal discretisation. The temporal discretisation can be extended to high-order methods without difficulty [31]. Discretising the momentum equation (Eq. 2.9) along the x-coordinate axis using a standard finite volume approach and neglecting body forces leads to

$$\frac{\rho V_P}{\Delta t}\left[u_P - u_P^{t-\Delta t}\right] + a_P u_P - \sum_Q a_Q u_Q = -\left.\frac{\partial p}{\partial x}\right|_P V_P \ , \tag{3.27}$$

where P denotes the cell under consideration and Q represents the neighbours of cell P, as illustrated in Figure 2.2a. The coefficient a represents the combined implicit coefficient of the convective and viscous terms of the discretised momentum equation at the current time instant as defined Section 3.2. After dividing by a_P, Eq. 3.27 becomes

$$\left[1 + \frac{\rho V_P}{a_P \Delta t}\right] u_P = \frac{1}{a_P}\sum_Q a_Q u_Q - \frac{V_P}{a_P}\left.\frac{\partial p}{\partial x}\right|_P + \frac{\rho V_P}{a_P \Delta t} u_P^{t-\Delta t} \ , \tag{3.28}$$

which can be further simplified with the abbreviations

$$c = \frac{\rho}{\Delta t} \tag{3.29}$$

$$d_P = \frac{V_P}{a_P} \tag{3.30}$$

$$\tilde{u}_P = \frac{1}{a_P}\sum_Q a_Q u_Q \ , \tag{3.31}$$

following as

$$[1 + c\ d_P]\, u_P = \tilde{u}_P - d_P\ \left.\frac{\partial p}{\partial x}\right|_P + c\ d_P\ u_P^{t-\Delta t}\ . \tag{3.32}$$

Because the advecting velocity is needed at face centres and not at cell centres, an analogous equation is written at the centre of face f shared by cells P and Q, given as

$$[1 + c\ d_f]\, u_f = \tilde{u}_f - d_f\ \left.\frac{\partial p}{\partial x}\right|_f + c\ d_f\ u_f^{t-\Delta t}\ . \tag{3.33}$$

The term $\tilde{u}_f$ is determined by means of the adjacent cell centres, applying Eq. 3.32 at cell centres P and Q, following as

$$\begin{aligned} \tilde{u}_f &= \frac{1}{2}\left(\tilde{u}_P + \tilde{u}_Q\right) \\ &= \frac{1}{2}\left\{\left([1 + c\ d_P]\, u_P + d_P\ \left.\frac{\partial p}{\partial x}\right|_P - c\ d_P\ u_P^{t-\Delta t}\right)\right. \\ &+ \left.\left([1 + c\ d_Q]\, u_Q + d_Q\ \left.\frac{\partial p}{\partial x}\right|_Q - c\ d_Q\ u_Q^{t-\Delta t}\right)\right\}\ . \end{aligned} \tag{3.34}$$

Inserting Eq. 3.34 in Eq. 3.33 leads to

$$\begin{aligned} [1 + c\ d_f]\, u_f &= \frac{1 + c\ d_P}{2} u_P + \frac{1 + c\ d_Q}{2} u_Q \\ &- \left[d_f\ \left.\frac{\partial p}{\partial x}\right|_f - \frac{1}{2}\left(d_P\ \left.\frac{\partial p}{\partial x}\right|_P + d_Q\ \left.\frac{\partial p}{\partial x}\right|_Q\right)\right] \\ &+ \left[c\ d_f\ u_f^{t-\Delta t} - \frac{1}{2}\left(c\ d_P\ u_P^{t-\Delta t} + c\ d_Q\ u_Q^{t-\Delta t}\right)\right]\ . \end{aligned} \tag{3.35}$$

This can be further simplified with

$$d_f = \frac{d_P + d_Q}{2} = \frac{1}{2}\left(\frac{V_P}{a_P} + \frac{V_Q}{a_Q}\right) \tag{3.36}$$

and

$$\hat{d}_f = \frac{d_f}{1 + c\, d_f}\ . \tag{3.37}$$

After inserting Eqs. 3.36 and 3.37, Eq. 3.35 becomes

$$\begin{aligned} u_f &= \frac{u_P + u_Q}{2} - \hat{d}_f\left[\left.\frac{\partial p}{\partial x}\right|_f - \frac{1}{2}\left(\left.\frac{\partial p}{\partial x}\right|_P + \left.\frac{\partial p}{\partial x}\right|_Q\right)\right] \\ &+ c\,\hat{d}_f\left[u_f^{t-\Delta t} - \frac{u_P^{t-\Delta t} + u_Q^{t-\Delta t}}{2}\right]\ . \end{aligned} \tag{3.38}$$

As previously mentioned, the pressure term in Eq. 3.38 is responsible for damping out spurious pressure oscillations resulting from the collocated variable arrangement, because

[195, 249, 270]

$$\left.\frac{\partial p}{\partial x}\right|_f - \frac{1}{2}\left(\left.\frac{\partial p}{\partial x}\right|_P + \left.\frac{\partial p}{\partial x}\right|_Q\right) \propto \frac{\partial^3 p}{\partial x^3} \ . \tag{3.39}$$

If this relationship does not hold, even a linear pressure profile would activate the pressure dissipation term on a non-equidistant mesh, which would violate the accuracy of u_f and the filtering function of the pressure term. A detailed derivation of the relationship described in Eq. 3.39 is given in Appendix B. The entire pressure term in Eq. 3.38 becomes redundant if the pressure varies linearly in space, which corresponds to the fact that Eqs. 3.1 and 3.2 yield the same result if the pressure field varies linearly and pressure and velocity are coupled by default.

Including all Cartesian coordinate axes, the advecting velocity u_f^n at face f for single-phase flows on equidistant Cartesian meshes follows by multiplying Eq. 3.38 with normal vector $\boldsymbol{n}_f$ as

$$\begin{aligned} u_f^n &= \boldsymbol{u}_f\boldsymbol{n}_f - \hat{d}_f\left[\left.\nabla p\right|_f \boldsymbol{n}_f - \frac{1}{2}\left(\left.\nabla p\right|_P + \left.\nabla p\right|_Q\right)\boldsymbol{n}_f\right] \\ &+ c\,\hat{d}_f\left[u_f^{n,t-\Delta t} - \boldsymbol{u}_f^{t-\Delta t}\boldsymbol{n}_f\right] \ , \end{aligned} \tag{3.40}$$

with

$$\boldsymbol{u}_f = \frac{\boldsymbol{u}_P + \boldsymbol{u}_Q}{2} \tag{3.41}$$

and analogously for $\boldsymbol{u}_f^{t-\Delta t}$. The combined implicit coefficient a of the convective and viscous terms of the discretised momentum equation, used in Eq. 3.36 to calculate coefficient d_f, is taken to be the average coefficient of the three momentum equations, *i.e.* $a = \text{avg}\{a_x, a_y, a_z\}$ with subscripts x, y and z denoting the three coordinate axes. The product of pressure gradient $\left.\nabla p\right|_f$ and normal vector $\boldsymbol{n}_f$ at face f is calculated as

$$\left.\nabla p\right|_f \boldsymbol{n}_f = \frac{p_Q - p_P}{\Delta s} \ , \tag{3.42}$$

with Δs being the distance between the adjacent cells P and Q, as depicted in Figure 2.2a. The velocity term $\boldsymbol{u}_f\boldsymbol{n}_f$ and the pressure gradient term $\left.\nabla p\right|_f \boldsymbol{n}_f$ at the face are implemented implicitly, providing the required coupling between velocity and pressure in order to close the fully-coupled equation system. The pressure gradient terms at cell centres P and Q are implemented explicitly using the pressure gradient of the previous iteration.

3.3.2. Modifications for Arbitrary Meshes

In the previous section the advecting velocity is derived for an equidistant Cartesian mesh using the momentum interpolation method. Corrections to Eq. 3.40 are required if arbitrary meshes are deployed, for instance non-orthogonal/non-equidistant hexahedral meshes or tetrahedral meshes. Figure 2.2b on page 48 illustrates an example of a face with its two adjacent cells as part of an unstructured mesh and the relevant interpolation entities.

To account for mesh skewness and varying distances from cell to face centres on arbitrary meshes, the face velocity is calculated by linear interpolation with inverse distance weighting from adjacent cell centred velocities, replacing the symmetrical linear interpolation of the face velocity on Cartesian meshes given in Eq. 3.41. The face velocity is, therefore, given at interpolation point f' as

$$\boldsymbol{u}_{f'} = (1-\delta)\, \boldsymbol{u}_P + \delta\, \boldsymbol{u}_Q \ , \tag{3.43}$$

where δ is the inverse distance interpolation coefficient. Additionally, the interpolated velocity $\boldsymbol{u}_{f'}$ is corrected to face centre f by the interpolated velocity gradient

$$\nabla \boldsymbol{u}|_{f'} = (1-\delta)\, \nabla \boldsymbol{u}|_P + \delta\, \nabla \boldsymbol{u}|_Q \ . \tag{3.44}$$

Thus, the velocity at face centre f becomes

$$\boldsymbol{u}_f = (1-\delta)\, \boldsymbol{u}_P + \delta\, \boldsymbol{u}_Q + \Big[(1-\delta)\, \nabla \boldsymbol{u}|_P + \delta\, \nabla \boldsymbol{u}|_Q \Big]\, \boldsymbol{r}_f \ , \tag{3.45}$$

where $\boldsymbol{r}_f$ represents the vector from interpolation point f' to face centre f.

Considering mesh with non-orthogonal faces, the pressure term requires a correction similar to the viscosity term of the momentum equation. The non-orthogonal correction presented below follows the approach of Zwart [277]. The first pressure term in Eq. 3.40, $\nabla p|_f\, \boldsymbol{n}_f$, is decomposed as

$$\nabla p|_f\, \boldsymbol{n}_f = \nabla p|_f\, (\alpha_f\, \boldsymbol{s}_f) + \overline{\nabla p}\big|_f\, (\boldsymbol{n}_f - \alpha_f\, \boldsymbol{s}_f) \ , \tag{3.46}$$

where $\alpha_f = (\boldsymbol{n}_f \cdot \boldsymbol{s}_f)^{-1}$ is the scaling factor of the non-orthogonal correction as presented in Section 2.2.4, based on the face normal vector $\boldsymbol{n}_f$ and the normalised vector $\boldsymbol{s}_f$ connecting the two adjacent cells. The pressure gradients on the right-hand side of Eq. 3.46 are discretised as

$$\nabla p|_f\, (\alpha_f\ \boldsymbol{s}_f) = \alpha_f\ \frac{p_Q - p_P}{\Delta s} \tag{3.47}$$

$$\overline{\nabla p}\big|_f = \frac{1}{2} \Big(\nabla p|_P + \nabla p|_Q \Big) \ . \tag{3.48}$$

Inserting the non-orthogonal correction in Eq. 3.40 leads to

$$\begin{aligned} u_f^n &= \boldsymbol{u}_f \boldsymbol{n}_f - \hat{d}_f \left[\alpha_f\ \frac{p_Q - p_P}{\Delta s} + \frac{1}{2} \Big(\nabla p|_P + \nabla p|_Q \Big) (\boldsymbol{n}_f - \alpha_f\ \boldsymbol{s}_f) \right. \\ &- \left. \frac{1}{2} \Big(\nabla p|_P + \nabla p|_Q \Big)\, \boldsymbol{n}_f \right] + c\ \hat{d}_f \left[u_f^{n,t-\Delta t} - \boldsymbol{u}_f^{t-\Delta t} \boldsymbol{n}_f \right] \ . \end{aligned} \tag{3.49}$$

After removing the obsolete pressure terms from Eq. 3.49, the advecting velocity at face

f on an arbitrary mesh becomes

$$\begin{aligned} u_f^n &= \boldsymbol{u}_f \boldsymbol{n}_f - \alpha_f \, \hat{d}_f \left[\frac{p_Q - p_P}{\Delta s} - \frac{1}{2} \left(\nabla p|_P + \nabla p|_Q \right) \boldsymbol{s}_f \right] \\ &+ \; c \, \hat{d}_f \left[u_f^{n,t-\Delta t} - \boldsymbol{u}_f^{t-\Delta t} \boldsymbol{n}_f \right] \; , \end{aligned} \tag{3.50}$$

where $\boldsymbol{u}_f$ and $\boldsymbol{u}_f^{t-\Delta t}$ are interpolated as described in Eq. 3.45.

Because the orthogonal contribution of the pressure term in Eq. 3.50 is interpolated symmetrically as defined in Eq. 3.47, it is essential that the interpolation coefficient of the non-orthogonal contribution defined in Eq. 3.48 is 1/2 even on arbitrary meshes to assure that the relationship defined in Eq. 3.39 holds. Similarly, the interpolation coefficient used to evaluate coefficient d_f, given in Eq. 3.36, must also be 1/2 on arbitrary meshes to satisfy the relationship defined in Eq. 3.39.

On an equidistant orthogonal mesh, such as the equidistant Cartesian mesh used in the previous section, the velocity interpolation presented in Eq. 3.45 reverts back to the interpolation defined in Eq. 3.41 and vector $\boldsymbol{s}_f = \boldsymbol{n}_f$. Thus, using Eq. 3.50 for the calculation of the advecting velocity u_f^n, the implementation for structured and unstructured meshes is identical and no changes based on the mesh type are required.

3.3.3. Modifications for Gravity

The volumetric gravity force $\boldsymbol{f}_g$, as well as any other body force, has to be taken into account for the calculation of the advecting velocity. Neglecting body forces can lead to substantial imbalances, particularly in two-phase flows [37, 154]. Assuming gravity is acting on a stationary single-phase flow without additional external forces, the momentum equation reduces to

$$\nabla p = \boldsymbol{S}_g \; . \tag{3.51}$$

In order to ensure this relationship is valid at the discrete level, the gravity source term $\boldsymbol{S}_g$ has to be included in the calculation of the advecting velocity using a similar interpolation as for the pressure gradient. Decomposing the pressure term in Eq. 3.38 into a flow contribution (denoted with subscript u) and a contribution resulting from the gravity force (denoted with subscript g), defined as

$$\nabla p|_f - \overline{\nabla p}\big|_f = \left\{ \nabla p|_f - \overline{\nabla p}\big|_f \right\}_u + \left\{ \nabla p|_f - \overline{\nabla p}\big|_f \right\}_g \; , \tag{3.52}$$

the gravity force must essentially be implemented so that

$$\left\{ \nabla p|_f - \overline{\nabla p}\big|_f \right\}_g - \left[\boldsymbol{S}_{g,f} - \overline{\boldsymbol{S}}_{g,f} \right] = 0 \tag{3.53}$$

to assure a balance between pressure gradient and gravity force. Thus, the advecting velocity including the gravity source term follows as

$$\begin{aligned} u_f^n &= \boldsymbol{u}_f \boldsymbol{n}_f - \alpha_f \, \hat{d}_f \left[\frac{p_Q - p_P}{\Delta s} - \frac{1}{2} \left(\nabla p|_P + \nabla p|_Q \right) \boldsymbol{s}_f \right] \\ &+ \alpha_f \, \hat{d}_f \left[\rho (\boldsymbol{g} \cdot \boldsymbol{s}_f) - \frac{1}{2} \left(\boldsymbol{S}_{g,P} + \boldsymbol{S}_{g,Q} \right) \boldsymbol{s}_f \right] \\ &+ c \, \hat{d}_f \left[u_f^{n,t-\Delta t} - \boldsymbol{u}_f^{t-\Delta t} \boldsymbol{n}_f \right] \ . \end{aligned} \tag{3.54}$$

The advecting velocity resulting from Eq. 3.54 ensures an exact balance between pressure gradient and gravity force. This principle works theoretically also for any other body force.

3.3.4. Modifications for Surface Tension

The simulation of two-phase flows requires further modification of the momentum interpolation method applied to determine the advecting velocity at cell faces, since large density jumps at the interface may occur and because surface tension significantly affects the pressure gradient.

The effect of a changing density has to be taken into account when simulating two-phase flows, specifically if large density jumps at interfaces occur. The density ρ_f at mesh face f is calculated by means of a harmonic average of the densities at adjacent cell centres, as defined in Eq. 3.22. The coefficient c, defined previously in Eq. 3.29, containing ρ_f at face f then becomes

$$c_f = \frac{\rho_f}{\Delta t} \tag{3.55}$$

and, therefore, Eq. 3.37 becomes

$$\hat{d}_f = \frac{d_f}{1 + c_f d_f} \ . \tag{3.56}$$

In order to improve the stability of the numerical framework in cases of large density ratios between the two fluids, it is proposed to weight the second pressure term of Eq. 3.50 by the ratio of face density and cell density. The motivation behind this weighting is to align the pressure term of the advecting velocity with the pressure term of the momentum equation, as the pressure term in the momentum equation is effectively weighted by the density as well, and to dampen pressure oscillations potentially arising from large density jumps at the interface. The proposed advecting velocity then follows as

$$\begin{aligned} u_f^n &= \boldsymbol{u}_f \boldsymbol{n}_f - \alpha_f \, \hat{d}_f \left[\frac{p_Q - p_P}{\Delta s} - \frac{\rho_f}{2} \left(\frac{\nabla p|_P}{\rho_P} + \frac{\nabla p|_Q}{\rho_Q} \right) \boldsymbol{s}_f \right] \\ &+ c_f \, \hat{d}_f \left[u_f^{n,t-\Delta t} - \boldsymbol{u}_f^{t-\Delta t} \boldsymbol{n}_f \right] \ . \end{aligned} \tag{3.57}$$

The potential of this approach regarding the stability of the numerical framework is demonstrated in Section 3.8.4.1.

Because of its significant impact on the pressure field, the surface force has to be included in the calculation of the advecting velocity for two-phase flows, similar to the inclusion of the gravity force discussed in Section 3.3.3. Considering a stationary two-phase flow without gravity or any other external body forces, the momentum equation reduces to

$$\nabla p = \boldsymbol{f}_s \ . \tag{3.58}$$

Thus, in a balanced-force framework the pressure gradient ∇p and the volumetric surface force $\boldsymbol{f}_s$ must match each other on a discrete level. Similar to the relationship defined in Eq. 3.53 for gravity, the condition

$$\left\{ \nabla p|_f - \overline{\nabla p}|_f \right\}_\sigma - \sigma \left[\kappa_f \, \nabla\gamma|_f - \overline{\kappa \nabla\gamma}|_f \right] = 0 \ , \tag{3.59}$$

where subscript σ denotes the pressure contribution resulting from surface tension, must be satisfied to provide a balance between pressure gradient and surface force. With surface force $\boldsymbol{f}_s$ defined according to Eq. 2.48 and by applying the same discretisation stencil for colour function and pressure, the advecting velocity at face f is given as

$$\begin{aligned} u_f^n \;&=\; \boldsymbol{u}_f \boldsymbol{n}_f - \alpha_f \, \hat{d}_f \left[\frac{p_Q - p_P}{\Delta s} - \frac{\rho_f}{2} \left(\frac{\nabla p|_P}{\rho_P} + \frac{\nabla p|_Q}{\rho_Q} \right) \boldsymbol{s}_f \right] \\ &+\; \alpha_f \, \hat{d}_f \, \sigma \left[\kappa_f \frac{\gamma_Q - \gamma_P}{\Delta s} - \frac{\rho_f}{2} \left(\frac{\kappa_P \nabla\gamma|_P}{\rho_P} + \frac{\kappa_Q \nabla\gamma|_Q}{\rho_Q} \right) \boldsymbol{s}_f \right] \\ &+\; c_f \, \hat{d}_f \left[u_f^{n,t-\Delta t} - \boldsymbol{u}_f^{t-\Delta t} \boldsymbol{n}_f \right] \ , \end{aligned} \tag{3.60}$$

with $\kappa_f = (\kappa_P + \kappa_Q)/2$. Including the gravity term as described in Section 3.3.3, weighted by the local densities as introduced in Eq. 3.57, follows as

$$\begin{aligned} u_f^n \;&=\; \boldsymbol{u}_f \boldsymbol{n}_f - \alpha_f \, \hat{d}_f \left[\frac{p_Q - p_P}{\Delta s} - \frac{\rho_f}{2} \left(\frac{\nabla p|_P}{\rho_P} + \frac{\nabla p|_Q}{\rho_Q} \right) \boldsymbol{s}_f \right] \\ &+\; \alpha_f \, \hat{d}_f \, \sigma \left[\kappa_f \frac{\gamma_Q - \gamma_P}{\Delta s} - \frac{\rho_f}{2} \left(\frac{\kappa_P \nabla\gamma|_P}{\rho_P} + \frac{\kappa_Q \nabla\gamma|_Q}{\rho_Q} \right) \boldsymbol{s}_f \right] \\ &+\; \alpha_f \, \hat{d}_f \left[\rho_f (\boldsymbol{g} \cdot \boldsymbol{s}_f) - \frac{\rho_f}{2} \left(\frac{\boldsymbol{S}_{g,P}}{\rho_P} + \frac{\boldsymbol{S}_{g,Q}}{\rho_Q} \right) \boldsymbol{s}_f \right] \\ &+\; c_f \, \hat{d}_f \left[u_f^{n,t-\Delta t} - \boldsymbol{u}_f^{t-\Delta t} \boldsymbol{n}_f \right] \ . \end{aligned} \tag{3.61}$$

Eq. 3.61 provides an equation to compute the advecting velocity at face centres for two-phase flows, which prevents pressure-velocity decoupling and provides an exact balance between pressure gradient, surface force and gravity force.

3.4. Compressive Volume-of-Fluid Method

In the compressive VOF approach the colour function transport equation (Eq. 2.43) is discretised using algebraic discretisation schemes and the colour function is transported in a time-marching fashion. In the following sections, the discretisation of the transient term and the spatial advection term of the colour function transport equation (Eq. 2.43) is presented and discussed. The presented methodology is applicable to arbitrary meshes and inherently conserves the colour function [238]. The numerical schemes presented in this section represent only one specific choice, deemed to be best suited for the discretisation of the colour function transport on unstructured meshes. Other schemes may be used in a similar way without changing other parts of the numerical framework.

3.4.1. Temporal Discretisation

The transient term of Eq. 2.43 is discretised using a second-order temporal discretisation scheme, such as the Crank-Nicolson scheme (Eq. 2.39) or the Second-Order Backward Euler scheme (Eq. 2.41) [39, 238], since first-order schemes, such as the First-Order Backward Euler scheme, are too diffusive to maintain the sharpness of the interface. Studies by Darwish [37], Jasak [106] and Ubbink [238] comprehensively demonstrate that the First-Order Backward Euler scheme as well as the First-Order Forward Euler scheme distort the shape of a circular interface advected at a constant oblique velocity on an equidistant Cartesian mesh. The Crank-Nicolson scheme, on the other hand, is able to preserve the shape of the circular interface of the same test case.

Moukalled and Darwish [161] proposed a class of temporal discretisation schemes, which switch between a compressive and a high-resolution scheme. The blending function, determining the transition between compressive and high-resolution schemes, is based on the angle between the interface normal vector and the velocity vector. In their paper, Moukalled and Darwish used the Second-Order Backward Euler scheme and a compressive Euler scheme. Advecting variously shaped interfaces at medium and high Courant numbers, *i.e.* $Co = 0.5$ and $Co = 1.0$, Moukalled and Darwish [161] reported better results using the new temporal discretisation method than deploying other commonly used schemes, such as the Crank-Nicolson scheme. However, at small Courant number ($Co = 0.25$) the Crank-Nicolson scheme yields similar or even better results than the new discretisation approach. Given the restrictive Courant number limit imposed by the spatial discretisation, explained in Section 3.4.3, there is no notable advantage gained from adopting the transient discretisation of Moukalled and Darwish [161] compared to the computationally more efficient and easier to implement Crank-Nicolson scheme.

In the presented numerical framework, the transient term of Eq. 2.43 is discretised using the Crank-Nicolson scheme. The discretised transport equation of the colour function is, therefore, given as

$$(\gamma_P^t - \gamma_P^{t-\Delta t})V_P = -\sum_f \frac{1}{2}\left(\gamma_f^t F_f^t + \gamma_f^{t-\Delta t} F_f^{t-\Delta t}\right)\Delta t \ , \tag{3.62}$$

with flux F through face f being

$$F_f = (\boldsymbol{u}_f \cdot \boldsymbol{n}_f)A_f \ . \tag{3.63}$$

The discretisation presented in Eq. 3.62 requires the fluxes through the face of two time instants, thus, increasing memory requirements. Moreover, as further discussed in Section

3.7.2, the fluxes of the new time instant are not yet known. Ubbink [238] proposed that if the time-step is small enough, as it is due to the Courant number limitations outlined in Section 3.4.3, the right-hand side of Eq. 3.62 can be approximated as

$$-\sum_f \frac{1}{2}\left(\gamma_f^t F_f^t + \gamma_f^{t-\Delta t} F_f^{t-\Delta t}\right)\Delta t \approx -\sum_f \gamma_f^* F_f \Delta t \ , \tag{3.64}$$

with the average colour function value at the face being

$$\gamma_f^* = \frac{\gamma_f^t + \gamma_f^{t-\Delta t}}{2} \ . \tag{3.65}$$

Hence, Eq. 3.62 simplifies to

$$(\gamma_P^t - \gamma_P^{t-\Delta t})V_P = -\sum_f \gamma_f^* F_f \Delta t \ . \tag{3.66}$$

Analogous to the continuity constraint defined in Eq. 3.25, the flux F_f in Eq. 3.66 is defined as

$$F_f = u_f^n A_f \ . \tag{3.67}$$

Thus, flux F_f used to advect the colour function satisfies continuity and the advection of the colour function is defined consistently with the flow advection.

3.4.2. Spatial Discretisation

As already mentioned in Section 2.3.2, low-order advection schemes are not suitable for the discretisation of the VOF colour function transport, as they lead to significant smearing of the interface, and the application of high-order discretisation schemes results in numerical oscillations and wrinkling of the interface. A number of spatial discretisation schemes specifically designed for compressive VOF methods have been published, most notably the CICSAM scheme [239], the Inter-Gamma scheme [108], the HiRAC scheme [96], the HRIC scheme [163] and the STACS scheme [39].

The CICSAM scheme of Ubbink and Issa [239] is implemented as part of the presented numerical framework to determine the value of colour function γ_f at mesh face f in Eq. 3.65. The CICSAM scheme is founded on the *Normalised Variable Diagram* (NVD) of Leonard [126]. By defining an acceptor cell A, a donor cell D and an upwind point U for the mesh face f under consideration, as illustrated in Figure 3.1, the normalised colour function value at face f is defined as

$$\tilde{\gamma}_f = \frac{\gamma_f - \gamma_U}{\gamma_A - \gamma_U} \tag{3.68}$$

and the normalised value of the donor cell is

$$\tilde{\gamma}_D = \frac{\gamma_D - \gamma_U}{\gamma_A - \gamma_U} \ . \tag{3.69}$$

On structured meshes the upwind value is readily available whereas on unstructured meshes this is generally not the case. Jasak [106] proposed an extrapolation of the colour function to overcome this problem, schematically illustrated in Figure 3.1, where the approximated upwind value is calculated as

$$\gamma_U = \gamma_A + 2\,(\boldsymbol{x}_D - \boldsymbol{x}_A) \cdot \nabla\gamma|_A \ , \tag{3.70}$$

with $\boldsymbol{x}_D$ and $\boldsymbol{x}_A$ representing the position vector of donor cell D and acceptor cell A, respectively.

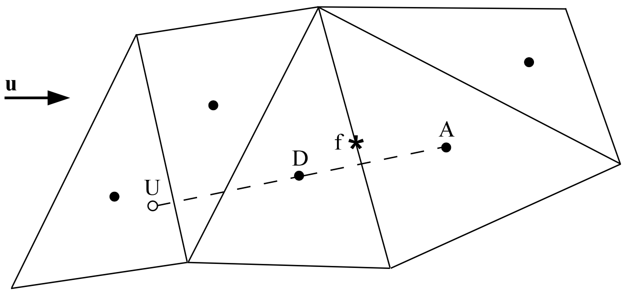


Figure 3.1.: Example of the extrapolation of upwind node U for the CICSAM scheme on unstructured meshes with respect to the face under consideration f, donor cell D and acceptor cell A.

The CICSAM scheme is based on the Hyper-C (HC) scheme, which follows the upper bound of the convective boundedness criteria [75], and the ULTIMATE QUICKEST (UQ) scheme of Leonard [126]. Using the normalised values defined in Eqs. 3.68 and 3.69, the HC scheme is given as

$$\tilde{\gamma}_f^{HC} = \begin{cases} \min\left\{1, \frac{\tilde{\gamma}_D}{Co_D}\right\} & \text{when } 0 \leq \tilde{\gamma}_D \leq 1 \\ \tilde{\gamma}_D & \text{when } \tilde{\gamma}_D > 0 \text{ and } \tilde{\gamma}_D < 1 \ , \end{cases} \tag{3.71}$$

where Co_D is the Courant number with respect to the fluxes leaving the donor cell,

$$Co_D = \sum_f \max\left\{\frac{u_f^n\,\Delta t}{V_D}, 0\right\} \ . \tag{3.72}$$

The UQ scheme is defined as

$$\tilde{\gamma}_f^{UQ} = \begin{cases} \min\left\{\frac{8\,Co_D\,\tilde{\gamma}_D + (1-Co_D)(6\,\tilde{\gamma}_D+3)}{8}, \tilde{\gamma}_f^{HC}\right\} & \text{when } 0 \leq \tilde{\gamma}_D \leq 1 \\ \tilde{\gamma}_D & \text{when } \tilde{\gamma}_D > 0 \text{ and } \tilde{\gamma}_D < 1 \ . \end{cases} \tag{3.73}$$

Having defined both discretisation schemes, a blending function ψ_f is required, so that

$$\tilde{\gamma}_f = \psi_f\ \tilde{\gamma}_f^{HC} + (1-\psi_f)\ \tilde{\gamma}_f^{UQ} \ . \tag{3.74}$$

The blending function proposed by Ubbink and Issa [239] is based on the angle θ_f between the interface normal vector $\boldsymbol{m}_f$ and the normalised vector $\boldsymbol{s}_f$ connecting the donor cell and acceptor cell. The blending function ψ_f is defined as

$$\psi_f = \min\left\{k_\psi \frac{\cos(2\theta_f)+1}{2}, 1\right\} \ , \tag{3.75}$$

with angle θ_f being

$$\theta_f = \arccos\left(|\boldsymbol{m}_f \cdot \boldsymbol{s}_f|\right) \ . \tag{3.76}$$

The constant k_ψ in Eq. 3.75 controls the dominance of the HC scheme (the compressive scheme), where an increasing value of k_ψ corresponds to a stronger dominance of the HC scheme. If not explicitly stated otherwise, $k_\psi = 1$ following the recommendation of Ubbink and Issa [239]. After algebraic manipulation, the colour function value at face f follows as [238, 239]

$$\gamma_f = (1-\beta_f)\ \gamma_D + \beta_f\ \gamma_A \ , \tag{3.77}$$

where the weighting factor β_f is calculated from the normalised colour function values as

$$\beta_f = \frac{\tilde{\gamma}_f - \tilde{\gamma}_D}{1-\tilde{\gamma}_D} \ . \tag{3.78}$$

Therefore, including the relationship described in Eq. 3.77, the average colour function value at face f, previously defined in Eq. 3.65, becomes

$$\gamma_f^* = (1-\beta_f)\frac{\gamma_D^t + \gamma_D^{t-\Delta t}}{2} + \beta_f \frac{\gamma_A^t + \gamma_A^{t-\Delta t}}{2} \ . \tag{3.79}$$

As pointed out by Ubbink [238], Eq. 3.79 does not guarantee a bounded solution, in particular on unstructured meshes. In order to obtain a bounded solution, a corrector step is implemented following Ubbink [238]. Because of the implicit implementation, the predicted value γ_f^* cannot be corrected directly but instead the weighting factor β_f should be corrected. If a negative colour function value in the donor cell D of face f is obtained, *i.e.* $\gamma_D < 0$, the weighting factor β_f is corrected by

$$\beta_f' = \begin{cases} \min\left\{\frac{E^-(2+Co_D - 2\,Co_D\,\beta_f)}{2\,Co_D(\Delta\gamma^* - E^-)}, \beta_f'\right\} & \text{when } \Delta\gamma^* > E^- \\ 0 & \text{when } \Delta\gamma^* \leq E^- \ , \end{cases} \tag{3.80}$$

with

$$E^- = \max\left\{-\gamma_D^t, 0\right\} \tag{3.81}$$

being the magnitude of the unbounded value and

$$\Delta\gamma^* = \frac{\gamma_A^t + \gamma_A^{t-\Delta t}}{2} - \frac{\gamma_D^t + \gamma_D^{t-\Delta t}}{2} \ . \tag{3.82}$$

Similarly, for $\gamma_D > 1$ the corrector follows as

$$\beta_f' = \begin{cases} \min\left\{\frac{E^+(2+Co_D - 2\,Co_D\,\beta_f)}{2\,Co_D(-\Delta\gamma^* - E^+)}, \beta_f'\right\} & \text{when } \Delta\gamma^* < -E^+ \\ 0 & \text{when } \Delta\gamma^* \geq -E^+ \ , \end{cases} \tag{3.83}$$

with

$$E^{+} = \max\left\{\gamma_D^t - 1, 0\right\} \ . \tag{3.84}$$

The corrected weighting factor is then calculated as

$$\beta_f^{**} = \beta_f - \beta_f' \tag{3.85}$$

and the corrected face value is given as

$$\gamma_f^{**} = (1 - \beta_f^{**}) \frac{\gamma_D^t + \gamma_D^{t-\Delta t}}{2} + \beta_f^{**} \frac{\gamma_A^t + \gamma_A^{t-\Delta t}}{2} \ . \tag{3.86}$$

For a detailed derivation of the corrector step the interested reader may consult the thesis of Ubbink [238] or the article of Ubbink and Issa [239].

Because the predictor-corrector approach outlined above can only determine the colour function advection within the predefined solver tolerance, the colour function value at any given time instant may not fulfil the boundedness criteria

$$0 \leq \gamma \leq 1 \ . \tag{3.87}$$

Thus, any colour function value lower than zero or larger than unity is explicitly set to the respective bound. The inherent conservation error of the colour function advection for every time-step is, therefore, not larger than the order of magnitude of the solver tolerance.

3.4.3. Advection Time Step

The time-step applied in Eq. 3.66 to transport the VOF colour function γ has a crucial influence on the accuracy of the simulation results. The advection of a circular interface performed by Ubbink [238, chap. 5] indicates significantly less numerical diffusion for decreasing Courant numbers, with reasonable results for $Co = |\boldsymbol{u}| \, \Delta t / \Delta x \leq 0.3$. Darwish and Moukalled [39] attributed the strong Courant number dependence to the combination of a spatial advection scheme designed for explicit calculations [126] and an implicit transient discretisation. Darwish and Moukalled argue that according to studies of Jasak [106], the numerical diffusion of transient and spatial scheme, using the Explicit Euler scheme and Upwind Differencing, cancel each other out when the Courant number approaches unity. On the other hand, using the Implicit Euler scheme in conjunction with Upwind Differencing, the numerical diffusion caused by both schemes accumulates with increasing Courant number. The case studies of Darwish and Moukalled [39] confirm the trend of significantly improved results for decreasing Courant numbers as presented by Ubbink [238], suggesting viable results for $Co \leq 0.25$. Studies performed by Gopala and van Wachem [81] suggest a Courant number limit of $Co \leq 0.01$ in order to maintain a sharp interface. Following the findings of the mentioned studies, an adaptive time-step assuring $Co_{max} \leq 0.01$ is used to advect the VOF colour function for the simulations presented in this thesis.

3.5. Implementation of Boundary Conditions

The boundary condition at each mesh face b coinciding with a domain boundary and for each variable is either extrapolated or set to a fixed value. Calculating the boundary value by extrapolation, the boundary face value ϕ_b follows as

$$\phi_b = \phi_P + \nabla\phi|_P \cdot \boldsymbol{r}_b \ , \tag{3.88}$$

with $\boldsymbol{r}_b$ being the vector from cell centre P to face centre b. Using a predefined boundary value, the boundary condition is implemented in the form

$$A\,\phi_b + B\,\left.\frac{\partial\phi}{\partial n}\right|_b = C \ , \tag{3.89}$$

where A, B and C represent coefficients which define the boundary type and boundary value. For instance, setting ϕ at a given boundary to the fixed value 10 would be achieved with the coefficients $A = 1$, $B = 0$ and $C = 10$. Similarly, a fixed gradient normal to the boundary face of 500 is defined as $A = 0$, $B = 1$ and $C = 500$. In order to calculate the boundary face value ϕ_b from Eq. 3.89, the adjacent cell centre is mirrored at the boundary face, as illustrated in Figure 3.2.

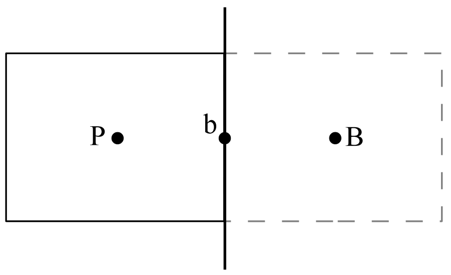


Figure 3.2.: Illustration of boundary mirroring across boundary-face b.

Therefore, the cell centre of the adjacent cell and its mirrored counterpart are equidistant in relation to the boundary face. The boundary face value at face b then follows as

$$\phi_b = \frac{\phi_P + \phi_B}{2} \tag{3.90}$$

and the gradient normal to face b becomes

$$\left.\frac{\partial\phi}{\partial n}\right|_b = \frac{\phi_B - \phi_P}{2\,|\boldsymbol{r}_b|} \ . \tag{3.91}$$

Inserting Eqs. 3.90 and 3.91 in Eq. 3.89, the value at the mirrored cell centre ϕ_B is

$$\begin{aligned} A\,\frac{\phi_P+\phi_B}{2}+B\,\frac{\phi_B-\phi_P}{2\,|\boldsymbol{r}_b|} &= C \\ \phi_B\left(\frac{A}{2}+\frac{B}{2\,|\boldsymbol{r}_b|}\right) &= \phi_P\left(-\frac{A}{2}+\frac{B}{2\,|\boldsymbol{r}_b|}\right)+C \\ \phi_B &= \phi_P\,\frac{-\frac{A}{2}+\frac{B}{2\,|\boldsymbol{r}_b|}}{\frac{A}{2}+\frac{B}{2\,|\boldsymbol{r}_b|}}+\frac{C}{\frac{A}{2}+\frac{B}{2\,|\boldsymbol{r}_b|}}\;. \end{aligned} \tag{3.92}$$

Irrespectively whether the boundary value is extrapolated, as described by Eq. 3.88, or holds a fixed predefined value, calculated by Eq. 3.92, the first term on the right-hand side of either equation (Eq. 3.88 or Eq. 3.92) is implemented implicitly and the second term on the right-hand side is implemented explicitly.

3.6. Gradient Evaluation

Two common methods to evaluate gradients in a finite volume framework are available: least-squares gradient construction methods and the Gauss theorem. *Least-squares methods* construct the gradient based on the values at an arbitrary finite number of neighbour points using a least-squares fit. Least-squares methods are independent of the underlying mesh in the sense that the mesh errors discussed in Section 2.2.1 do not affect the result or stability of the gradient evaluation. However, the accuracy of least-squares methods depends strongly on the computational stencil [89] and the applied weighting function [153]. The *Gauss theorem* is a cornerstone of finite volume methods, as described in Section 2.2, and gradients can be readily computed with the available data. The Gauss theorem has no directional dependence, is conservative in nature and the computational stencil includes only direct neighbour cells. Moreover, the Gauss theorem is computationally cheaper than least-squares methods and its implicit implementation is straightforward. However, the result and stability of the gradient evaluation using the Gauss theorem in a finite volume framework is significantly affected by the mesh errors discussed in Section 2.2.1. Because the numerical discretisation presented in Sections 3.2 - 3.5 is based on the Gauss theorem, the Gauss theorem is also applied to explicitly evaluate gradients. This is essential to assure the discrete balance between the pressure gradient and the acting body forces.

The gradient at cell P is defined using the Gauss theorem as

$$\nabla\phi|_P = \frac{1}{V_P}\sum_f \phi_f\,\boldsymbol{n}_f\,A_f\;, \tag{3.93}$$

where V_P is the volume of cell P, subscript f denotes the faces bounding cell P, $\boldsymbol{n}_f$ is the outward-pointing normal vector of face f and A_f is the area of face f. As previously discussed in Section 2.2.2, the value interpolated at face centres from the two adjacent cell centers must be corrected if the mesh is skewed. The face value then becomes

$$\phi_f = (1-\delta)\,\phi_P+\delta\,\phi_Q+\Big[(1-\delta)\;\nabla\phi|_P+\delta\;\nabla\phi|_Q\Big]\,\boldsymbol{r}_f\;, \tag{3.94}$$

where P and Q are the cells adjacent to face f, δ is the inverse distance interpolation coefficient as defined in Eq. 2.19 and $\boldsymbol{r}_f$ is the vector connecting the interpolation point f' and the actual face centre f, as illustrated in Figure 2.2b. Using Eq. 3.94 to calculate the face values ϕ_f for Eq. 3.93 requires an iterative algorithm to determine the cell-centred gradients, because Eq. 3.94 explicitly includes thus cell-centred gradients. Eq. 3.93 is solved iteratively until the predefined maximum number of iterations is reached or the maximum gradient residual in the domain is smaller than a predefined threshold. The threshold for the gradient residual is defined as

$$\max\{|\ \nabla\phi|_P^i - \nabla\phi|_P^{i-1}\ |\} < \epsilon_{\mathrm{rel}}\ , \tag{3.95}$$

with ϵ_{rel} representing the relative tolerance of the flow solver and superscripts i and $i-1$ denoting the gradients of the current and the previous iteration, respectively.

The convergence of the gradient evaluation may be adversely affected at boundaries where the values are extrapolated, in particular the pressure gradient evaluation when large body forces are acting on the fluid. In order to improve the convergence of the gradient evaluation, a different implementation than the straightforward method described in the previous paragraph is proposed for cells bounded by at least one face to which the relevant values are extrapolated. For face e, representing an extrapolation-boundary, the value at its face centre is

$$\phi_e = \phi_P + \nabla\phi|_P \cdot \boldsymbol{r}_e\ , \tag{3.96}$$

where $\boldsymbol{r}_e$ is the vector from cell centre P to face centre e. Following Eq. 3.93, the gradient at cell centre P is given as

$$\nabla\phi|_P = \frac{1}{V_P}\sum_f \phi_f\,\boldsymbol{n}_f\,A_f = \frac{1}{V_P}\left(\phi_e\,\boldsymbol{n}_e\,A_e + \sum_{f\neq e}\phi_f\,\boldsymbol{n}_f\,A_f\right)\ . \tag{3.97}$$

Inserting Eq. 3.97 in Eq. 3.96 follows as

$$\phi_e = \phi_P + \phi_e\frac{A_e\,(\boldsymbol{n}_e\cdot\boldsymbol{r}_e)}{V_P} + \frac{1}{V_P}\sum_{f\neq e}\phi_f\,A_f\,(\boldsymbol{n}_f\cdot\boldsymbol{r}_e)\ . \tag{3.98}$$

By rearranging Eq. 3.98, the extrapolated value at face center e becomes

$$\phi_e\left(1-\frac{A_e\,(\boldsymbol{n}_e\cdot\boldsymbol{r}_e)}{V_P}\right) = \phi_P + \frac{1}{V_P}\sum_{f\neq e}\phi_f\,A_f\,(\boldsymbol{n}_f\cdot\boldsymbol{r}_e) \tag{3.99}$$

$$\phi_e = \frac{\phi_P + \frac{1}{V_P}\sum_{f\neq e}\phi_f\,A_f\,(\boldsymbol{n}_f\cdot\boldsymbol{r}_e)}{1-\frac{A_e\,(\boldsymbol{n}_e\cdot\boldsymbol{r}_e)}{V_P}}\ . \tag{3.100}$$

Because mesh skewness is corrected in an iterative manner using cell-centred gradients, ϕ_f in Eq. 3.100 might be an extrapolated value itself, should cell P be bounded by more than one face holding an extrapolation boundary condition. A preceding step is proposed before extrapolating face values as described in Eq. 3.100, in order to circumvent significant inaccuracies caused by other extrapolated values when extrapolating a face value. This preceding step is proposed to avoid the divergence of the gradient evaluation. All extrapolated face values are first estimated based on the average of gradients at neighbouring cells, which is defined for all face centres e coinciding with an extrapolation boundary as

$$\phi_e = \phi_P + \frac{\boldsymbol{r}_e}{N_Q} \sum_Q \nabla\phi|_Q \ , \tag{3.101}$$

where subscript Q denotes the neighbours of cell P and N_Q is the number of neighbours of cell P. This provides a smoother transition between iterations and significantly improves the convergence and stability of the gradient evaluation.

3.7. Solution Procedure

The numerical discretisation presented in Sections 3.2 - 3.6 leads to two algebraic equation systems, one that represents the fluid flow and one that describes the advection of the VOF colour function, which are solved separately. The two equation systems are explicitly coupled by the face fluxes F_f and the surface force $\boldsymbol{F}_s$. For each partial differential equation (PDE) describing the fluid flow or the VOF colour function advection, an algebraic equation representing the discretised PDE is defined for each mesh cell as

$$A_P\,\phi_P + \sum_Q A_Q\,\phi_Q = b_P \ , \tag{3.102}$$

where P denotes the cell under consideration and Q denotes its neighbour cells. The right-hand side coefficient b_P contains all known terms, such as boundary conditions and data of previous time instants. Written in matrix form, the equation system follows as

$$\boldsymbol{A}\,\boldsymbol{\phi} = \boldsymbol{b} \ . \tag{3.103}$$

Matrix $\boldsymbol{A}$ is a squared matrix of size $(M \cdot N) \times (M \cdot N)$, with M being the number of discretised PDEs and N being the global number of mesh cell.

As a result of the explicit treatment of the surface force in the flow equation system, the momentum transport is numerically only stable if the propagation of capillary waves is resolved by the time-step Δt. According to Brackbill *et al.* [19], the capillary time-step constraint is defined as

$$\Delta t_c \leq \sqrt{\frac{(\rho_A + \rho_B)\ \Delta x^3}{4\pi\sigma}} \ , \tag{3.104}$$

where Δx denotes the mesh spacing and subscripts A and B denote the two fluids.

Solving the two equation systems separately provides the opportunity to use different time-steps for the colour function advection and for the fluid flow, satisfying the Courant number requirement for the colour function advection discussed in Section 3.4.3 without imposing the same strict Courant number limit to the fluid flow. Thus, the time-step of the colour function advection Δt_γ follows as

$$\Delta t_\gamma = \frac{\Delta t}{C_\gamma} \ , \tag{3.105}$$

where C_γ is a whole-number variable, determined at every time instant to fulfil the Courant number requirement for the colour function advection. Hence, for every fluid time-step Δt, the colour function advection is divided into C_γ time-steps Δt_γ.

In what follows, the individual coefficients and contributions to the equation systems describing the fluid flow and the VOF colour function advection are presented in detail. Subsequently, the sequence of the solution procedure is outlined.

3.7.1. Flow Equation System

As mentioned in Section 3.1, the solution procedure of the equation system representing the fluid flow follows a fully-coupled implicit approach. The aim of the fully-coupled implicit approach adopted in this work is to solve the governing equations of the fluid flow in one linear equation system. Therefore, a linear equation system is constructed from the discretised non-conservative momentum equations (Eq. 2.52) and the continuity constraint (Eq. 3.25), arranged as

$$\underbrace{\begin{pmatrix} \boldsymbol{A}_x & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{B}_x \\ \boldsymbol{0} & \boldsymbol{A}_y & \boldsymbol{0} & \boldsymbol{B}_y \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{A}_z & \boldsymbol{B}_z \\ \boldsymbol{C}_u & \boldsymbol{C}_v & \boldsymbol{C}_w & \boldsymbol{D} \end{pmatrix}}_{\boldsymbol{A}} \cdot \underbrace{\begin{pmatrix} \boldsymbol{\phi}_u \\ \boldsymbol{\phi}_v \\ \boldsymbol{\phi}_w \\ \boldsymbol{\phi}_p \end{pmatrix}}_{\boldsymbol{\phi}} = \underbrace{\begin{pmatrix} \boldsymbol{b}_x \\ \boldsymbol{b}_y \\ \boldsymbol{b}_z \\ \boldsymbol{b}_c \end{pmatrix}}_{\boldsymbol{b}} . \tag{3.106}$$

Inside matrix $\boldsymbol{A}$, submatrix $\boldsymbol{A}_i$ contains the coefficients of the velocity terms and submatrix $\boldsymbol{B}_i$ contains the coefficients of the pressure term of the i-th momentum equation. The coefficients of velocity component j following from the continuity constraint, derived using the momentum interpolation method presented in Section 3.3, are placed in submatrix $\boldsymbol{C}_j$. Submatrix $\boldsymbol{D}$ holds the coefficients of the pressure term of the continuity constraint. Solution vector $\boldsymbol{\phi}$ is constituted by the solution subvectors of the three velocity components u, v and w, and pressure p. Every row of the equation system is normalised by the respective diagonal coefficient of matrix $\boldsymbol{A}$ to improve the convergence of the numerical solution algorithm, particularly on unstructured meshes.

Discretising the momentum equations for the three Cartesian coordinates using the numerical schemes presented in Section 3.2 leads to a set of coefficients which are placed in the respective positions of the coefficient matrix $\boldsymbol{A}$ and to contributions to the right-hand side vector $\boldsymbol{b}$. Applying the Second-Order Backward Euler scheme to discretise the transient term of the momentum equation results in a contribution to the diagonal coefficient

$$A_{i,P} \mathrel{+}= \frac{\Delta\tau^2 - \Delta t_1^2}{\Delta\tau^2\,\Delta t_1 - \Delta\tau\,\Delta t_1^2}\,\rho_P^t\,V_P \ , \tag{3.107}$$

and the right-hand side vector

$$b_{i,P} \mathrel{+}= \left(\frac{\Delta\tau}{\Delta\tau\,\Delta t_1 - \Delta t_1^2}\; u_i^{t-\Delta t_1} - \frac{\Delta t_1}{\Delta\tau^2 - \Delta\tau\,\Delta t_1} u_i^{t-\Delta\tau} \right) \rho_P^t\,V_P \ , \tag{3.108}$$

where subscript i denotes any of the three Cartesian coordinate axes. The convective term of the momentum equations contributes to the diagonal coefficients

$$A_{i,P} \mathrel{+}= \begin{cases} \sum_f (1-\psi\, l_D)\, \dot{m}_f \ , & \text{when} \quad \dot{m}_f \geq 0 \\ \sum_f \psi\, l_D\, \dot{m}_f \ , & \text{when} \quad \dot{m}_f < 0 \end{cases} \tag{3.109}$$

and the off-diagonal coefficients

$$A_{i,Q} \mathrel{+}= \begin{cases} \sum_f \psi\, l_D\, \dot{m}_f \ , & \text{when} \quad \dot{m}_f \geq 0 \\ \sum_f (1-\psi\, l_D)\, \dot{m}_f \ , & \text{when} \quad \dot{m}_f < 0 \end{cases} \tag{3.110}$$

of the matrix, with mass flux $\dot{m}_f$ being defined as given in Eq. 3.26. The coefficients and right-hand side contribution resulting from the viscous stresses using a deferred correction approach are

$$A_{i,P} \quad \mathrel{+}= \quad \sum_f -\frac{\mu_f\, \alpha_f\, A_f}{\Delta s} \tag{3.111}$$

$$A_{i,Q} \quad \mathrel{+}= \quad \sum_f \frac{\mu_f\, \alpha_f\, A_f}{\Delta s} \tag{3.112}$$

$$b_{i,P} \quad \mathrel{-}= \quad \sum_f \mu_f\, A_f \left[\left. \overline{\frac{\partial u_i}{\partial x_j}} \right|_f (\boldsymbol{n}_f - \alpha_f\, \boldsymbol{s}_f) + \left. \overline{\frac{\partial u_j}{\partial x_i}} \right|_f \boldsymbol{n}_f \right] \ . \tag{3.113}$$

The velocity gradients of Eq. 3.113 are averaged as described in Section 3.2.3. The pressure term of the momentum equation only yields off-diagonal matrix coefficients, following as

$$B_{i,P} \quad = \quad \sum_f (1-\delta)\ n_{i,f}\ A_f \tag{3.114}$$

$$B_{i,Q} \quad = \quad \sum_f \delta\ n_{i,f}\ A_f \ , \tag{3.115}$$

and, if the mesh is skewed, a contribution to the right-hand side vector

$$b_{i,P} \mathrel{-}= \sum_f n_{i,f}\, A_f\, \left.\overline{\nabla p}\right|_f \boldsymbol{r}_f \ . \tag{3.116}$$

The gravity term, if applicable, results in a straightforward contribution to the right-hand side vector

$$b_{i,P} \mathrel{+}= S_{gi,P}\, V_P \ . \tag{3.117}$$

and, similarly, the surface force only contributes to the right-hand side vector with

$$b_{i,P} \mathrel{+}= \sigma\, \kappa_P \left. \frac{\partial \gamma}{\partial x_i} \right|_P V_P \ . \tag{3.118}$$

The continuity constraint, presented in Section 3.3, represents the fourth equation of the linear equation system, as mentioned previously. For each cell P the velocity term of the continuity constraint results in the off-diagonal coefficients

$$\begin{aligned}
C_{u,P} &= \sum_f (1-\delta)\, n_{x,f}\, A_f && (3.119)\\
C_{u,Q} &= \sum_f \delta\, n_{x,f}\, A_f && (3.120)\\
C_{v,P} &= \sum_f (1-\delta)\, n_{y,f}\, A_f && (3.121)\\
C_{v,Q} &= \sum_f \delta\ n_{y,f}\, A_f && (3.122)\\
C_{w,P} &= \sum_f (1-\delta)\, n_{z,f}\, A_f && (3.123)\\
C_{w,Q} &= \sum_f \delta\ n_{z,f}\, A_f\ . && (3.124)
\end{aligned}$$

In the presence of mesh skewness the correction of the interpolated velocity at face centres contributes with

$$b_{c,P} \mathrel{-}= \sum_f \left[\left((1-\delta)\ \nabla \boldsymbol{u}|_P + \delta\ \nabla \boldsymbol{u}|_Q \right) \boldsymbol{r}_f \right] \boldsymbol{n}_f\, A_f \tag{3.125}$$

to the right-hand side vector. The transient term, consisting only of values from the previous time instant, adds to the right-hand side vector and is defined as

$$b_{c,P} \mathrel{-}= \sum_f c_f\, \hat{d}_f\, A_f\ \left(u_f^{n,t-\Delta t} - \boldsymbol{u}_f^{t-\Delta t} \boldsymbol{n}_f \right)\ . \tag{3.126}$$

The pressure term of the continuity constraint crucially contributes to the diagonal coefficient of the matrix

$$D_P = \sum_f \frac{\hat{d}_f\, \alpha_f\, A_f}{\Delta s}\ , \tag{3.127}$$

assuring a non-zero diagonal of the matrix, as well as to the off-diagonal coefficients

$$D_Q = \sum_f - \frac{\hat{d}_f\, \alpha_f\, A_f}{\Delta s} \tag{3.128}$$

and the right-hand side vector

$$b_{c,P} \mathrel{+}= \sum_f \frac{\hat{d}_f\, \alpha_f\, \rho_f\, A_f}{2} \left(\frac{\nabla p|_P}{\rho_P} + \frac{\nabla p|_Q}{\rho_Q} \right) \boldsymbol{s}_f\ . \tag{3.129}$$

If applicable, the gravity term

$$b_{c,P} \mathrel{-}= \sum_f \alpha_f\, \hat{d}_f\, A_f\ \left[\rho_f\, (\boldsymbol{g} \cdot \boldsymbol{s}_f) - \frac{\rho_f}{2} \left(\frac{\boldsymbol{S}_{g,P}}{\rho_P} + \frac{\boldsymbol{S}_{g,Q}}{\rho_Q} \right) \boldsymbol{s}_f \right] \tag{3.130}$$

and the surface force term

$$b_{c,P} \mathrel{-}= \sum_f \alpha_f \, \hat{d}_f \, \sigma \left[\kappa_f \frac{\gamma_Q - \gamma_P}{\Delta s} - \frac{\rho_f}{2} \left(\frac{\kappa_P \nabla\gamma|_P}{\rho_P} + \frac{\kappa_Q \nabla\gamma|_Q}{\rho_Q} \right) \boldsymbol{s}_f \right] \tag{3.131}$$

contribute to the right-hand side vector.

The solution of the equation system is performed in a time-marching fashion, starting with an initial set of values. To account for the non-linearity of the equations, the mass fluxes are calculated using the deferred advecting velocity u_f^n, as defined in Eq. 3.26, and non-linear iterations are performed within each time-step, known as *inexact Newton* method [48, 186]. Typically, $2-50$ non-linear iterations are required at every time instant to yield a converged result. The number of non-linear iterations is dependent on the flow characteristics, the mesh size and the solution tolerance. Initially in each time-step, a converged implicit solution for the linear equation system is found, using the result of the previous time instant as an initial guess. Subsequently, the spatial gradients of the primitive variables and the new mass fluxes are calculated, the equation system is updated and solved again. This iterative procedure continues until the non-linear problem converged to a sufficiently small residual. The linear equation system is preconditioned using a JACOBI method and solved by an enhanced BiCGSTAB method [218]. The iterative solver and the preconditioner are incorporated in the freely-available PETSc library [11, 12], which is integrated in the presented numerical framework to handle and solve linear equation systems. In general, the numerical framework and solving sequence are independent of the iterative solver or preconditioning method used and the presented choice represents only one possible option.

3.7.2. VOF Equation System

A linear equation system is constructed from the discretised equations describing the colour function transport in each mesh element, which is solved utilising an iterative solving method. The transient discretisation, presented in Section 3.4.1, is included in the equation system as

$$A_P \mathrel{+}= \frac{V_P}{\Delta t_\gamma} \tag{3.132}$$

$$b_P \mathrel{+}= \frac{V_P}{\Delta t_\gamma} \, \gamma_P^{t-\Delta t_\gamma} \; . \tag{3.133}$$

The time-step Δt_γ represents the time-step applied to advect the colour function, which, as previously mentioned, might be smaller than the time-step used to calculate the flow. Discretising the advection term using the CICSAM scheme, as presented in Section 3.4.2, results in the contributions to the diagonal coefficients of the matrix

$$A_P \mathrel{+}= \sum_f \frac{1}{2} \, w_{\beta,P} \, F_f \; , \tag{3.134}$$

the off-diagonal coefficients

$$A_Q \mathrel{+}= \sum_f \frac{1}{2}\, w_{\beta,Q}\, F_f \tag{3.135}$$

and the right-hand side vector

$$b_P \mathrel{-}= \sum_f \frac{w_{\beta,P}\ \gamma_P^{t-\Delta t_\gamma} + w_{\beta,Q}\ \gamma_Q^{t-\Delta t_\gamma}}{2}\ F_f\ , \tag{3.136}$$

where subscript P denotes the element under consideration and subscript Q denotes its neighbour elements. If cell P is the acceptor cell with respect to its adjacent face f, the weighting factors resulting from the CICSAM scheme are

$$w_{\beta,P} = \beta_f \tag{3.137}$$
$$w_{\beta,Q} = 1 - \beta_f\ , \tag{3.138}$$

and if cell P is the donor cell of face f, the weighting factors are

$$w_{\beta,P} = 1 - \beta_f \tag{3.139}$$
$$w_{\beta,Q} = \beta_f\ . \tag{3.140}$$

As for the flow equation system, every row of the VOF equation system is normalised with the respective diagonal coefficient of the matrix to improve convergence of the numerical solving algorithm.

Similar to the solution of the flow equation system discussed in Section 3.7.1, the VOF equation system is preconditioned with a JACOBI method and solved using a standard BiCGSTAB method [244], both incorporated in the PETSc library [11, 12]. Initially in each time-step, the CICSAM weighting factor β is calculated for each mesh face and the VOF equation system is assembled and solved. Following the solution of the equation system, the CICSAM weighting factors β_f are corrected if the resulting colour function values are not bounded and the equation system is solved again. This iterative predictor-corrector procedure continues until the colour function value at every mesh cell satisfies the bounds defined by the predefined solver tolerance. After a converged and bounded solution has been found, time-step Δt_γ is incremented and the sequence starts again. This iterative procedure continues for C_γ time-steps Δt_γ, at which point the new colour function distribution for the present fluid time-step Δt is found.

3.7.3. Solution Sequence

At the beginning of every time instant, the data of previous time instants (velocity, pressure, flux and colour function) are copied to the respective arrays, *e.g.* velocity data that are stored as data of the previous time instant are declared as data of the previous-previous time instant.

Subsequently, the VOF equation system is assembled and solved as described in Section 3.7.2. The colour function is advected based on the flow field resulting from the previous time instant. Because the colour function is advected prior to computing the flow field at a given time instant, the pressure field is calculated based on the new interface position. In contrast, updating the interface position after the new flow field has been calculated would result in a lag between pressure distribution and interface position for any given time instant. When a new interface position is found, the fluid properties and the colour function gradient are updated and the new interface curvature is calculated.

The new interface position is then used to calculate the new gravity source term. After that, the coefficient matrix and the right-hand side vector of the flow equation system are assembled and the flow equation system is solved as described in Section 3.7.1. When a converged result for the equation system is found, the spatial gradients of the primitive variables are evaluated and the new face fluxes are calculated. Subsequently, the residual of the non-linear iteration is determined and, if the predefined convergence criteria are not satisfied, the flow equation system is updated with the new fluxes and new spatial gradients of the primitive variables and solved again. As soon as a converged result for the non-linear flow field is reached, the maximum Courant number is updated and, if necessary, the time-step is adapted to fulfil the predefined Courant limit.

3.8. Validation

The validation of the numerical framework focuses on demonstrating the ability of simulating single-phase and two-phase flows on structured and unstructured meshes with comparable accuracy. Furthermore, the correct treatment of surface tension, the accurate balancing of body forces and pressure gradient as well as the satisfaction of continuity and the conservation of the colour function are demonstrated.

3.8.1. Fluid under Gravity

A fluid under gravity in a closed container is simulated to verify the correct description of the gravity force term. Assuming a stationary fluid, the pressure gradient must precisely match the gravity force acting on the fluid, as described in Eq. 3.51. The computational domain is cubical with an edge length of $1\,m$, the density of the fluid is $\rho = 1\,kg\,m^{-3}$ and the gravitational acceleration acting on the fluid is $g = 10\,m\,s^{-2}$. Therefore, assuming zero pressure at the top of the domain, the pressure at the bottom of the domain is $p = g\,\rho\,y = 10\,Pa$. Two meshes are considered for this test case, an equidistant Cartesian mesh with $80 \times 80 \times 80$ cells and a tetrahedral mesh with approximately 4.8×10^5 cells, both illustrated in Figure 3.3. Figure 3.4 shows the pressure distribution resulting from gravity computed by the presented numerical framework. On both the Cartesian mesh as well as the tetrahedral mesh the computed pressure distribution accurately reproduces the analytical pressure distribution.

3.8.2. Hagen-Poiseuille Flow

The pressure drop of a flow with constant velocity in a circular pipe due to friction, known as Hagen-Poiseuille flow, is evaluated to validate the accurate interaction of inertial mo-

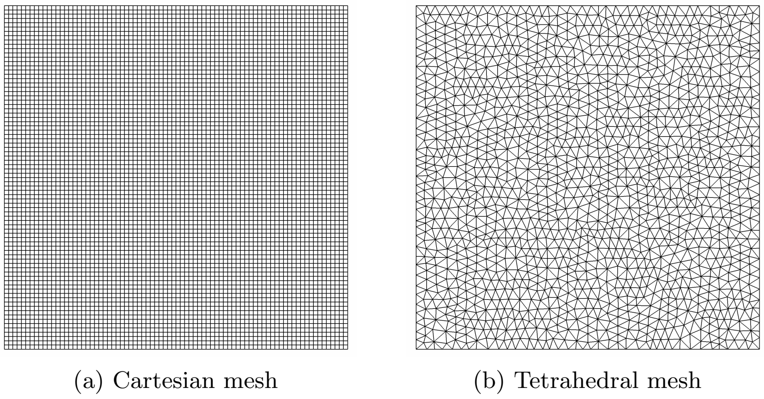

(a) Cartesian mesh (b) Tetrahedral mesh

Figure 3.3.: The equidistant Cartesian mesh and the tetrahedral mesh used to validate the correct description of the gravity force by the presented numerical framework.

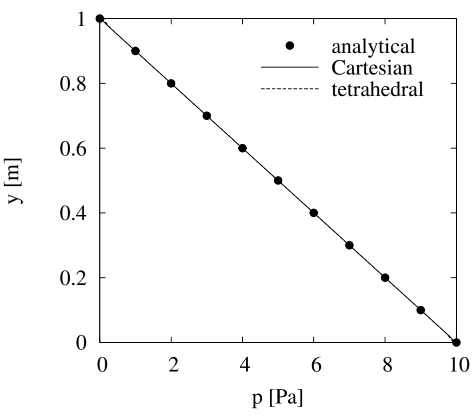


Figure 3.4.: Pressure profile of a fluid with density $1\,kg\,m^{-3}$ in a cubical domain with edge length $1\,m$ due to gravitational acceleration of $10\,m\,s^{-2}$.

mentum, viscous stresses and pressure. For a flow in a circular pipe, a pressure difference between two distinct points develops in the axial direction of the pipe as a result of viscous stresses. Assuming an axisymmetric flow, the axial velocity component u_x with respect to the radial coordinate $\tilde{r}$ is [13]

$$u_x(r) = \frac{p_0 - p_1}{4\,\mu\,L}(r - \tilde{r}) \ , \tag{3.141}$$

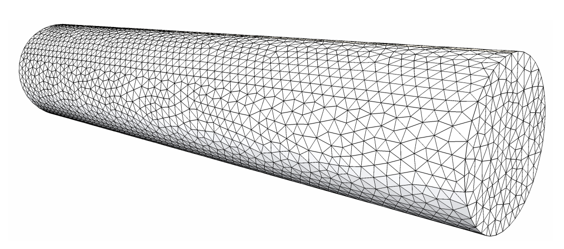
where L is the length of the considered section of the circular pipe, r is the radius of the pipe and p_0 and p_1 is the pressure at the beginning and at the end of the considered section of the pipe, respectively. The volume flux F through the pipe follows by integrating the axial velocity over the radius

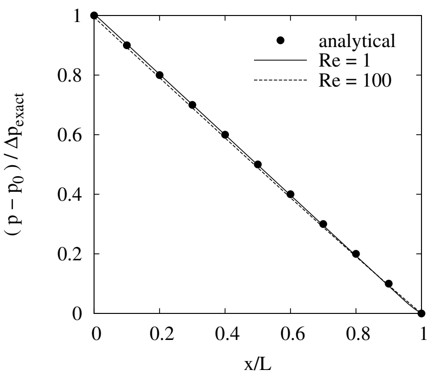
$$F = \int_0^r 2\pi\, u_x\, \tilde{r}\ dr = \frac{\pi\, r^4\, (p_0 - p_1)}{8\,\mu\, L} \ . \tag{3.142}$$

The volume flux through the pipe is also readily available as $F = u_x \, \pi \, r^2$ and the expected pressure jump is, therefore,

$$\Delta p = p_0 - p_1 = \frac{8 \, \mu \, u_x \, L}{r^2} \; . \tag{3.143}$$

The pressure drop predicted on a tetrahedral mesh with approximately 9.5×10^4 cells, shown in Figure 3.5a, for Reynolds numbers $Re_r = u_x \, \rho \, r / \mu = 1$ and $Re_r = 100$ is depicted in Figure 3.5b, together with the analytical result from Eq. 3.143. The numerical framework accurately predicts the pressure difference in the circular pipe for both Reynolds numbers.


(a) Computational mesh

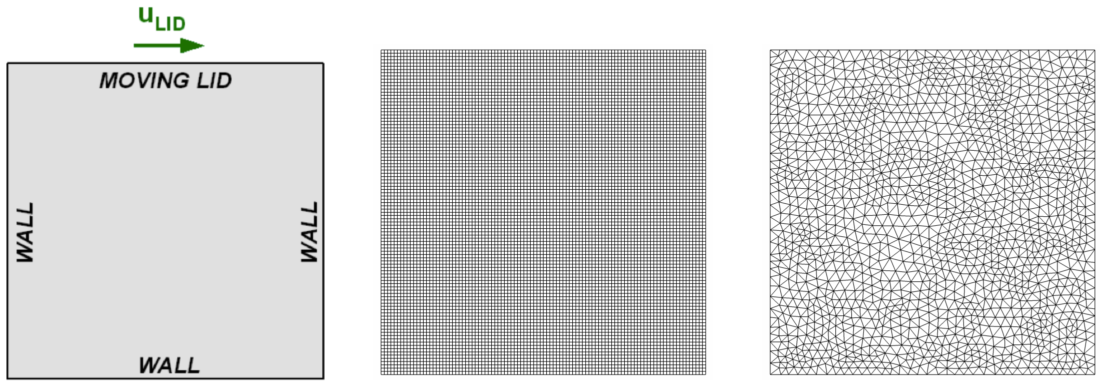


(b) Pressure profile

Figure 3.5.: Normalised pressure drop, computed and analytical, as a function of non-dimensional length for a flow with constant volume flux in a circular pipe discretised by a tetrahedral mesh of approximately 9.5×10^4 cells.

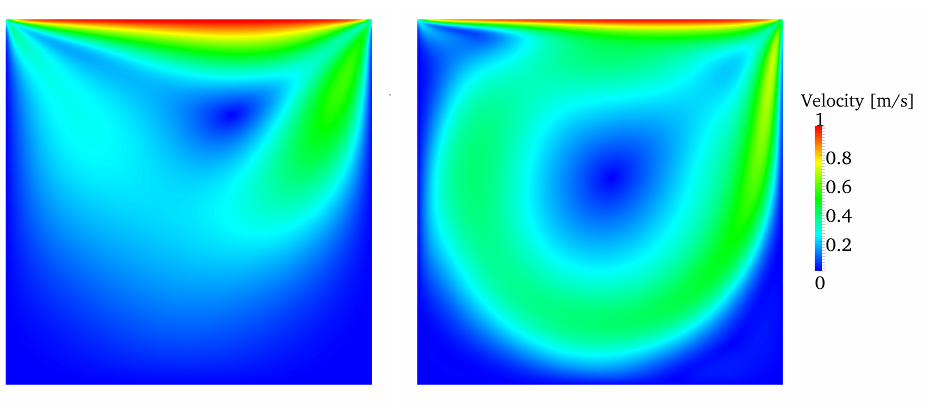
3.8.3. Lid-Driven Cavity

The lid-driven cavity is a common test case to validate numerical methods for fluid flows because it captures convective and viscous transport of the fluid. Ghia *et al.* [77] published a vast amount of high-fidelity numerical reference data of the flow in a lid-driven cavity for various Reynolds numbers. Because the laminar solution of the lid-driven cavity is steady, $Re_L = u_w \, \rho \, L / \mu = 100$ and $Re_L = 1000$ are considered to validate the numerical framework, with u_w representing the constant velocity of the top wall, *i.e.* the lid, and L denoting the edge length of the domain. Although the results for the laminar cases as well as the reference results of Ghia *et al.* [77] are two-dimensional, the simulations are carried out in three-dimensional domains because the implementation of the numerical framework requires three spatial dimensions. However, because the cavity walls perpendicular to the third dimension hold a free-slip boundary condition, no differences compared to the two-dimensional results are expected by using three-dimensional domains, apart from an overhead in computational resources. Figure 3.6a illustrates schematically the domain with its boundary conditions. The side walls and the bottom wall are stationary and hold a no-slip condition. A no-slip condition is also imposed at the top wall, which moves at a constant velocity u_w and, thus, drives the flow in the cavity. Two equidistant Cartesian meshes, $100 \times 100 \times 5$ cells and $200 \times 200 \times 5$ cells, and two tetrahedral meshes, 5.3×10^4 cells and 2.4×10^5 cells, are deployed. The two coarser meshes are shown in Figures 3.6b and 3.6c. To assist the understanding of the resulting flow in the lid-driven cavity, Figure 3.7 shows the contours of the computed velocity distribution for the two considered Reynolds numbers. The vortex developing in the cavity is rotating clockwise.



(a) Boundary conditions (b) Cartesian mesh (c) Tetrahedral mesh

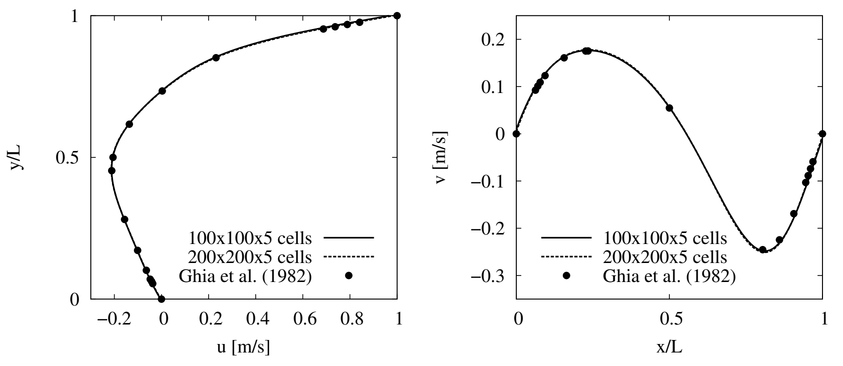
Figure 3.6.: Boundary conditions and meshes applied for the lid driven cavity.



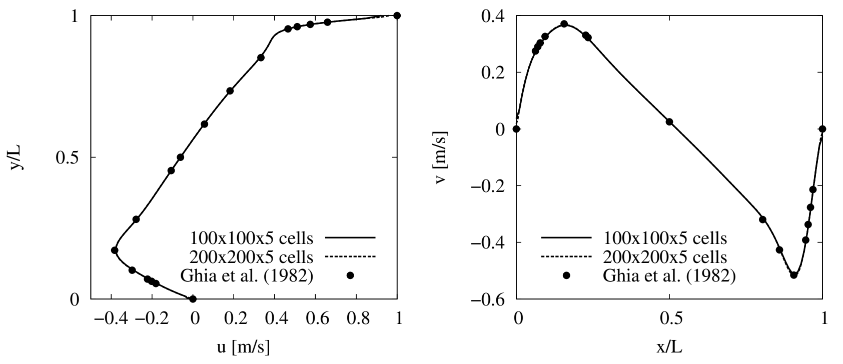
(a) $Re = 100$ (b) $Re = 1000$

Figure 3.7.: Contours of the velocity magnitude of the flow in the x-y plane running through the centre of the lid driven cavity domain.

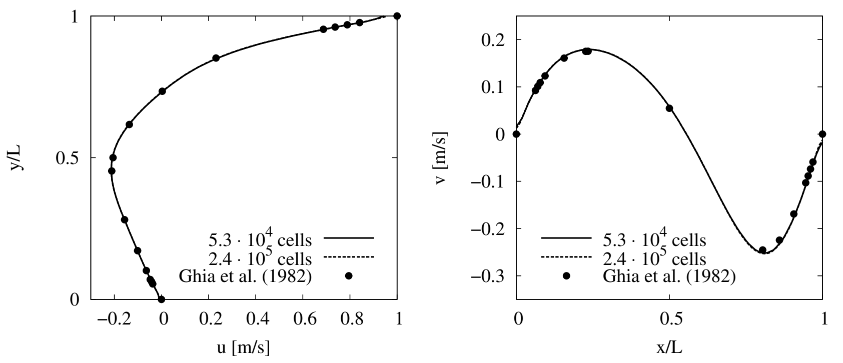
Figure 3.8 shows the velocity profiles obtained for the considered Reynolds numbers in the two relevant dimensions along a line through the domain centre on the Cartesian meshes and the tetrahedral meshes. The results of both Cartesian meshes and the reference results of Ghia *et al.* [77] are in excellent agreement. The results obtained on the tetrahedral meshes are in equally good agreement with each other as well as with the reference data. The only minor disagreement observable in the graphs occurs at the walls, where the simulation results do not exactly reproduce the velocity predefined at the walls.



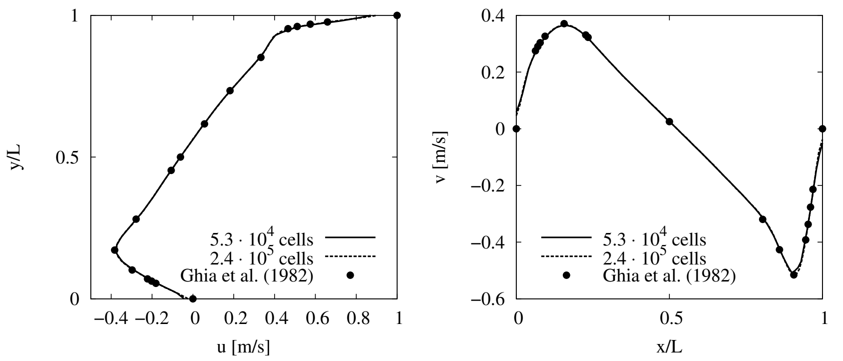
(a) $Re = 100$, Cartesian meshes



(b) $Re = 1000$, Cartesian meshes



(c) $Re = 100$, tetrahedral meshes



(d) $Re = 1000$, tetrahedral meshes

Figure 3.8.: Velocity profiles of the flow in a lid driven cavity at two different Reynolds numbers on equidistant Cartesian meshes and tetrahedral meshes along the vertical and horizontal lines through the domain centre. The results of Ghia *et al.* [77] are included as a reference.

This disagreement is merely a post-processing issue, resulting from the finite distance between the cell centre closest to the wall, where the velocity value is stored, and the actual cell face holding the boundary condition. Nevertheless, the results for both mesh types are overall in very good agreement with the reference data and are not affected by the mesh type.

3.8.4. Surface Tension

The balanced-force implementation of the presented numerical framework is verified using a stationary and a moving surface-tension-dominated interface. The numerical framework is designed to provide an exact balance between pressure gradient and surface force (as well as other body forces). Test cases to verify the balance between pressure gradient and surface force have to satisfy certain conditions. Firstly, other body forces, such as gravity, should be absent. Secondly, the interface should be spherical so that the interface is in equilibrium and the geometrically exact interface curvature, defined as

$$\kappa = \frac{2}{r} \ , \tag{3.144}$$

where r is the radius of the spherical interface, can be imposed. Thus, the interface curvature does not have to be calculated numerically, eliminating errors arising from the interface curvature evaluation. Thirdly, all velocity gradients should be negligible, thus, the momentum equation is linear and reduces to Eq. 3.58. Following these conditions, the pressure gradient and the surface force are in equilibrium by definition and the pressure jump across the interface is defined by the Young-Laplace equation given in Eq. 2.46. Therefore, errors in pressure jump across the interface and velocity magnitudes larger than the solver tolerance (or machine precision) are the result of a numerical imbalance between pressure gradient and surface force. Francois *et al.* [71] and Mencinger and Žun [154] presented force-balancing at stationary interfaces and Denner and van Wachem [54, 55] recently demonstrated force-balancing at a moving interface, the results of which are presented in Section 3.8.4.2.

3.8.4.1. Stationary Interface

The content of this section has in parts been published in:
[55] Denner, F. and van Wachem, B.G.M.: Fully-coupled balanced-force VOF framework for arbitrary meshes with least-squares curvature evaluation from volume fraction. Numerical Heat Transfer, Part B: Fundamentals, accepted for publication. DOI: 10.1080/10407790.2013.849996

The force-balancing of the numerical framework at a stationary interface is evaluated using an inviscid static drop in mechanical equilibrium with the specifications previously used by Francois *et al.* [71]. The inviscid drop with surface tension coefficient $\sigma = 73\, N\, m^{-1}$ and radius $r = 2\, m$ is positioned at the centre of a three-dimensional cubical domain with edge length $8\, m$. No gravity is present. As previously mentioned, the pressure jump across the interface for a stationary spherical fluid particle in equilibrium and zero gravity is given by the Young-Laplace equation (Eq. 2.46) and the interface curvature κ for a three-dimensional spherical fluid particle is defined as in Eq. 3.144. As a result, the exact pressure jump across the interface of the considered drop is $\Delta p_{exact} = 73\,Pa$. In order to validate the balanced-force surface force implementation, the geometrically exact curvature $\kappa = 1\,m^{-1}$ is applied to calculate the surface force of the static inviscid drop in equilibrium. If the implementation of pressure gradient and surface force is in balance, the developing parasitic currents and the pressure error must be of the order of the solver tolerance. Three different meshes are deployed, shown in Figure 3.9: an equidistant Cartesian mesh with $40 \times 40 \times 40$ cells, a non-orthogonal hexahedral mesh with $40 \times 40 \times 40$ cells and a tetrahedral mesh with approximately 6.0×10^4 cells.

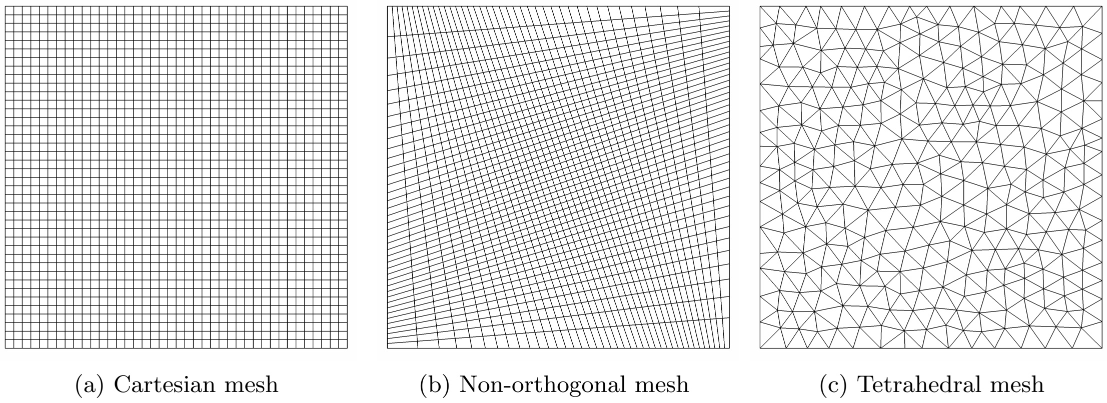

(a) Cartesian mesh (b) Non-orthogonal mesh (c) Tetrahedral mesh

Figure 3.9.: The three different meshes used to validate the balanced-force implementation of the surface force in the presented numerical framework for a stationary interface.

The simulations are performed with two different time-steps Δt and four different density ratios ρ_i/ρ_o. Although theoretically the density ratio should not influence the outcome of the simulations, a higher density ratio multiplies the acceleration term of the momentum equations and, thus, even small imbalances in the numerical framework can reduce the accuracy notably or cause stability issues, as observed with respect to the proposed density weighting in the following paragraph. Up to date only Francois *et al.* [71] have presented balanced-force results for two-phase flows with density ratios $\geq 10^6$. The maximum parasitic currents $|\boldsymbol{u}|_{max}$ and the pressure error

$$E(\Delta p_{max}) = \frac{|\Delta p_{max} - \Delta p_{exact}|}{\Delta p_{exact}} \tag{3.145}$$

for the different meshes are presented in Table 3.1. In Eq. 3.145, Δp_{max} represents the difference between the maximum and the minimum pressure in the entire domain. The results show no considerable parasitic currents developing on the tested meshes with the applied density ratios, *i.e.* the magnitude of the parasitic currents is of equal magnitude or smaller magnitude than the applied solver tolerance $\epsilon_{rel} = 10^{-15}$. The resulting pressure error has a vanishingly small magnitude as well. The pressure profile along a line parallel to the x-axis and crossing through the centre of the drop for a density ratio of $\rho_i/\rho_o = 10^9$

and a time-step of $\Delta t = 10^{-3}\,s$, given in Figure 3.10, shows no pressure oscillations in the vicinity of the interface. On both the equidistant Cartesian mesh, shown in Figure 3.10a, and the tetrahedral mesh, shown in Figure 3.10b, the numerical framework is able to predict the pressure profile accurately, as the comparison with the exact pressure profile following from the Young-Laplace equation (Eq. 2.46) demonstrates. Hence, the numerical framework is maintaining an exact balance between pressure gradient and surface force at a stationary interface.

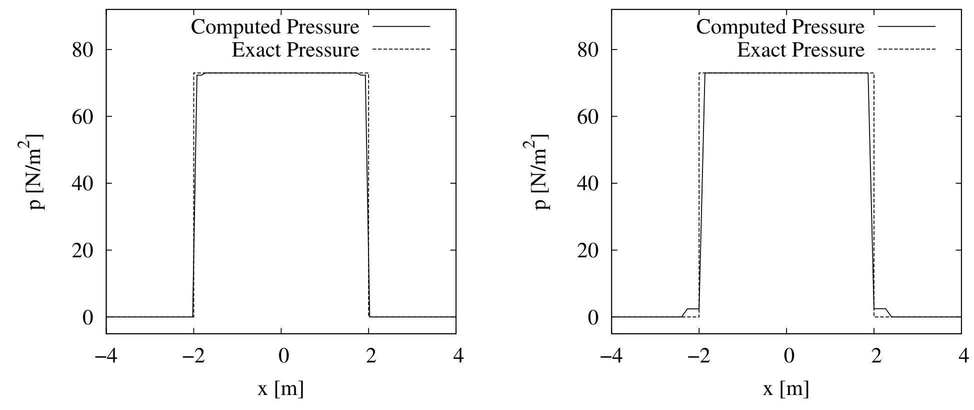
Table 3.1.: Maximum velocity magnitude and pressure error after one time-step for an inviscid static drop in equilibrium with exact curvature. The static inviscid drop with surface tension coefficient $\sigma = 73\,N\,m^{-1}$ and radius $r = 2\,m$ is positioned at the center of the $8\,m \times 8\,m \times 8\,m$ domain.

Mesh	ρ_i/ρ_o	$\lvert\boldsymbol{u}\rvert_{max}[m/s]$		$E(\Delta p_{max})$	
		$\Delta t = 10^{-3}s$	$\Delta t = 10^{-6}s$	$\Delta t = 10^{-3}s$	$\Delta t = 10^{-6}s$
Cartesian 40^3 cells	10^0	4.924×10^{-16}	3.236×10^{-19}	3.407×10^{-14}	1.869×10^{-14}
	10^3	3.545×10^{-16}	1.440×10^{-19}	2.920×10^{-14}	7.981×10^{-15}
	10^6	2.753×10^{-16}	2.736×10^{-19}	1.869×10^{-14}	1.888×10^{-14}
	10^9	3.867×10^{-16}	1.379×10^{-19}	2.842×10^{-14}	1.143×10^{-13}
Non-orthogonal 40^3 cells	10^0	1.193×10^{-15}	1.102×10^{-18}	3.816×10^{-14}	2.180×10^{-14}
	10^3	1.152×10^{-15}	1.253×10^{-18}	3.660×10^{-14}	7.125×10^{-14}
	10^6	1.118×10^{-15}	1.144×10^{-18}	6.541×10^{-14}	3.991×10^{-14}
	10^9	1.164×10^{-15}	2.461×10^{-18}	5.061×10^{-14}	1.781×10^{-13}
Tetrahedral $\approx 6 \times 10^4$ cells	10^0	7.916×10^{-16}	9.366×10^{-19}	2.667×10^{-14}	2.823×10^{-14}
	10^3	9.967×10^{-16}	1.325×10^{-18}	5.470×10^{-14}	8.040×10^{-14}
	10^6	1.540×10^{-15}	1.484×10^{-18}	9.052×10^{-14}	8.643×10^{-14}
	10^9	1.352×10^{-15}	2.468×10^{-18}	7.884×10^{-14}	1.141×10^{-13}

The development of the residual of the numerical solution for the inviscid static drop in equilibrium is examined in order to assess the potential of weighting the pressure term and the body force terms of the advecting velocity by the ratio of face density and cell density, as proposed in Eq. 3.57. The equidistant Cartesian mesh with $40 \times 40 \times 40$ cells is deployed and time-step $\Delta t = 10^{-3}\,s$ is applied. The pressure field is initialised with $p = 0\,Pa$ in the entire domain. The first time-step is numerically particularly challenging in this situation because the pressure jump across the interface has to be developed by the solving algorithm. Figure 3.11 depicts the maximum residual, defined as

$$R_{max} = \max\left\{\left|u_P^i - u_P^{i-1}\right|, \left|v_P^i - v_P^{i-1}\right|, \left|w_P^i - w_P^{i-1}\right|, \left|p_P^i - p_P^{i-1}\right|\right\} \ , \tag{3.146}$$

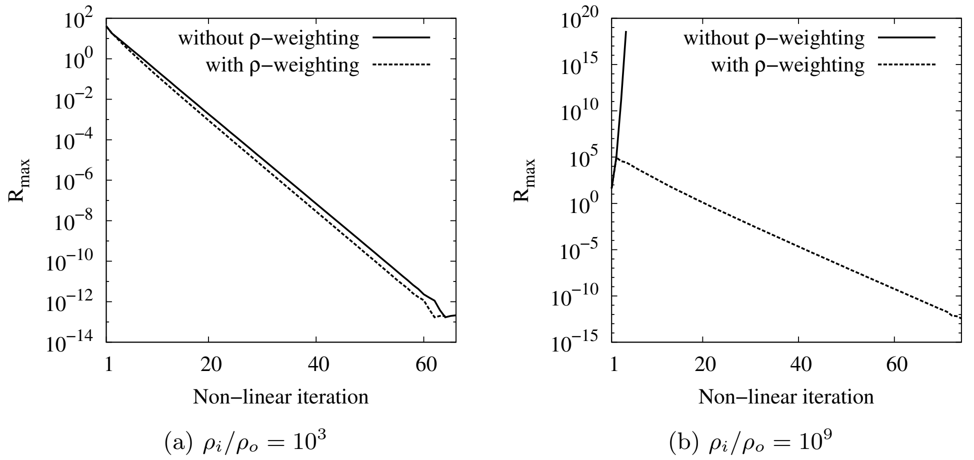
as a function of non-linear iterations for the first time-step of the simulation. In Eq. 3.146 superscripts i and $i-1$ denote the values of the present iteration and the previous iteration, respectively. For a density ratio of $\rho_i/\rho_o = 10^3$, shown in Figure 3.11a, the first time-step converges regardless whether the pressure term and the surface force term of the advecting velocity are density-weighted or not. If the density ratio of the two fluids



(a) Cartesian mesh (b) Tetrahedral mesh

Figure 3.10.: The computed pressure profile and the exact pressure profile, as defined by the Young-Laplace equation (Eq. 2.46), along a line crossing through the centre of the domain on an equidistant Cartesian and a tetrahedral mesh. The static inviscid drop with surface tension coefficient $\sigma = 73\,N\,m^{-1}$, density ratio $\rho_i/\rho_o = 10^9$ and radius $r = 2\,m$ is positioned at the center of the $8\,m \times 8\,m \times 8\,m$ domain.

is increased to $\rho_i/\rho_o = 10^9$, however, shown in Figure 3.11b, the simulation diverges if no density-weighting is applied. On the other hand, including the density-weighting proposed in Eq. 3.57 in the momentum interpolation method provides a stable convergence even for a density ratio as high as $\rho_i/\rho_o = 10^9$.



(a) $\rho_i/\rho_o = 10^3$ (b) $\rho_i/\rho_o = 10^9$

Figure 3.11.: Maximum residual R_{max} of the first time-step as a function of non-linear iterations of the inviscid static drop in equilibrium for two density ratios, with and without density-weighting of the pressure term and the surface force term of the advecting velocity.

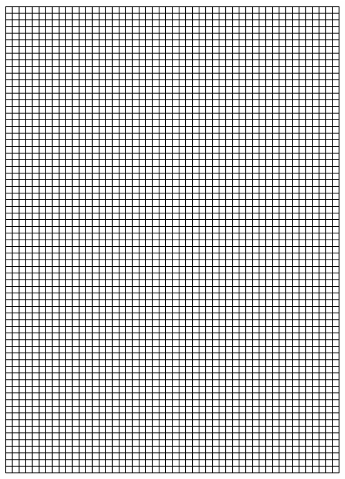
3.8.4.2. Moving Interface

The content of this section has been published in:

[54] Denner, F. and van Wachem, B.G.M.: Force-balancing at moving surface-tension-dominated interfaces on collocated unstructured meshes. 8th International Conference on Multiphase Flow (ICMF 2013), 26 - 31 May 2013, Jeju, Korea.

[55] Denner, F. and van Wachem, B.G.M.: Fully-coupled balanced-force VOF framework for arbitrary meshes with least-squares curvature evaluation from volume fraction. Numerical Heat Transfer, Part B: Fundamentals, accepted for publication. DOI: 10.1080/10407790.2013.849996

As mentioned previously, any imbalance between body forces and pressure gradient manifests itself in an erroneous prediction of the pressure jump across the interface and in parasitic currents. The force-balancing at a moving interface is assessed by means of a spherical fluid particle moving at a constant velocity (*i.e.* velocity gradients are negligible) through a rectangular channel. Both fluids have the same constant velocity and, thus, there is no slip at the interface. The viscosity ratio of both fluids is unity and the considered density ratios ρ_i/ρ_o are 10^0, 10^3 and 10^6. The applied Reynolds number and the applied capillary number of the moving interface are $Re_d = |\boldsymbol{u}|\, \rho_i\, d/\mu = 0.01$ and $Ca = |\boldsymbol{u}|\, \mu/\sigma = 0.01$, respectively. The geometrically exact curvature, as defined in Eq. 3.144, is imposed as a fixed constant at the interface. The simulations are performed on an equidistant Cartesian mesh with $50 \times 70 \times 50$ cells, shown in Figure 3.12a, and a tetrahedral mesh with approximately 1.5×10^5 cells, shown in Figure 3.12b.


(a) Cartesian mesh

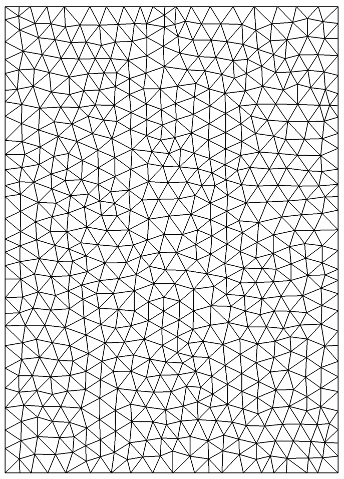

(b) Tetrahedral mesh

Figure 3.12.: The equidistant Cartesian mesh and the tetrahedral mesh used to validate the balanced-force implementation of the surface force of the presented numerical framework for a moving interface.

Figure 3.13 presents the maximum error of the pressure jump across the interface as a function of time on the equidistant Cartesian mesh and the tetrahedral mesh. On the Cartesian mesh the pressure error is negligible as it is of the same order of magnitude as the solver tolerance $\epsilon_{rel} = 10^{-10}$, regardless of the applied density ratio. The pressure error observed on the tetrahedral mesh is one order of magnitude higher than on the Cartesian mesh as the solver reaches the predefined maximum number of non-linear iterations per time-step (50 iterations). A fully converged result, *i.e.* an error equal to the solver tolerance, could be achieved by raising the maximum number of non-linear iterations, however, at a significantly higher computational cost. Figure 3.14 shows a similar result for the maximum velocity error in the domain, being essentially negligible on both types of meshes and regardless of the applied density ratio. The results, therefore, prove that the presented numerical framework provides an accurate balance between pressure gradient and surface force for moving interfaces as well.

If the surface force is not included in the definition of the advecting face velocity (see Eq. 3.60), the pressure and velocity errors increase by several orders of magnitude, as

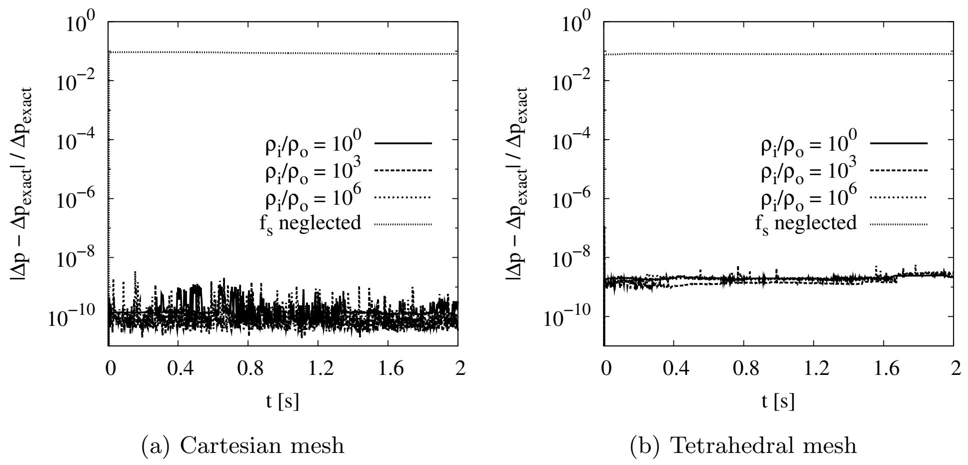


Figure 3.13.: Pressure error for an interface moving at a constant velocity and using the exact curvature to determine surface force on an equidistant Cartesian and a tetrahedral mesh.

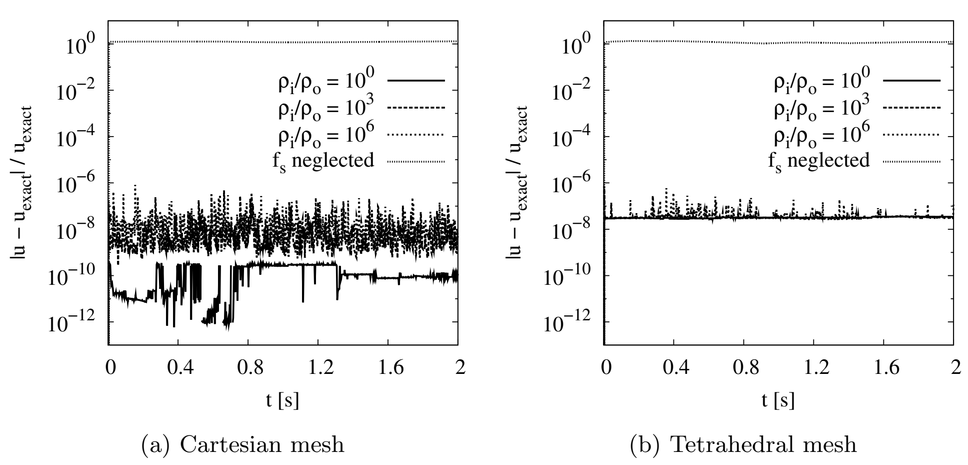


Figure 3.14.: Velocity error for an interface moving at a constant velocity and using the exact curvature to determine surface force on an equidistant Cartesian and a tetrahedral mesh.

depicted in Figures 3.13 and 3.14. This demonstrates the necessity to account for body forces in the pressure-velocity coupling.

3.8.5. Continuity and Mass Conservation

The conservation of mass, defined for incompressible flows by the continuity equation given in Eq. 2.2, is fundamental to fluid flow. According to the implementation of the continuity constraint defined in Eq. 3.25, continuity is implicitly satisfied by the numerical framework within the limits of the numerical solver tolerance and the machine precision.

Figure 3.15 shows the cumulative continuity error of the simulation of the Hagen-Poiseuille flow in a circular pipe presented in Section 3.8.2. For both considered Reynolds numbers the continuity error is negligible. The magnitude of the continuity error of $\lesssim 10^{-8}$ is expected given the applied solver tolerance of $\epsilon_{rel} = 10^{-8}$ in this test case. The cumulative continuity error of the test case presented in Section 3.8.4.2, simulating a surface-tension-dominated interface moving at a constant velocity in a rectangular channel, is shown in Figure 3.16. Similar as for the single-phase flow, the presented results show negligible continuity errors. It should be noted that the mesh type and the applied density ratio do not affect the continuity error. The results prove that the presented numerical framework is satisfying continuity for single-phase flows as well as two-phase flows.

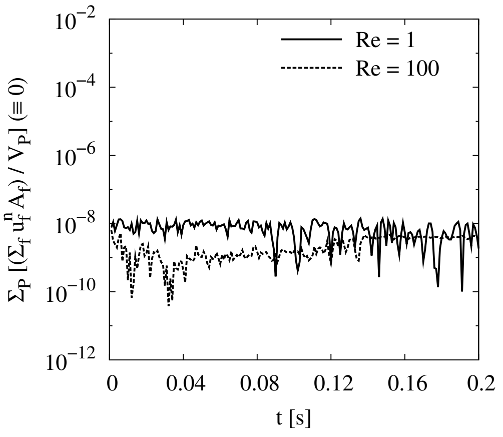
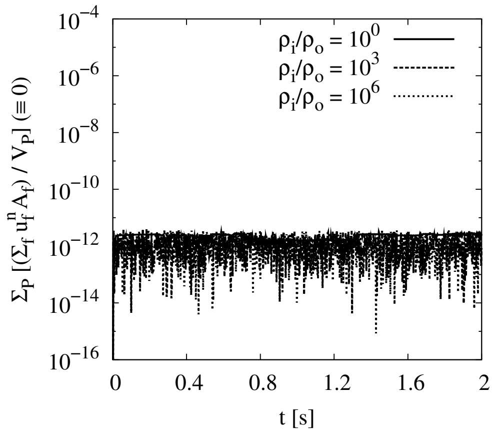
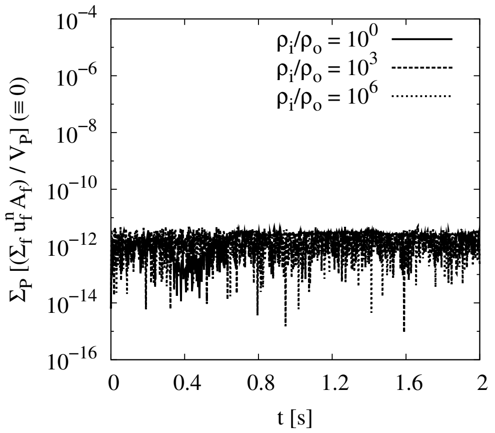


Figure 3.15.: Cumulative continuity error of the Hagen-Poiseuille flow in a circular pipe presented in Section 3.8.2.

The implemented compressive VOF method inherently conserves the colour function and, therefore, by virtue of Eq. 2.44 the mass of each phase within the limit of the solver tolerance applied to solve the VOF equation system (see Section 3.7.2). As observed in Figure 3.17, depicting the evolution of the relative volume error of the volume inside the moving surface-tension-dominated interface presented in Section 3.8.4.2 as represented by the colour function, the conservation error of the implemented compressive VOF method is negligible as the cumulative conservation error after 800 time-steps is $< 10^{-7}\,\%$. The resulting error at every individual time-step is well within the limits of the solver tolerance, given the applied solver tolerance of $\epsilon_{rel} = 10^{-10}$ in this specific case. Only minor differences between Cartesian and tetrahedral meshes are observed.



(a) Cartesian mesh



(b) Tetrahedral mesh

Figure 3.16.: Cumulative continuity error of the moving surface-tension-dominated interface presented in Section 3.8.4.2 on an equidistant Cartesian and a tetrahedral mesh.

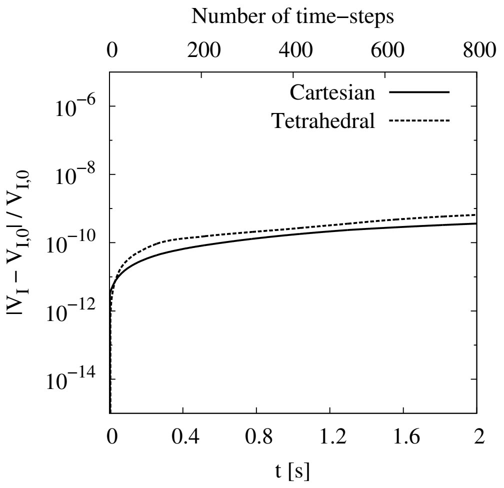


Figure 3.17.: Relative volume error of the volume inside the moving surface-tension-dominated interface presented in Section 3.8.4.2 on an equidistant Cartesian and a tetrahedral mesh.

3.9. Summary

In this chapter a fully-coupled implicit numerical framework has been presented to simulate single-phase flows as well as two-phase flows with surface tension on unstructured meshes. The numerical framework has been developed with emphasis on:

- an accurate flow prediction on unstructured meshes,
- a reliable and strong pressure-velocity coupling,
- a precise balance between pressure gradient and body forces (*i.e.* surface force and gravity), and

- numerical stability on meshes with significant skewness and non-orthogonality.

The presented validation has demonstrated the capabilities of the numerical framework to accurately predict single-phase and two-phase flows. Most crucially, the results obtained on structured and unstructured meshes have been shown to be comparable. Furthermore, the numerical framework provides an accurate balance between pressure gradient and body forces, which is a common source of errors in two-phase flow simulations. The accurate balance between pressure gradient and body forces has been verified for both stationary and moving interfaces, a test case which has not been presented previously in the literature. Contrary to most two-phase flow frameworks reported in the literature which are limited to specific density ratios [120, 217, 227], the presented framework can handle any practical density ratio.

4. Interface Treatment

The numerical treatment of the interface, in particular the thickness of the interface region and the evaluation of the interface curvature, is pivotal for the accurate prediction of two-phase flows. Considerable research efforts have been dedicated to improve the numerical interface treatment in the past two decades but advanced methods are mostly limited to Cartesian meshes. In Section 4.1 the general difficulties associated with the interface representation in VOF frameworks are examined and, in Section 4.2, relevant existing methods to treat the interface are reviewed and discussed. In Section 4.3 a new method to evaluate the interface curvature directly from the colour function field is presented and validated. The new curvature evaluation method is applicable to unstructured meshes and shows results comparable to or better than existing methods, both on Cartesian meshes and unstructured meshes. Subsequently, Section 4.4 examines how surface force and fluid properties should be treated with respect to convolution and Section 4.5 demonstrates how convolution can adversely affect simulation results. The chapter closes with a short summary in Section 4.6.

4.1. Interface Representation in VOF Frameworks

The interface in VOF frameworks is implicitly represented by a colour function value of $0 < \gamma < 1$, as previously explained in Section 2.3.2. Hence, only the cells in which an interface is located are known but not the exact position of this interface. This creates problems for the numerical evaluation of the interface curvature, the application of the surface force and the treatment of fluid properties.

The implicit treatment of the interface by means of the VOF colour function, which is abruptly varying in space, induces significant errors upon differentiation. Derivatives of the colour function are required to determine the surface force (first derivative) and the interface curvature (second derivative). The differentiation of the abruptly varying colour function is ill-posed and results in substantial errors, as a small change in colour function value between neighbouring cells can cause a large change in its derivatives [33, 258, 265]. Cummins *et al.* [34] referred to such errors as *aliasing errors*, due to the similarity with aliasing in digital signal processing and image processing [220, 258]. Since the colour function varies abruptly in space, sudden changes in colour function value between neighbouring cells occur regularly. Contrary to general intuition, the errors associated with differentiating an abruptly varying or noisy function increase as the mesh becomes more refined as a result of the decreasing mesh spacing. Studies of Stickel [220] show that the differentiation of noisy data in general follows the correct trend but that the error magnitude is significant. Brackbill *et al.* [19] observed errors of almost 100% when the interface curvature is calculated directly based on the discontinuous colour function using a standard finite difference approach. The surface force term, which includes the first derivative of the colour function, suffers from the abruptly varying colour function in a similar way as the interface curvature, resulting in an abruptly varying body force in the interface region.

Similar to the regularisation of noisy signals in digital data processing [33, 220], smoothing the colour function field by means of a convolution method is a widely exercised but controversial approach to reduce the adverse effects of the abruptly varying colour function. Calculating the interface curvature from a convoluted representation of the colour function reduces fluctuations of the curvature value along the interface but also omits geometric information of the interface. Similarly, using the first derivative of the convoluted colour function to determine the surface force, convolution smooths the force acting on the fluid but equally spreads the pressure jump at the interface, which is sharp in reality, over a larger region.

Applying a VOF method means that the surface force is stored at cell centres and is, therefore, not a singular force as it is theoretically based on the continuum assumption. The application of the surface force, and with it the thickness of the interface, is a controversial issue. Analogously, the fluid properties, *i.e.* viscosity and density, change at the interface, raising the question whether they should be calculated based on the original (unconvoluted) colour function or the convoluted colour function. Spreading the surface force and the fluid properties over a wider region, for instance by using the convoluted colour function and its gradients, facilitates a smooth transition of momentum across the interface. On the other hand, keeping the fluid properties and the surface force as sharp as possible is desired from a physical viewpoint.

4.2. Review of Existing Methods

This section reviews the available literature on interface treatment in VOF frameworks. Firstly, the convolution of the VOF colour function is introduced and explained. Subsequently, Section 4.2.2 discusses state-of-the-art methods to evaluate the interface curvature in VOF frameworks. Section 4.2.3 summarises the findings concerning an adequate mesh resolution for interfacial flows with respect to VOF methods.

4.2.1. Convolution

Convolution is a mathematical operation which constructs a function by overlapping two other functions. In two-phase flows, convolution[7] is used to transform a discontinuous interface indicator function, such as the VOF colour function, into a more continuous scalar field. The convoluted colour function is frequently applied to reduce the aforementioned aliasing errors occurring when evaluating the interface curvature [19, 34, 46, 71, 238, 262, 263] and to facilitate a smooth momentum transition across the interface [119, 123, 226,

[7] *Smoothing*, *mollification* and *regularisation* are often used synonyms for *convolution* in the relevant literature on two-phase flow modelling.

240, 256, 262, 263, 272]. The quality of the results obtained with convoluted variables depends strongly on the applied convolution length and the convolution method itself. Various convolution methods with application to multiphase flows can be found in the literature, such as polynomial [46, 262, 263, 267], spline [19, 167, 224, 232, 233, 262, 271], Gaussian [79, 157] and trigonometrical convolution functions [179, 232, 233, 240, 271]. Lafaurie *et al.* [123] and Ubbink [238] used an iterative method they referred to as *Laplacian filter*. The implementation of convolution methods on unstructured meshes is typically straightforward.

The convolution of the colour function by means of a Laplacian filter is an iterative process, defined as

$$\gamma_P^{c,i} = \frac{\sum_f \gamma_f^{c,i-1} A_f}{\sum_f A_f} \ , \tag{4.1}$$

where γ^c is the convoluted colour function, subscript f denotes all bounding faces of cell P and superscript i represents the iteration number. The iterative process, with the initial condition $\gamma^{c,i=0} = \gamma$, is performed for a predefined number of iterations. Ubbink [238] recommends two iterations, as the filter then includes the first and second neighbours of each cell.

Using a generic convolution kernel K to convolute the VOF colour function, the convoluted colour function γ^c is defined as

$$\gamma^c(\boldsymbol{x}) = K * \gamma(\boldsymbol{x}) = \int_\Omega \gamma(\boldsymbol{x}')\, K(\boldsymbol{x}' - \boldsymbol{x}) \ d\boldsymbol{x}' \ , \tag{4.2}$$

where Ω denotes the support of the kernel. The support of the kernel is defined as the region, bounded by the finite convolution length ϵ, in which $K \neq 0$. Of course, this procedure is not limited to the colour function but can be applied to any scalar or vector field. Williams *et al.* [263] identified five basic requirements for convolution kernels applied to interfacial flows:

1. the kernel has a finite support Ω,
2. the kernel is monotonically decreasing with increasing distance from the convolution centre,
3. the kernel must be smooth and at least three times continuously differentiable,
4. $\int_\Omega K(\boldsymbol{x}, \epsilon) \ d\boldsymbol{x} = 1$, and
5. the kernel collapses to a Dirac Delta function for $\Omega \rightarrow 0$.

Applying non-monotonic convolution kernels (see condition 2), such as the Nordmark kernel [167], to the colour function results in an oscillating and potentially unbounded colour function field. The convolution kernel should be smooth (condition 3) and should have continuous partial derivatives of at least third-order, since curvature is proportional to the second derivative of the colour function.

Tornberg and co-workers [233, 271] applied a piecewise linear convolution kernel, which in its radially symmetric form is defined as

$$K_{lin}(r,\epsilon) = \begin{cases} A\,(1 - r/\epsilon) & \text{if } r < \epsilon \\ 0 & \text{if } r \geq \epsilon \ , \end{cases} \tag{4.3}$$

where r is the distance to the centre of the kernel and A is a constant chosen to normalise the kernel. The linear kernel K_{lin} fulfils all the requirements mentioned above apart from condition 3, as it is not continuously differentiable.

Another class of convolution kernels frequently applied in interfacial flow simulations are spline kernels. Williams [262] applied smooth, monotonic kernels such as the K_3 kernel, which had been previously applied in smooth particle hydrodynamics (SPH) simulations [158, 160] and is defined as

$$K_3(r,\epsilon) = \begin{cases} A\left(1 - 6\frac{r^2}{\epsilon^2} + 6\frac{r^3}{\epsilon^3}\right) & \text{if } r \leq \epsilon/2 \\ A\left(1 - \frac{r}{\epsilon}\right)^3 & \text{if } \epsilon/2 < r \leq \epsilon \\ 0 & \text{if } r \geq \epsilon \ . \end{cases} \tag{4.4}$$

This kernel has been applied in VOF simulations by Lörstad and co-workers [137, 138] as well. As Morris *et al.* [160] pointed out, the second derivative of the K_3 kernel (and of similarly constructed spline kernels) is not continuous but piecewise linear and, therefore, does not fulfil condition 3 of the requirements discussed above.

Two radially symmetric kernels satisfying all of the above requirements are the sixth-order and eighth-order kernels of Williams [262]. The sixth-order kernel is given as

$$K_6(r,\epsilon) = \begin{cases} A(\epsilon^2 - r^2)^3 & \text{if } r < \epsilon \\ 0 & \text{if } r \geq \epsilon \end{cases} \tag{4.5}$$

and, similarly, the eighth-order kernel follows as

$$K_8(r,\epsilon) = \begin{cases} A(\epsilon^2 - r^2)^4 & \text{if } r < \epsilon \\ 0 & \text{if } r \geq \epsilon \ . \end{cases} \tag{4.6}$$

The K_6 and K_8 kernels are among the most widely used convolution kernels for VOF simulations (see *e.g.* [34, 71, 154, 263]).

Peskin [179] introduced a convolution method using a cosine function to define the weighting of the cells within convolution length ϵ. A separate cosine function for each Cartesian direction i is solved, with the convoluted colour function being calculated as [179, 233, 240]

$$\gamma^c(\boldsymbol{x}) = \int_\Omega \gamma(\boldsymbol{x}') \prod_i \left[K_{cos}(x_i' - x_i)\right] \, d\boldsymbol{x}' \ . \tag{4.7}$$

The convolution kernel K_{cos}, which fulfils all five requirements outlined above, is defined as

$$K_{cos}(x_i,\epsilon) = \begin{cases} A/2\epsilon \ \left(1 + cos\left(\pi x_i/\epsilon\right)\right) & \text{if } x_i < \epsilon \\ 0 & \text{if } x_i \geq \epsilon \ . \end{cases} \tag{4.8}$$

Williams [262] performed an extensive study on convolution kernels, analysing different high-order polynomial kernels as well as spline kernels. Williams comprehensively demonstrated that the sixth-order and eighth-order kernels yield better results than spline kernels, such as the K_3 kernel defined in Eq. 4.4. The sixth-order and eighth-order kernels of Williams [262] are depicted in Figure 4.1 alongside the cosine kernel of Peskin [179] as a function of normalised distance from the convolution centre. The K_6 and K_8 kernels give more weight to values located at discrete points (*i.e.* cell centres) close to the centre of the convolution than the K_{cos} kernel. Williams [262] also pointed out that the magnitude of the kernel derivatives become very large for decreasing convolution length ϵ, as they become singular for $\epsilon \to 0$. Thus, an increasing number of discrete points is required for decreasing convolution length ϵ to numerically represent the derivatives of the convolution kernel or of a convoluted variable. This becomes important when surface force or interface curvature are evaluated from the convoluted colour function field. Comparing the three convolution kernels depicted in Figure 4.1, the eighth-order kernel has the steepest slope (first derivative) and the highest curvature (second derivative) of the three kernels and, therefore, requires the highest number of discrete points for an adequate resolution. The K_{cos} kernel provides a smoother transition for small convolution lengths, *i.e.* $\epsilon < 3\Delta x$, whereas the K_6 and the K_8 kernel are predicated for larger convolution lengths ($\epsilon \geq 3\Delta x$). Williams [262, p. 42] explicitly recommends a convolution length of $\epsilon \geq 4\Delta x$ for the K_6 kernel.

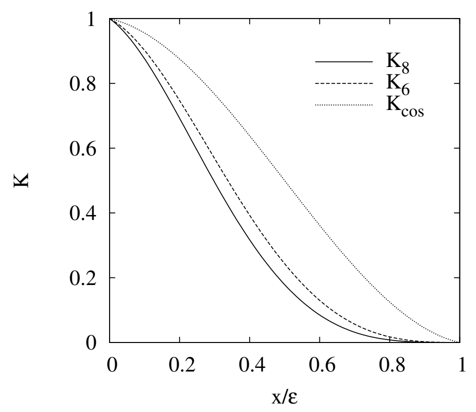


Figure 4.1.: The K_8, K_6 and K_{cos} convolution kernels as a function of the distance x from the convolution centre relative to convolution length ϵ.

4.2.2. Interface Curvature

As previously explained in Section 2.3.3, the force acting at the interface due to surface tension depends directly on the local interface curvature. The accurate numerical evaluation of the interface curvature in VOF frameworks is a particularly challenging task, because:

- the colour function is not a smooth function,

- the explicit position of the interface is not know but only the cells in which an interface is located, and
- the interface curvature evaluation depends strongly on the mesh resolution.

A number of methods to evaluate the interface curvature directly from the colour function or indirectly by some sort of reconstructed function have been presented in the literature. The following sections review relevant methods to calculate the interface curvature in VOF frameworks and highlight the characteristics of each method.

4.2.2.1. Direct Differentiation Methods

Direct differentiation methods refers to methods which evaluate the curvature by differentiating the colour function field directly. The colour function field is differentiated twice by a suitable differentiation method to evaluate the local interface curvature. Various finite difference, finite volume and least-squares methods for the differentiation of the colour function can be found in the literature. The extension to unstructured meshes is typically straightforward and has been demonstrated in previous studies [100, 238, 262]. Regardless which method is used for differentiation, the curvature is defined following Eq. 2.49 as [19]

$$\kappa = -\left(\nabla \cdot \frac{\nabla\gamma}{|\nabla\gamma|}\right) . \tag{4.9}$$

Standard finite difference and finite volume methods are among the most intuitive and widely used methods to calculate the interface curvature. Finite difference and finite volume methods do not require significant computational resources but suffer from inaccurate curvature estimates, because of the discontinuous nature of the colour function. Brackbill *et al.* [19] applied a standard finite difference method to differentiate the colour function and proposed to reformulate Eq. 4.9 as

$$\kappa = \frac{1}{\nabla\gamma}\left[\left(\frac{\nabla\gamma}{|\nabla\gamma|}\cdot\nabla\right)|\nabla\gamma| - (\nabla\cdot\nabla\gamma)\right] , \tag{4.10}$$

to shift the major contribution of the curvature from the edges of the differentiation stencil to its centre. Using a finite volume framework, the interface curvature can be evaluated straightforward using the Gauss theorem [119, 120, 238]

$$\kappa_P = -\frac{1}{V_P}\int_{V_P} \nabla\cdot\boldsymbol{m}\, dV \approx -\frac{1}{V_P}\sum_f \boldsymbol{m}_f\, A_f = -\frac{1}{V_P}\sum_f \left.\frac{\nabla\gamma}{|\nabla\gamma|}\right|_f A_f , \tag{4.11}$$

where γ may either be the convoluted or unconvoluted colour function.

In order to diminish the aforementioned aliasing errors and provide more accurate curvature estimates, the colour function is often smoothed by means of a convolution method for the purpose of curvature evaluation [4, 19, 34, 46, 71, 123, 154, 238, 262, 263]. Williams *et al.* [263] introduced a hybrid approach, using the first derivative of a convolution kernel to evaluate the interface normal vector

$$\boldsymbol{m} \approx \frac{\nabla K * \gamma}{|\nabla K * \gamma|} \tag{4.12}$$

and a finite difference method to calculate the curvature from the interface normal vector, following on a two-dimensional equidistant Cartesian mesh as

$$\kappa = -\nabla \cdot \boldsymbol{m} \approx -\frac{m_x^{i+1,j} - m_x^{i-1,j}}{2\Delta x} - \frac{m_y^{i,j+1} - m_y^{i,j-1}}{2\Delta y} \; . \tag{4.13}$$

The extension to three dimensions is straightforward. Aleinov and Puckett [4] used the second derivative of a convolution kernel to directly determine the curvature. However, this approach was effectively dismissed by Williams [262] as he demonstrated that the interface would need to be convoluted over at least 64 cells to obtain adequately resolved second derivatives of the convolution function, which severely violates the local character of the interface and is computationally very expensive.

Studies published by Cummins *et al.* [34] and Raessi *et al.* [191] found that curvature values calculated by finite difference and finite volume methods do not converge with mesh refinement, representing a severe drawback of such methods. The reason for the observed divergence are aliasing errors resulting from the differentiation of the colour function [34].

Kothe *et al.* [120], van Wachem and Schouten [247] and Wang [253] applied a least-squares fit of a Taylor series to calculate the first derivative of the colour function and to evaluate the interface normal vector. Taylor series expansions are formed from cell P to all adjacent cells Q, defined as

$$\gamma_Q = \gamma_P + \left.\frac{\partial \gamma}{\partial x}\right|_P (x_Q - x_P) + \left.\frac{\partial \gamma}{\partial y}\right|_P (y_Q - y_P) + \left.\frac{\partial \gamma}{\partial z}\right|_P (z_Q - z_P) \; . \tag{4.14}$$

The resulting equations are then solved in a 3×3 equation system

$$\boldsymbol{A} \cdot \nabla\gamma|_P = \boldsymbol{b} \tag{4.15}$$

using a least-squares minimisation. The matrix $\boldsymbol{A}$ and the right-hand side vector $\boldsymbol{b}$ follow from the Taylor series expansion as

$$\boldsymbol{A} = \begin{pmatrix} \sum_Q \frac{(x_Q-x_P)^2}{\Delta s_Q} & \sum_Q \frac{(x_Q-x_P)(y_Q-y_P)}{\Delta s_Q} & \sum_Q \frac{(x_Q-x_P)(z_Q-z_P)}{\Delta s_Q} \\ \sum_Q \frac{(x_Q-x_P)(y_Q-y_P)}{\Delta s_Q} & \sum_Q \frac{(y_Q-y_P)^2}{\Delta s_Q} & \sum_Q \frac{(y_Q-y_P)(z_Q-z_P)}{\Delta s_Q} \\ \sum_Q \frac{(x_Q-x_P)(z_Q-z_P)}{\Delta s_Q} & \sum_Q \frac{(y_Q-y_P)(z_Q-z_P)}{\Delta s_Q} & \sum_Q \frac{(z_Q-z_P)^2}{\Delta s_Q} \end{pmatrix} \tag{4.16}$$

and

$$\boldsymbol{b} = \begin{pmatrix} \sum_Q \frac{(\gamma_Q-\gamma_P)(x_Q-x_P)}{\Delta s_Q} \\ \sum_Q \frac{(\gamma_Q-\gamma_P)(y_Q-y_P)}{\Delta s_Q} \\ \sum_Q \frac{(\gamma_Q-\gamma_P)(z_Q-z_P)}{\Delta s_Q} \end{pmatrix} , \tag{4.17}$$

where Δs_Q is the distance of cell P to its neighbour Q. The curvature is, subsequently, calculated using a finite difference or finite volume method. The least-squares fit provides an inherent spatial smoothing of the abruptly varying colour function field for the purpose of calculating its derivatives [253].

4.2.2.2. Height Function Techniques

Recent efforts to find more accurate ways to determine the interface curvature have focused predominantly on height function (HF) techniques [1, 18, 34, 66, 71, 72, 91, 129, 135, 183]. HF techniques construct fluid heights as a basis for the curvature evaluation by integrating the colour function along the largest interface normal vector component. The curvature is then calculated from the derivatives of the fluid heights. For instance, if the z-component is the largest component of the colour function gradient, the curvature is calculated as [71, 135]

$$\kappa = \frac{h_{xx} + h_{yy} + h_{xx} h_y^2 + h_{yy} h_x^2 - 2 h_{xy} h_x h_y}{\left(1 + h_x^2 + h_y^2\right)^{3/2}} \ , \tag{4.18}$$

where the derivatives of the height function, h_i and h_{ij}, are calculated using a finite difference approach with central differencing. Studies carried out by Cummins *et al.* [34], Francois *et al.* [71], López *et al.* [135] and Popinet [183] presented excellent results using HF techniques compared to other curvature evaluation methods and demonstrated second-order convergence on equidistant Cartesian meshes. Liovic *et al.* [129] successfully reduced the orientation dependency of the original HF technique on orthogonal meshes, further increasing the accuracy of curvature evaluation with height functions, by not only including fluid heights generated along the coordinate axes but also including fluid heights defined on the diagonals of the Cartesian mesh.

Lörstad, Fuchs and co-workers [137, 138] proposed a curvature evaluation method founded on the principles of the height function technique, called *Direction Averaged Curvature* (DAC) model. The DAC model computes the curvature from fluid heights in the direction of the largest interface normal vector component. Assuming the z-component is the largest component of the interface normal vector $\boldsymbol{m}$, the interface curvature is defined as [138]

$$\kappa = \frac{m_z}{|m_z|} \left(\frac{h_{ii}}{|\boldsymbol{m}|} - \frac{h_i h_j h_{ij}}{|\boldsymbol{m}|^3} \right) \ . \tag{4.19}$$

The major drawback of HF techniques are inconsistent curvature estimates if the interface is poorly resolved [34, 183], *i.e.* when the curvature radius approaches the mesh size. This is a serious drawback, as the error in curvature is largest in these areas. Furthermore, no extension of HF techniques to unstructured meshes has yet been presented.

4.2.2.3. Other Methods

The *advected normal vectors* method developed by Raessi *et al.* [191, 193, 194] advects the normal vector of the interface as a separate variable together with the colour function. At every time instant the interface curvature is calculated by Eq. 4.11 from the advected interface normal vector using central differencing [188]. Based on the underlying flow field, a transport equation for the interface normal vector

$$\frac{\partial \boldsymbol{m}}{\partial t} + \nabla(\boldsymbol{u} \cdot \boldsymbol{m}) = 0 \tag{4.20}$$

is solved. A specifically designed algorithm, see [191] for details, ensures that the magnitude of the interface normal vectors is unity and that the interface normal vectors and the colour function remain coupled. The key advantage of this method is the decreasing curvature error with increasing mesh resolution, contrary to standard finite difference and finite volume methods. However, the interface normal vectors become inaccurate in cases where the interface is locally underresolved, which also affects the accuracy at succeeding time instants due to the advection of the ill-defined normals.

The coupling of VOF methods and level-set methods, often called VOF-LS or CLSVOF methods, has experienced considerable attention in recent years. The basic idea behind combining VOF and LS is to exploit the advantages and mask the disadvantages of the two approaches. A VOF method is used to ensure mass conservation and a level-set method is used to compute accurate interface normals and interface curvature, since the level-set function is smooth and continuous, using standard finite difference or finite volume methods. The LS distance function is either reconstructed based on the advected VOF colour function [3, 171, 222, 257] or the LS distance function is advected separately and coupled with the VOF method subsequently [76, 219, 223, 227, 242, 256]. Results presented in a number of studies, *e.g.* in [141, 222, 223], show a reduction of the error in curvature using VOF-LS methods compared to traditional VOF methods and improved mass conservation properties compared to standard LS methods.

Other methods to calculate the interface curvature in interface capturing frameworks, but not widely used and, therefore, not considered to be relevant in the context of this thesis, are the curvature evaluation from a fitted high-order polynomial [182] and the curvature evaluation based on a parametrisation method [203].

4.2.3. Mesh Resolution

An adequate mesh resolution at the interface, with respect to interface curvature and surface force, is critical for the success of two-phase flow simulations. For two fluids with considerable surface tension, the interface is usually the dominating flow feature and, thus, governs the mesh resolution. The resolution of the interface is particularly important because the spatial accuracy of the interface position is limited to first order [20, 61, 227].

The mesh resolution tangential to the interface is critical for the accurate evaluation of the local interface curvature, *e.g.* a stronger curved interface requires a higher resolution. An insufficiently resolved interface curvature typically results in an underestimated local curvature magnitude and, thus, leads to an underprediction of the local surface force. It is obvious that from a geometrical standpoint a higher resolution is always preferable as it reduces the numerical discretisation error of the interface shape representation. However, as explained in Section 4.1, a higher mesh resolution increases errors arising from differentiating the abruptly varying colour function if direct differentiation methods are applied. Various recommendations regarding the mesh resolution with respect to interface curvature can be found in the literature. According to Brackbill *et al.* [19], the curvature varies quasi-continuously along the interface contours if $\kappa \cdot \Delta x < 1$. Raessi *et al.* [194] deem an interface to be underresolved if $\kappa \cdot \Delta x \geq 1/3$ and results by Malik *et al.* [143] indicate an effectively curvature independent solution error for $\kappa \cdot \Delta x \leq 0.05$. Popinet [183] and Cummins *et al.* [34] found height function techniques struggle to produce consistent curvature estimates if $\kappa \cdot \Delta x > 0.25$ or $\kappa \cdot \Delta x > 0.2$, respectively. A popular mesh resolution choice found in the literature for equidistant Cartesian meshes is 20 mesh cells per diameter for circular and spherical interfaces [34, 71, 92, 129, 262, 263], which corresponds to $\kappa \cdot \Delta x = 0.2$.

The mesh resolution normal to the interface is equally important as it determines the thickness of the interface region. From a physical point a view, minimising the interface thickness is, of course, highly desirably. However, minimising the interface thickness emphasises the discontinuity at the interface, which potentially poses problems in terms of numerical stability and momentum error at the interface for high density and viscosity ratios. Studies by Francois *et al.* [71] and Zahedi *et al.* [272] indicate that minimising the interface thickness is beneficial with respect to surface force and the resulting pressure jump across the interface. On the other hand, spreading the fluid properties over several mesh cells provides a continuous momentum transfer between the fluids [98, 256].

The convolution of the colour function is strongly affected by the mesh resolution and two cases should be distinguished:

1. the convolution length ϵ is constant, meaning that the mesh resolution affects the number of computational points within the support of the convolution kernel, and

2. the convolution length ϵ is proportional to the mesh distance Δx, meaning that the mesh resolution affects the convolution length.

As Williams [262] demonstrated, a higher resolution of the convolution kernel, and equally of the interface region, reduces curvature errors significantly. Studies by Francois *et al.* [71] and Cummins *et al.* [34] found an improved convergence behaviour for curvature estimates calculated directly from the colour function field if the convolution length is constant or $\epsilon \propto \sqrt{\Delta x}$, which supports the findings of Williams [262]. However, including more neighbour cells in the convolution support, *i.e.* increasing the computational stencil, increases the required computational effort considerably, as the number of included neighbour cells, and with it the required computational time, increases with power three for three-dimensional simulations.

4.3. The CELESTE Method for Interface Curvature Evaluation

The content of this section has in parts been published in:

[53] Denner, F. and van Wachem, B.G.M.: Two-Phase Flow Modelling on Arbitrary Meshes: Superior VOF Curvature Estimation and the Issue of Convolution. International Conference on Numerical Methods in Multiphase Flows, 12 - 14 June 2012, State College, PA, USA.

[55] Denner, F. and van Wachem, B.G.M.: Fully-coupled balanced-force VOF framework for arbitrary meshes with least-squares curvature evaluation from volume fraction. Numerical Heat Transfer, Part B: Fundamentals, accepted for publication. DOI: 10.1080/10407790.2013.849996

A new method to evaluate the interface normal vector and the interface curvature, called *CELESTE* (**C**urvature **E**valuation with **LE**ast-**S**quares fit of **T**aylor **E**xpansion), is presented in this section. As the literature survey of methods to evaluate curvature in Section 4.2.2 indicates, the curvature evaluation on unstructured meshes is essentially limited at present to direct differentiation methods, which are straightforward to implement but produce curvature estimates of insufficient quality. The new curvature evaluation method is developed based on the following requirements:

1. improved curvature estimates compared to standard finite difference and finite volume methods,
2. applicable to arbitrary meshes,
3. diminishing aliasing errors upon differentiation,
4. straightforward extension for parallel computer systems, and
5. variable stencil size.

The CELESTE method is founded on a least-squares fit of a Taylor series and is applicable to arbitrary meshes. In contrast to the least-squares method reported by Kothe *et al.* [120] discussed in Section 4.2.2.1, CELESTE is constructed around an overdefined system of equations and utilises a least-squares fit not only for the evaluation of the interface normal vector but also for the evaluation of the interface curvature. The CELESTE method is constituted by three distinct steps. Firstly, the first derivative of the colour function field is calculated using a least-squares fit of a second-order Taylor series expansion, based on a discrete number of neighbouring cells. The interface normal vector is obtained by normalising the resulting first derivative of the colour function. Subsequently, the interface curvature is evaluated with a similar least-squares fit using the interface normal vector. Thirdly, a weighted local average of the curvature is calculated in order to obtain a representative curvature in the entire interface region. CELESTE diminishes numerical noise as a result of the least-squares fit, is applicable to arbitrary meshes, can be implemented in parallelised software frameworks without major changes to the existing data structure (see Appendix A.2 for details) and various stencil sizes can be applied to evaluate the interface curvature.

4.3.1. Least-Squares Approach

The proposed method is based on a second-order Taylor series expansion of the colour function γ from cell P to its neighbour cells Q. In three dimensions the Taylor series expansion is defined as

$$
\begin{aligned}
\gamma_Q &= \gamma_P + \sum_{i=1}^{3} \underline{\left.\frac{\partial \gamma}{\partial x_i}\right|_P} (x_{i,Q} - x_{i,P}) + \sum_{i=1}^{3}\sum_{j=1}^{3} \underline{\left.\frac{\partial^2 \gamma}{\partial x_i\, \partial x_j}\right|_P} \frac{(x_{i,Q} - x_{i,P})\,(x_{j,Q} - x_{j,P})}{2} \\
&+ \mathcal{O}(\Delta x^3, \Delta y^3, \Delta z^3) \ . \qquad (4.21)
\end{aligned}
$$

There are 9 unknowns, underlined in Eq. 4.21, so ideally 9 points around the mesh cell P should be used. However, it is important to have a symmetric stencil of points around cell P, as explained in Section 4.3.4. In practice, matching both conditions is not possible and an overdefined equation system is obtained, defined as

$$
\boldsymbol{A} \cdot \boldsymbol{\phi} = \boldsymbol{b} \ , \qquad (4.22)
$$

which is solved using a least-squares algorithm. The coefficients of the unknown derivatives, *e.g.* $x_Q - x_P$, are placed in matrix $\boldsymbol{A}$ and the known values at cell P and its neighbours Q are placed in the right-hand side vector $\boldsymbol{b}$. In the presented study the least-squares fit is performed using the *dgels* routine of the freely-available software package LAPACK [7].

4.3.2. Interface Normal Vector

The computation of the interface normal vector follows the least-squares approach outlined in the previous section. Firstly, the gradient of the colour function field is determined from which, secondly, the interface normal vector is readily available by normalising the colour function gradient. The coefficient matrix $\boldsymbol{A}$ and the right-hand side vector $\boldsymbol{b}$ are weighted by a geometric weighting factor Δs_Q^{-2}, which is the squared inverse distance from neighbour point Q to centre point P. Hence, the the equation system given in Eq. 4.22 for the colour function gradient is constituted by the coefficient matrix

$$
\boldsymbol{A} = \begin{pmatrix}
\frac{\Delta x_1}{\Delta s_1^2} & \frac{\Delta y_1}{\Delta s_1^2} & \frac{\Delta z_1}{\Delta s_1^2} & \frac{\Delta x_1^2}{2\Delta s_1^2} & \frac{\Delta y_1^2}{2\Delta s_1^2} & \frac{\Delta z_1^2}{2\Delta s_1^2} & \frac{\Delta x_1 \Delta y_1}{\Delta s_1^2} & \frac{\Delta x_1 \Delta z_1}{\Delta s_1^2} & \frac{\Delta y_1 \Delta z_1}{\Delta s_1^2} \\
& \cdots & & & \cdots & & & \cdots & \\
\frac{\Delta x_Q}{\Delta s_Q^2} & \frac{\Delta y_Q}{\Delta s_Q^2} & \frac{\Delta z_Q}{\Delta s_Q^2} & \frac{\Delta x_Q^2}{2\Delta s_Q^2} & \frac{\Delta y_Q^2}{2\Delta s_Q^2} & \frac{\Delta z_Q^2}{2\Delta s_Q^2} & \frac{\Delta x_Q \Delta y_Q}{\Delta s_Q^2} & \frac{\Delta x_Q \Delta z_Q}{\Delta s_Q^2} & \frac{\Delta y_Q \Delta z_Q}{\Delta s_Q^2} \\
& \cdots & & & \cdots & & & \cdots & \\
\frac{\Delta x_{N_Q}}{\Delta s_{N_Q}^2} & \frac{\Delta y_{N_Q}}{\Delta s_{N_Q}^2} & \frac{\Delta z_{N_Q}}{\Delta s_{N_Q}^2} & \frac{\Delta x_{N_Q}^2}{2\Delta s_{N_Q}^2} & \frac{\Delta y_{N_Q}^2}{2\Delta s_{N_Q}^2} & \frac{\Delta z_{N_Q}^2}{2\Delta s_{N_Q}^2} & \frac{\Delta x_{N_Q} \Delta y_{N_Q}}{\Delta s_{N_Q}^2} & \frac{\Delta x_{N_Q} \Delta z_{N_Q}}{\Delta s_{N_Q}^2} & \frac{\Delta y_{N_Q} \Delta z_{N_Q}}{\Delta s_{N_Q}^2}
\end{pmatrix} , \qquad (4.23)
$$

the solution vector

$$
\boldsymbol{\phi} = \left(\left.\frac{\partial \gamma}{\partial x}\right|_P \ \left.\frac{\partial \gamma}{\partial y}\right|_P \ \left.\frac{\partial \gamma}{\partial z}\right|_P \ \left.\frac{\partial^2 \gamma}{\partial x \partial x}\right|_P \ \left.\frac{\partial^2 \gamma}{\partial y \partial y}\right|_P \ \left.\frac{\partial^2 \gamma}{\partial z \partial z}\right|_P \ \left.\frac{\partial^2 \gamma}{\partial x \partial y}\right|_P \ \left.\frac{\partial^2 \gamma}{\partial x \partial z}\right|_P \ \left.\frac{\partial^2 \gamma}{\partial y \partial z}\right|_P \right)^T , \qquad (4.24)
$$

and the right-hand side vector

$$\boldsymbol{b} = \begin{pmatrix} \frac{\Delta\gamma_1}{\Delta s_1^2} \\ \ldots \\ \frac{\Delta\gamma_Q}{\Delta s_Q^2} \\ \ldots \\ \frac{\Delta\gamma_{N_Q}}{\Delta s_{N_Q}^2} \end{pmatrix} \tag{4.25}$$

where N_Q is the number of neighbours evaluated for cell P. The coefficients $\Delta x_Q = x_Q - x_P$, $\Delta y_Q = y_Q - y_P$ and $\Delta z_Q = z_Q - z_P$ in matrix $\boldsymbol{A}$ represent the distance from neighbour Q to cell P and similarly $\Delta\gamma_Q = \gamma_Q - \gamma_P$ in the right-hand side vector of Eq. 4.24. The interface normal vector $\boldsymbol{m}$ is obtained by normalising the first derivatives of the colour function, following Eq. 2.50. The inclusion of the second derivatives of the colour function in the linear equation system described above, although not used directly, is essential for the accuracy of the subsequent curvature evaluation, as the interface curvature follows from the second derivatives of the colour function.

4.3.3. Interface Curvature

In principle, the calculation of the interface curvature follows the same approach as the calculation of the colour function gradient presented in the previous section. However, only the first derivatives are taken into account for the curvature, as higher-order derivatives do not improve the curvature evaluation notably. For each component k of interface normal vector $\boldsymbol{m}$ the equation system

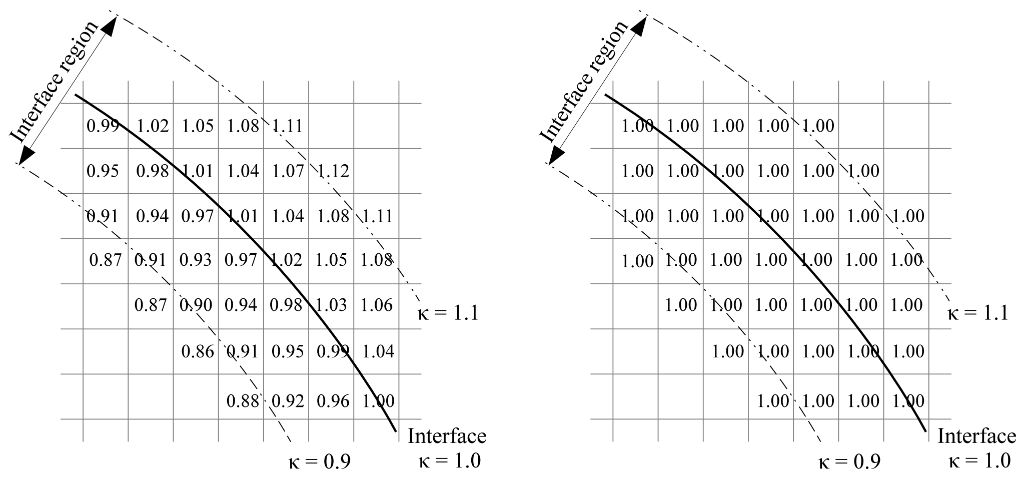
$$\begin{pmatrix} \Delta x_1 & \Delta y_1 & \Delta z_1 \\ & \ldots & \\ \Delta x_Q & \Delta y_Q & \Delta z_Q \\ & \ldots & \\ \Delta x_{N_Q} & \Delta y_{N_Q} & \Delta z_{N_Q} \end{pmatrix} \cdot \begin{pmatrix} \left.\frac{\partial m_k}{\partial x}\right|_P \\ \left.\frac{\partial m_k}{\partial y}\right|_P \\ \left.\frac{\partial m_k}{\partial z}\right|_P \end{pmatrix} = \begin{pmatrix} \Delta m_{k,1} \\ \ldots \\ \Delta m_{k,Q} \\ \ldots \\ \Delta m_{k,N_Q} \end{pmatrix} \tag{4.26}$$

is solved using a least-squares fit. From the resulting gradients of the interface normal vector, the curvature is readily available for each cell P as

$$\kappa_P = - \left.\frac{\partial m_i}{\partial x_i}\right|_P . \tag{4.27}$$

The numerical representation of the interface as a region of finite thickness presents a problem for the application of the computed interface curvature to calculate the surface force, schematically illustrated in Figure 4.2. Even a geometrically accurate curvature estimate of Eq. 4.27 as depicted in Figure 4.2a can result in considerable numerical errors, since the curvature varies in the direction normal to the interface. For the surface force, however, a constant interface curvature in the direction normal to the interface is desirable, as illustrated by the example in Figure 4.2b, because the surface force should not vary in the direction normal to the interface, to resemble reality as closely as possible.

In order to get a representative curvature value in the entire interface region, a two step



(a) Geometrically accurate curvature values at cell centres within the interface region

(b) Numerically desired curvature values at cell centres within the interface region

Figure 4.2.: Schematical illustration of interface curvature κ at cell centres located in the interface region. The example interface (thick black line) has a constant curvature of $\kappa = 1.0\,m^{-1}$ with a numerical interface region of finite thickness (bounded by the dash-dotted line) and is illustrated on an equidistant quadrilateral mesh. In this example, the theoretical curvature of the inner bound of the interface region is $\kappa = 0.9\,m^{-1}$ and $\kappa = 1.1\,m^{-1}$ for the outer bound of the interface region, as illustrated in both figures. All given curvature values have the unit m^{-1}.

weighted average of the curvature is calculated. The first step, which aims on smoothing the curvature field, calculates an intermediate curvature κ^* in each cell by weighting the curvature value with the related colour function value, defined as

$$\kappa_P^* = \frac{\kappa_P w_{\gamma,P} + \sum_Q \kappa_Q w_{\gamma,Q}}{w_{\gamma,P} + \sum_Q w_{\gamma,Q}} \ , \tag{4.28}$$

where subscript Q denotes the neighbour cells of cell P. Because the calculated interface normal vectors, and as a result the interface curvature, become inaccurate as the colour function reaches zero or unity [34, 197], the highest weight ($w_\gamma = 1$) is assigned to cells with a colour function value of $\gamma = 0.5$. Cells which do not contain an interface, *i.e.* $\gamma = 0$ or $\gamma = 1$, are omitted. Accordingly, the weighting factor w_γ follows as

$$w_{\gamma,i} = \left[1 - 2\left(|0.5 - \gamma_i|\right)\right]^8 \ . \tag{4.29}$$

To reduce the curvature variation normal to the interface, the second step of the averaging procedure weights the curvature in neighbouring cells additionally with respect to the interface normal vector and is given as

$$\overline{\kappa}_P = \frac{\kappa_P^* w_{\gamma,P} + \sum_Q \kappa_Q^* w_{\gamma,Q} w_{m,Q}}{w_{\gamma,P} + \sum_Q w_{\gamma,Q} w_{m,Q}} \ . \tag{4.30}$$

The weighting factor w_m is defined as

$$w_{m,Q} = |\boldsymbol{m}_P \cdot \boldsymbol{s}_Q|^8 \ , \tag{4.31}$$

where $\boldsymbol{m}_P$ represents the interface normal vector at cell P and $\boldsymbol{s}_Q$ is the normalised vector connecting cell P and its neighbour Q. Numerical experiments proved 8 to be a desirable choice for the exponents in Eqs. 4.29 and 4.31, although other exponents could be used as well.

4.3.4. Computational Stencil

The computational stencil on which the presented procedure to evaluate the interface curvature is applied affects the resulting interface curvature estimates significantly. Haselbacher and Vasilyev [89] identified two contradicting criteria for the stencil of a gradient reconstruction using a least-squares fit of a Taylor series: a) the stencil should be as small as possible to minimise the truncation error constants and b) the stencil should be as symmetric as possible to cancel out terms in the truncation error. Kothe *et al.* [120] concluded that a wide, symmetric stencil for the evaluation of the interface normal vector is required, *e.g.* 27 cells in three-dimensional simulations. Also, considering the minimisation of numerical noise arising from the abruptly varying colour function upon differentiation, the number of cells included in the stencil must be larger than the number of unknown gradients.

In order to make use of a symmetric stencil for the least-squares fit, a cubical computational stencil with edge length l_s is used for CELESTE. Thus, every cell within a distance of $l_s/2$ to centre cell P with respect to the three Cartesian coordinate axes is part of the computational stencil. Figure 4.3 illustrates a cubical stencil in two dimensions on a Cartesian mesh and a triangular mesh. Special caution must be exercised for problems where high interface curvature values compared to the mesh resolution occur, as large stencils can adversely affect the results. With respect to the stencil size used for CELESTE, Lisita [130] found an edge length for the interface normal evaluation of $l_{s,m} = 4\Delta x$ and for the interface curvature evaluation of $l_{s,\kappa} = 2\Delta x$ to be preferable choices.

On unstructured meshes the mesh spacing is neither constant nor readily available. Therefore, a reference mesh distance d^* is defined as a basis for the stencil size on unstructured meshes. This reference mesh distance d^* is defined as the average distance between two adjacent cell centres in proximity of the interface. In the specific case of an equidistant hexahedral mesh, d^* is equal to the mesh spacing Δx.

4.3.5. Validation

To validate the CELESTE method, a static inviscid drop in equilibrium is simulated to quantify the errors resulting from the evaluation procedure of the interface curvature. A static inviscid fluid particle (drop or bubble) in equilibrium is an often facilitated test case, as for instance in [71, 129, 134, 135, 263], due to its simplicity and informative value. Firstly, because the drop is in equilibrium, the interface is spherical by definition, allowing

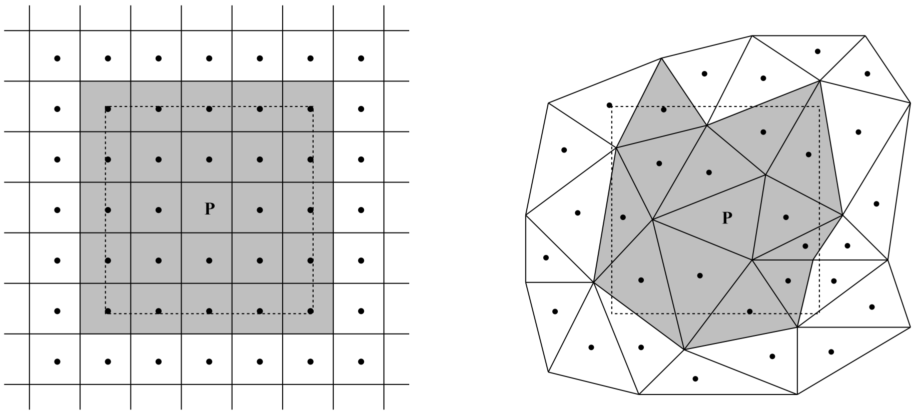


Figure 4.3.: Cubical stencil (dashed box) of edge length l_s on a Cartesian (left) and a triangular (right) mesh. The centre of the stencil cell is denoted with P and cell centroids of neighbouring cells are illustrated with dots. Highlighted cells are part of the CELESTE stencil.

to compare the calculated curvature value at every discrete point in the interface region against the exact curvature value given by Eq. 3.144. Secondly, since the velocity field is initially stationary and given a balanced-force numerical framework such as the one described in Chapter 3, all observed velocities are parasitic currents and can be attributed to an inaccurate curvature evaluation. Lastly, considering an inviscid spherical fluid particle and no additional external body forces such as gravity, the pressure jump across the interface is exactly defined by Eq. 2.46. It is important that curvature error, parasitic currents and pressure error are evaluated separately, as they do not correlate with each other, even though the inaccuracy in evaluating the interface curvature is the source of all three errors. The curvature error represents the maximum (or average in other studies) difference between calculated and theoretical curvature. The parasitic currents depend on the distribution and magnitude of the curvature error and the pressure error results from an inaccurate estimation of the mean interface curvature.

The drop used for the validation of the interface curvature evaluation has a radius of $r = 2\,m$ and is positioned at the centre of a cubical domain with edge length $8\,m$. The surface tension coefficient of the fluid pair is $\sigma = 73\,N\,m^{-1}$. Therefore, the exact pressure jump across the interface for the considered drop is $\Delta p = 73\,Pa$. The density inside and outside the drop is $\rho_i = 1.0\,kg\,m^{-3}$ and $\rho_o = 0.1\,kg\,m^{-3}$, respectively. A fixed time-step of $\Delta t = 10^{-3}\,s$ is applied. This particular test case with the given properties has previously been used in several publications to examine interface curvature evaluation methods [71, 129, 134, 263]. Thus, the results obtained with CELESTE can be compared to the results obtained with other commonly applied methods to evaluate the interface curvature.

The inviscid drop is simulated on three equidistant Cartesian meshes and three tetrahedral meshes, using CELESTE to determine the interface curvature with various stencil sizes. The Cartesian meshes consist of 20^3, 40^3 (Figure 3.9a) and 80^3 cells, and the applied tetrahedral meshes have approximately 6.0×10^4 (Figure 3.9c), 1.1×10^5 and 2.7×10^5

cells. Alongside simulations without convolution of the colour function, denoted in the remainder with *n.c.* (no convolution), simulations applying the K_{cos}, K_6 and K_8 convolution kernels, discussed in Section 4.2.1, are presented in order to make a sound judgment about the capabilities of the new curvature evaluation method. Stencils of equal size l_s are applied in the following simulations to convolute the colour function, with convolution length $\epsilon = l_s/2$, and to evaluate the interface normal vector and curvature. For the three considered tetrahedral meshes the reference length scale d^* equals to $0.17\,m$ for the mesh with 6.0×10^4 cells, $0.14\,m$ for the mesh with 1.1×10^5 cells and $0.10\,m$ for the mesh with 2.7×10^5 cells.

The accuracy of the interface normal vector and interface curvature evaluation is assessed by means of the L_2 and L_∞ error norms, defined for the interface normal vector as

$$L_2(\boldsymbol{m}) = \sqrt{\frac{1}{N_I}\sum_{i=1}^{N_I}(\boldsymbol{m}_i - \boldsymbol{m}_{exact})^2} \tag{4.32}$$

$$L_\infty(\boldsymbol{m}) = \max\{|\boldsymbol{m} - \boldsymbol{m}_{exact}|\} \tag{4.33}$$

and for the interface curvature as

$$L_2(\kappa) = \sqrt{\frac{1}{N_I}\sum_{i=1}^{N_I}\left(\frac{\kappa_i - \kappa_{exact}}{\kappa_{exact}}\right)^2} \tag{4.34}$$

$$L_\infty(\kappa) = \max\left\{\frac{|\kappa - \kappa_{exact}|}{\kappa_{exact}}\right\}\,, \tag{4.35}$$

where N_I is the number of cells in the interface region. Eqs. 4.32 - 4.35 are evaluated for cells in the interface region only. A cell is considered to lie within the interface region if $10^{-5} \leq \gamma^c \leq 1.0 - 10^{-5}$, following the work of Cummins *et al.* [34]. The exact curvature κ_{exact} for a spherical interface follows from Eq. 3.144. The difference between the computed pressure jump across the interface and the exact pressure jump across the interface (see Eq. 2.46) is quantified by Eq. 3.145.

4.3.5.1. Interface Normal Vector

The accurate evaluation of the interface normal vector is an important feature, since the interface curvature directly follows as a result. Figure 4.4 shows the $L_2(\boldsymbol{m})$ and $L_\infty(\boldsymbol{m})$ error norms for interface normal vector $\boldsymbol{m}$ obtained with CELESTE on the three equidistant Cartesian meshes as a function of stencil size and convolution kernel. The graphs clearly show a decreasing error for calculating the interface normal vector from a convoluted colour function field with increasing stencil size. The differences between the tested convolution kernels reduces with increasing mesh resolution. On all three meshes the polynomial convolution kernels, *i.e.* K_6 and K_8, show a steeper slope for increasing stencil size than the K_{cos} kernel. In instances where the normal vectors are calculated using the original colour function field without convolution, the error remains constant regardless of the stencil size. Considering an unconvoluted colour function, the gradient of the colour function is only dependent on the value at the cell containing the interface and its direct neighbours. Therefore, it does not make a difference if the stencil for the gradient computation is increased. Furthermore, only minor differences are observed for the tested meshes.

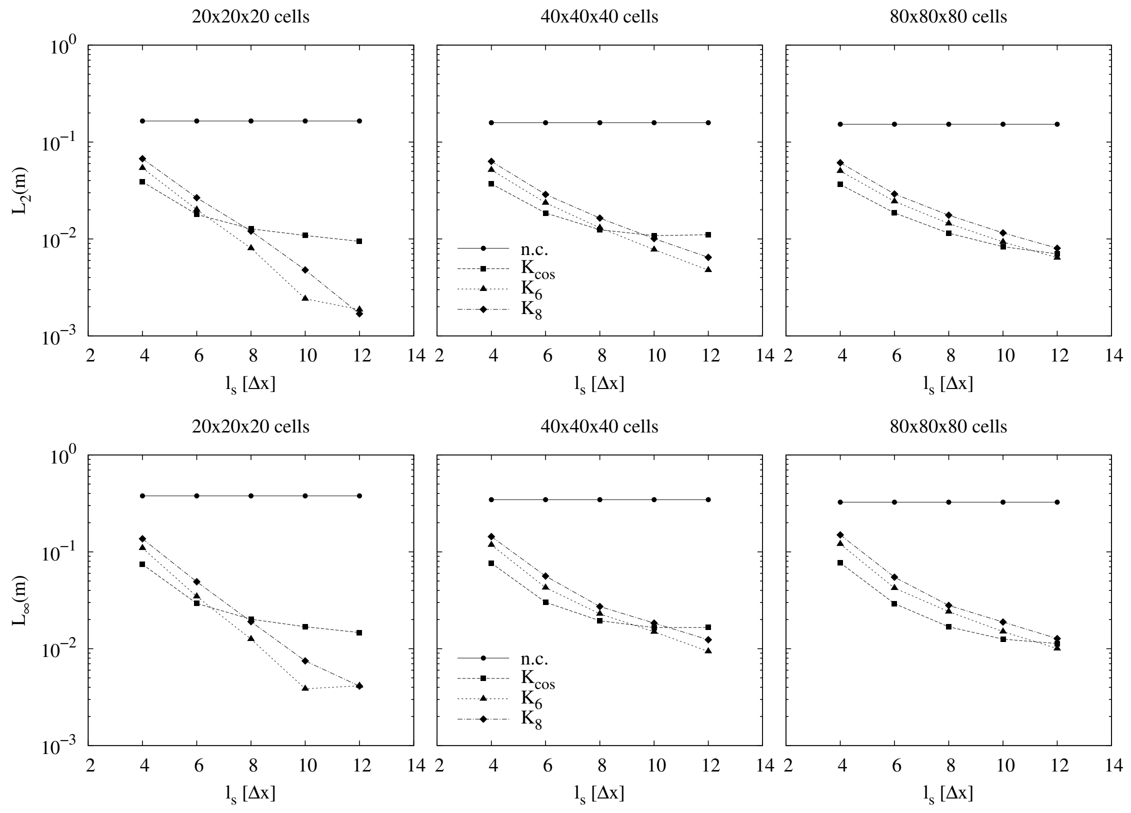


Figure 4.4.: L_2 and L_∞ error norms of interface normal vector $\boldsymbol{m}$ as a function of stencil size l_s and convolution kernel for a spherical drop on three equidistant Cartesian meshes using the CELESTE method.

Figure 4.5 shows the error norms for the interface normal vector as a function of stencil size and convolution kernel on the three tetrahedral meshes. The $L_2(\boldsymbol{m})$ and $L_\infty(\boldsymbol{m})$ error are of similar value on the three considered meshes. As for the Cartesian meshes, a decreasing error is observed with increasing stencil size for cases with convoluted colour function and a constant error is observed using the unconvoluted colour function. The errors for the cases with convoluted colour function are about one order of magnitude higher compared to the errors observed on the Cartesian meshes. The choice of convolution kernel carries only a small influence on the resulting error of the interface normal vector.

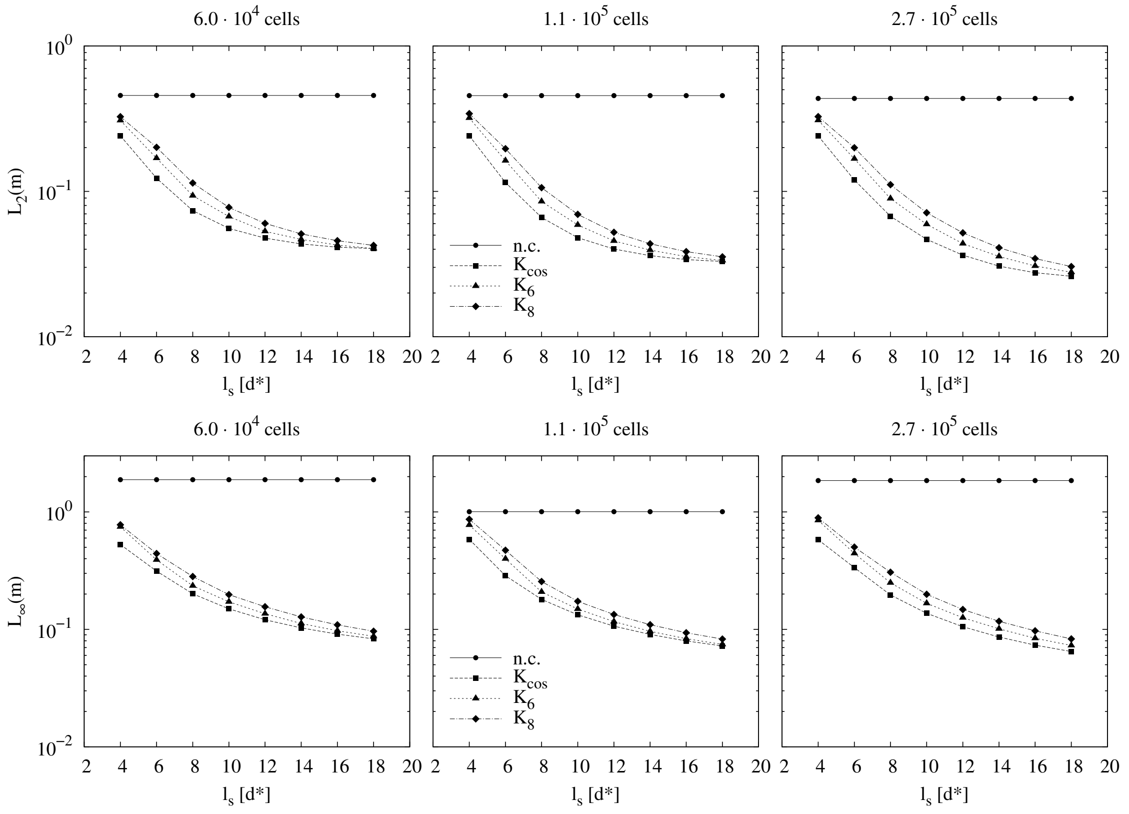


Figure 4.5.: L_2 and L_∞ error norms of interface normal vector $\boldsymbol{m}$ as a function of stencil size l_s and convolution kernel for a spherical drop on three tetrahedral meshes using the CELESTE method.

4.3.5.2. Interface Curvature

The errors induced by the curvature evaluation using the CELESTE method on equidistant Cartesian meshes, presented in Figure 4.6, show no clear correlation with the stencil size. Considering the cases using a convoluted colour function, a large difference for the resulting error between different stencil sizes and different convolution kernels can be observed on the coarsest mesh (20^3 cells). The variations between the presented stencil sizes and convolution kernels successively decrease on the finer meshes (40^3 and 80^3 cells). The results without convolution of the colour function show similar trends as the results obtained with convolution of the colour function. Generally, a higher curvature error is obtained by using the unconvoluted colour function compared to the cases with convolution. However, the difference of the errors induced by the curvature estimate between cases with and without convolution decreases for higher resolved meshes.

The maximum curvature error, represented by $L_\infty(\kappa)$, for the convoluted cases ranges from $1.5 - 38\,\%$ on the 20^3 cells mesh, $0.8 - 5\,\%$ on the 40^3 cells mesh and $2.0 - 3.8\,\%$ on the 80^3 cells mesh. Applying no convolution to the colour function, the curvature errors are $1.9 - 69\,\%$ on the 20^3 cells mesh, $3.2 - 14.7\,\%$ on the 40^3 cells mesh and $6.2 - 8.2\,\%$ on the 80^3 cells mesh. For a two-dimensional circular fluid particle, Mencinger and Žun [154] reported errors of around $8.9\,\%$ and $4.5\,\%$ on Cartesian meshes resolving the radius of the fluid particle radius with 10 and 20 cells, respectively. For this simulations Mencinger and Žun applied the K_8 convolution kernel with a convolution length ϵ equal to the fluid particle radius. Simulations conducted by Martins Villar [151], published in [57], presented a maximum curvature error for a spherical fluid particle of 0.4 % using a height function technique and resolving the interface with 20 cells per diameter. Also reported in [57] are results of van der Heul [241] applying a mass-conserving level-set method, called MCLS [243], to a spherical interface with a reported curvature error of 7.5 %. Studies of Cummins *et al.* [34] applied the K_8 kernel with a convolution length of $\epsilon = 4\Delta x$ to a two-dimensional circular interface. The resulting curvature errors are included in Figure 4.6 for comparison. The curvature error obtained with CELESTE, even without convolution, is equal to or smaller than in the results reported by Cummins *et al.* [34] using standard finite difference methods for the evaluation of the interface curvature.

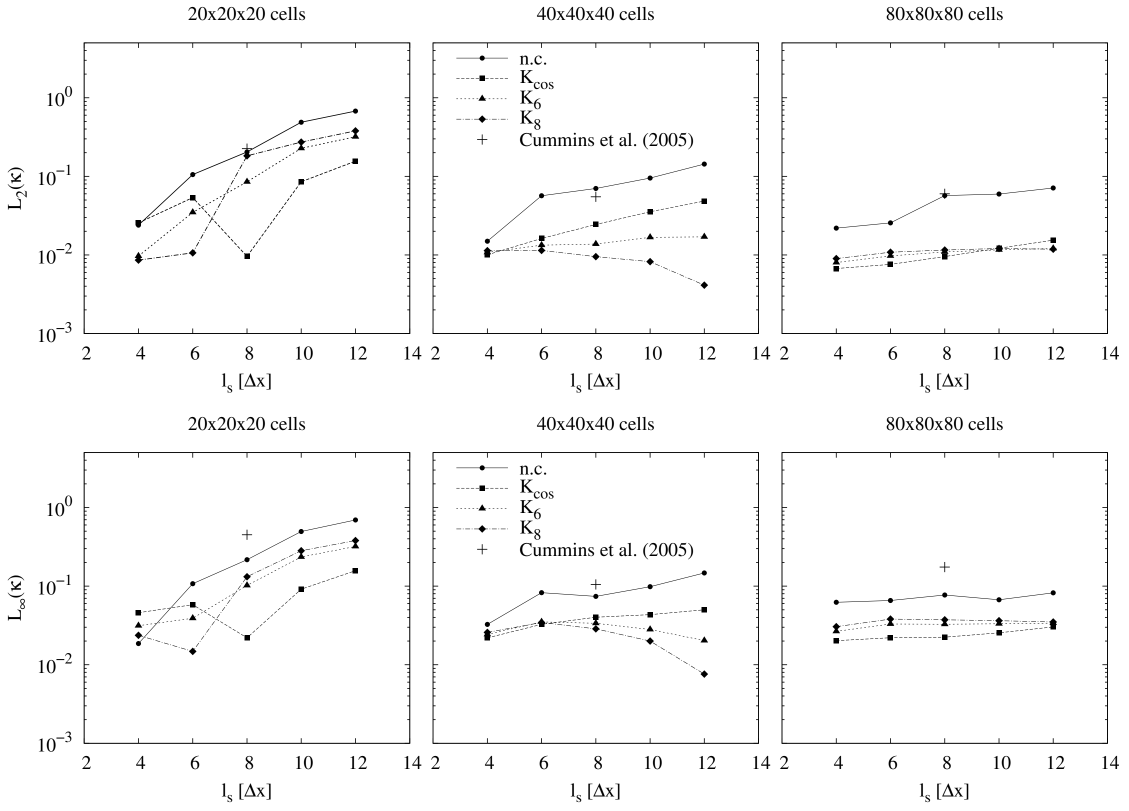


Figure 4.6.: L_2 and L_∞ error norms of interface curvature κ as a function of stencil size l_s and convolution kernel for a spherical drop on three equidistant Cartesian meshes using the CELESTE method.

On the tetrahedral meshes, the interface curvature does not feature the large errors observed for the interface normal vector. The $L_2(\kappa)$ and $L_\infty(\kappa)$ errors in Figure 4.7 show similar error magnitudes on the tetrahedral meshes as previously seen on the Cartesian meshes. The curvature errors for a stencil size of $l_s = 4d^*$, however, are approximately one order of magnitude higher than for a stencil size of $l_s = 6d^*$. This strongly suggests that a stencil size of $l_s = 4d^*$ is not large enough. The results of all three convolution kernels and the results obtained without convolution of the colour function show similar trends on all tested tetrahedral meshes.

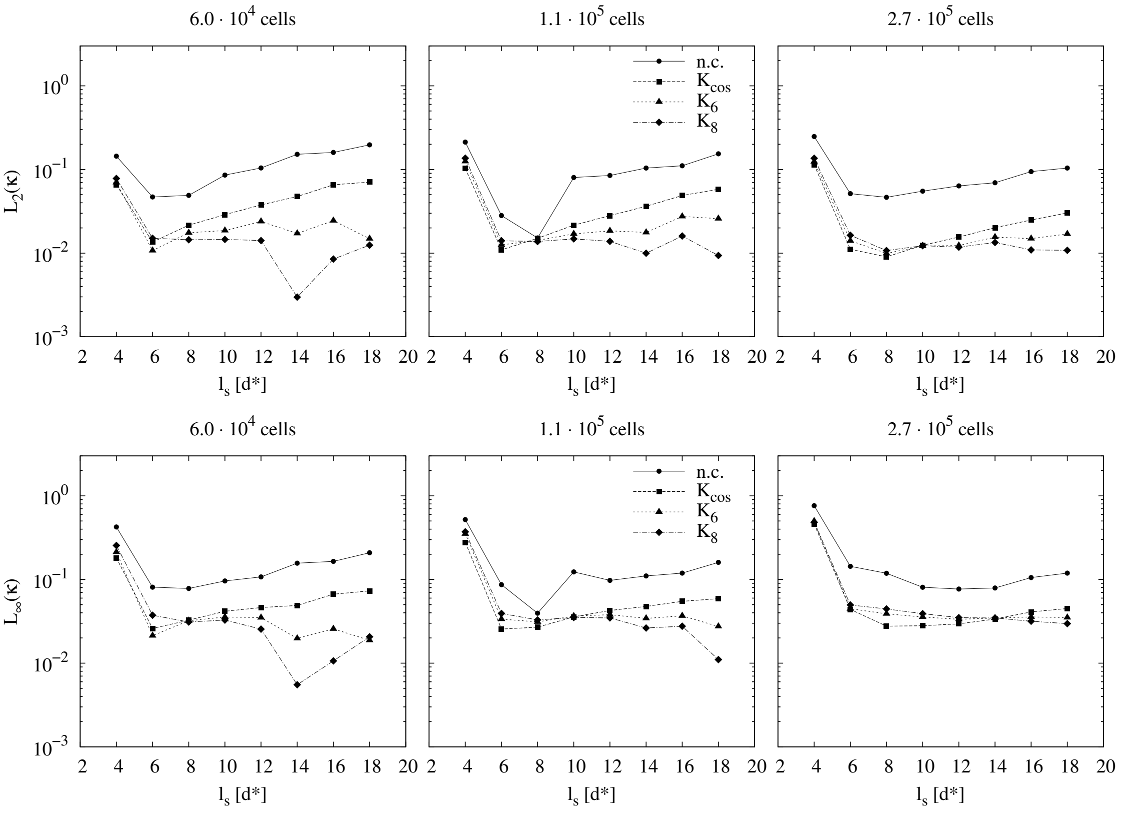


Figure 4.7.: L_2 and L_∞ error norms of interface curvature κ as a function of stencil size l_s and convolution kernel for a spherical drop on three tetrahedral meshes using the CELESTE method.

4.3.5.3. Pressure Error

The error in pressure jump across the interface obtained on the Cartesian meshes with varying stencil size is shown in Figure 4.8. With increasing mesh resolution, the variation in $E(\Delta p_{max})$ between the different stencil sizes reduces significantly. If convolution is applied, the difference between convolution kernels similarly diminishes on higher resolved meshes. The pressure error is generally around the order of 10^{-2}.

The magnitude of the pressure error $E(\Delta p_{max})$ on the tetrahedral meshes is of similar magnitude compared to the pressure errors obtained on the Cartesian meshes, being around 10^{-2} as shown in Figure 4.9. A notable decrease in variation of $E(\Delta p_{max})$ on finer meshes for the tested stencil sizes and convolution methods is observed. By not convoluting the colour function, the pressure error has a similar trend with respect to stencil size and mesh resolution as the results of the cases in which the colour function is convoluted, however, the magnitude of the error is higher.

A close relationship between the pressure error across the interface $E(\Delta p_{max})$ after one time-step and the curvature error $L_2(\kappa)$ can be identified by comparing the graphs in Figures 4.6 and 4.8 for Cartesian meshes and in Figures 4.7 and 4.9 for tetrahedral meshes. The observed correlation between pressure error and curvature error has physical meaning given the relationship between pressure jump across the interface and surface force.

Assuming a constant surface tension coefficient, the pressure jump across the interface follows directly from the surface force, which is proportional to the interface curvature.

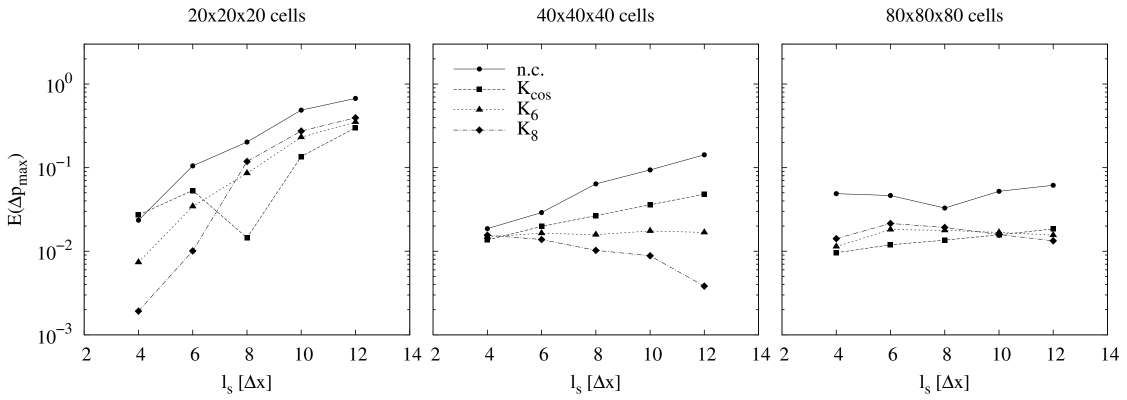


Figure 4.8.: Pressure error $E(\Delta p_{max})$ after one time-step as a function of stencil size l_s and convolution kernel for a static inviscid drop in equilibrium on three equidistant Cartesian meshes using the CELESTE method. The drop with surface tension coefficient $\sigma = 73\,N\,m^{-1}$ and radius $r = 2\,m$ is positioned at the centre of the $8\,m \times 8\,m \times 8\,m$ domain and the time-step is $10^{-3}\,s$.

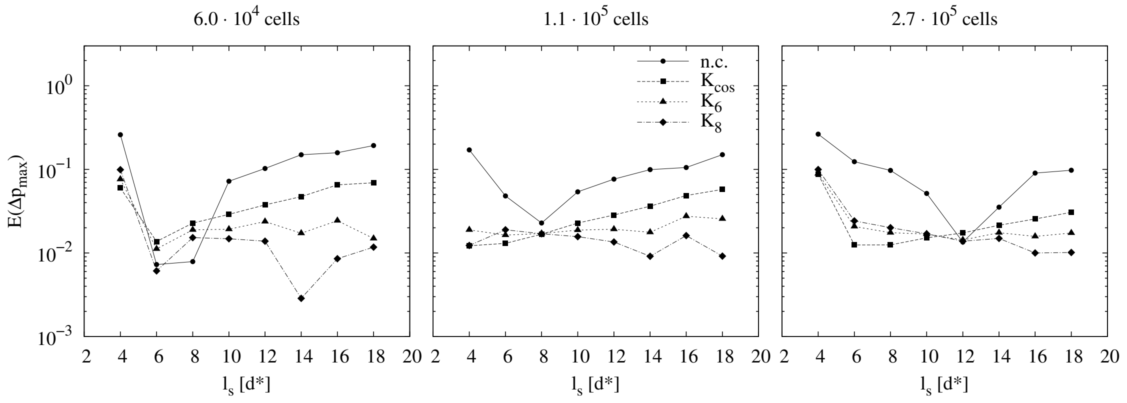


Figure 4.9.: Pressure error $E(\Delta p_{max})$ after one time-step as a function of stencil size l_s and convolution kernel for a static inviscid drop in equilibrium on three tetrahedral meshes using the CELESTE method. The drop with surface tension coefficient $\sigma = 73\,N\,m^{-1}$ and radius $r = 2\,m$ is positioned at the centre of the $8\,m \times 8\,m \times 8\,m$ domain and the time-step is $10^{-3}\,s$.

4.3.5.4. Parasitic Currents

The magnitude of parasitic currents after one time-step on the applied equidistant Cartesian meshes, depicted in Figure 4.10, shows no clear trend concerning mesh resolution or stencil size. The polynomial convolution kernels show a similar performance, with the K_{cos} kernel performing slightly better on finer meshes than its polynomial counterparts. The parasitic currents developing with the unconvoluted colour function are of similar magnitude compared to the convoluted cases on the 20^3 mesh, but are considerably higher on the finer meshes. The results show a widening gap between the case using the unconvoluted colour function to the cases using the convoluted colour function with increasing mesh resolution. This suggests an increasing impact and importance of convolution on finer meshes with respect to parasitic currents. Figure 4.11 depicts the typical velocity distribution around the interface after one and fifty time-steps for the case using the K_{cos} convolution function and a stencil size of $l_s = 4\Delta x$.

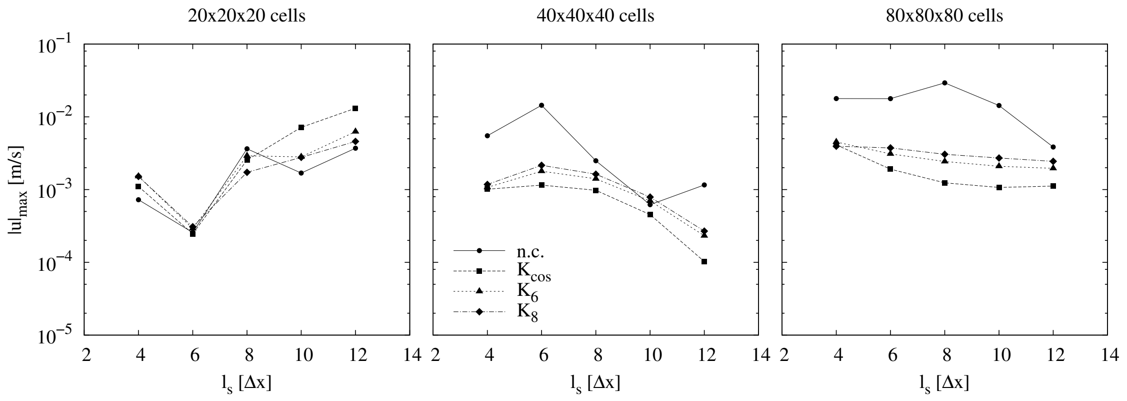
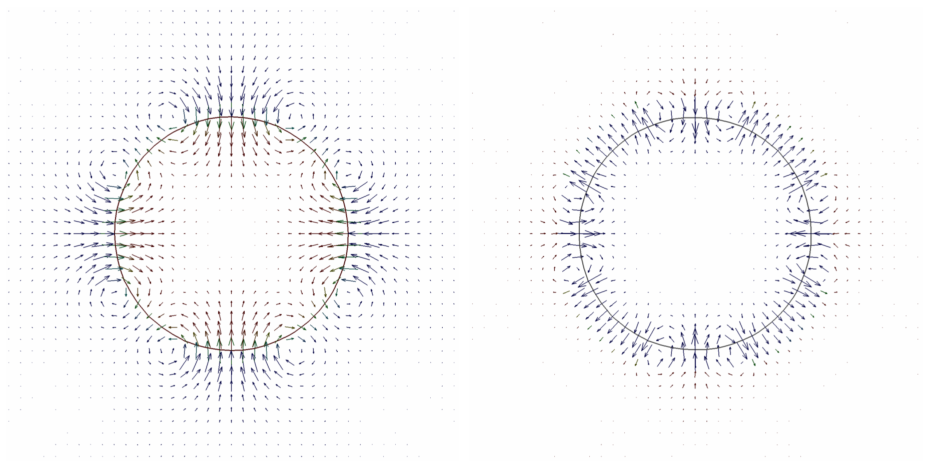


Figure 4.10.: Maximum velocity magnitude after one time-step as a function of stencil size l_s and convolution kernel for a static inviscid drop in equilibrium on three equidistant Cartesian meshes using the CELESTE method. The drop with surface tension coefficient $\sigma = 73\,N\,m^{-1}$ and radius $r = 2\,m$ is positioned at the centre of the $8\,m \times 8\,m \times 8\,m$ domain and the time-step is $10^{-3}\,s$.



(a) $t = \Delta t$ (b) $t = 50\,\Delta t$

Figure 4.11.: Vector field of the parasitic currents obtained using the CELESTE method in conjunction with the K_{cos} convolution function and a stencil size of $l_s = 4\Delta x$ after one time-step and fifty time-steps. The velocity vectors are shown in the x-y plane crossing through the centre of the domain. The length of the vectors represents the velocity magnitude, with the maximum magnitudes given in Table 4.2. The vector field after one time-step (a) is magnified 10 times compared to the vector field after fifty time-steps (b).

Table 4.1 presents maximum velocity magnitudes published in different studies for the static inviscid drop in equilibrium on an equidistant Cartesian mesh of 40^3 cells and

Table 4.1.: Reference values for the maximum velocity magnitude $[m/s]$ after one and fifty time-steps for a static inviscid drop in equilibrium. The static inviscid drop with surface tension coefficient $\sigma = 73\,N\,m^{-1}$ and radius $r = 2\,m$ is positioned at the centre of the $8\,m \times 8\,m \times 8\,m$ domain, which is resolved by an equidistant Cartesian mesh of 40^3 cells and a time-step of $10^{-3}\,s$.

Method	Publication	$t = \Delta t$	$t = 50\Delta t$
Convolution	Williams *et al.* [263]	8.55×10^{-2}	3.86×10^{-1}
	Francois *et al.* [71]	4.87×10^{-3}	1.63×10^{-1}
Height function	Denner *et al.* [57]	7.92×10^{-3}	4.26×10^{-2}
	Francois *et al.* [71]	4.02×10^{-3}	4.02×10^{-2}
	López *et al.* [135]	$\approx 4.0 \times 10^{-3}$	$\approx 4.0 \times 10^{-2}$
	Liovic *et al.* [129]	2.30×10^{-3}	3.35×10^{-2}
Mass-Conserving Level-Set	Denner *et al.* [57]	2.98×10^{-3}	2.65×10^{-2}

Table 4.2 presents the velocity magnitudes obtained with the proposed curvature evaluation method CELESTE. Comparing Tables 4.1 and 4.2, the CELESTE method performs very well against the listed references. The maximum velocity magnitude obtained with CELESTE after one time-step is smaller with all tested setups using one of the three convolution kernels than the results presented in the literature. The studies using a height function method generally show a smaller increase in velocity over time, leading to smaller parasitic currents after fifty time-steps in some instances.

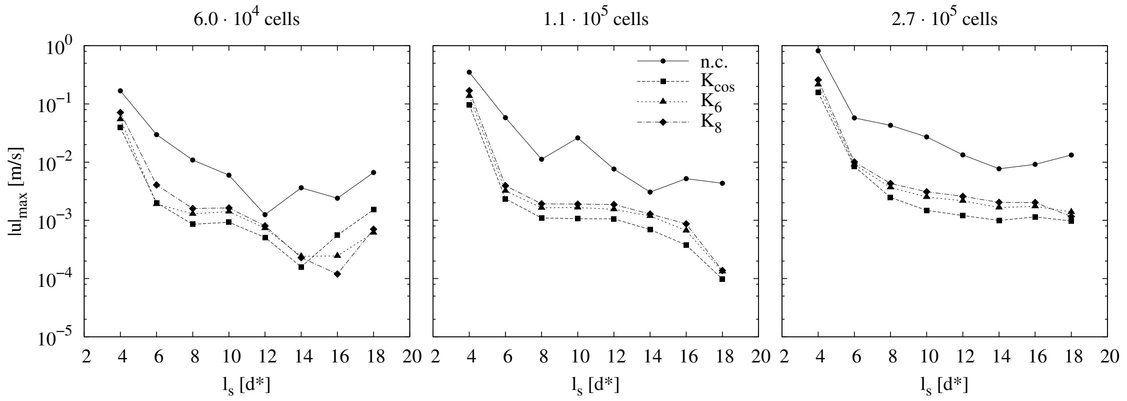


Figure 4.12.: Maximum velocity magnitude after one time-step as a function of stencil size l_s and convolution kernel for a static inviscid drop in equilibrium on three tetrahedral meshes using the CELESTE method. The drop with surface tension coefficient $\sigma = 73\,N\,m^{-1}$ and radius $r = 2\,m$ is positioned at the centre of the $8\,m \times 8\,m \times 8\,m$ domain and the time-step is $10^{-3}\,s$.

The magnitude of parasitic currents on the tetrahedral meshes, shown in Figure 4.12, are of similar magnitude as the parasitic currents observed on Cartesian meshes, especially on the higher resolved meshes. Generally, the maximum velocity magnitude after one time-step is around $10^{-3}\,m/s$ for cases with convoluted colour function, with lower magnitude

Table 4.2.: Maximum velocity magnitude $[m/s]$ obtained with CELESTE after one and fifty time-steps for a static inviscid drop in equilibrium. The static inviscid drop with surface tension coefficient $\sigma = 73\,N\,m^{-1}$ and radius $r = 2\,m$ is positioned at the centre of the $8\,m\times 8\,m\times 8\,m$ domain, resolved by an equidistant Cartesian mesh of 40^3 cells and a time step of $10^{-3}\,s$.

Convolution kernel	Stencil	$t = \Delta t$	$t = 50\Delta t$
K_{cos}	$4\Delta x$	1.010×10^{-3}	6.746×10^{-2}
	$6\Delta x$	1.153×10^{-3}	8.118×10^{-2}
	$8\Delta x$	9.794×10^{-4}	7.102×10^{-2}
	$10\Delta x$	4.527×10^{-4}	3.356×10^{-2}
	$12\Delta x$	1.025×10^{-4}	7.436×10^{-3}
K_6	$4\Delta x$	1.074×10^{-3}	6.896×10^{-2}
	$6\Delta x$	1.789×10^{-3}	1.234×10^{-1}
	$8\Delta x$	1.407×10^{-3}	1.021×10^{-1}
	$10\Delta x$	6.700×10^{-4}	5.168×10^{-2}
	$12\Delta x$	2.324×10^{-4}	1.700×10^{-2}
K_8	$4\Delta x$	1.178×10^{-3}	7.619×10^{-2}
	$6\Delta x$	2.155×10^{-3}	1.488×10^{-1}
	$8\Delta x$	1.630×10^{-3}	1.176×10^{-1}
	$10\Delta x$	7.885×10^{-4}	5.825×10^{-2}
	$12\Delta x$	2.689×10^{-4}	1.962×10^{-2}

for larger stencils. No significant differences are observed between the tested convolution kernels. In cases where no convolution is applied, the magnitude of parasitic currents is higher than for the cases with convolution of the colour function, but with a similar trend concerning stencil size. Similar to the aforementioned curvature error, the stencil size of $l_s = 4d^*$ produces significantly higher parasitic currents. As observed on the Cartesian meshes, the impact of convolution increases on finer meshes.

4.3.5.5. Parasitic Kinetic Energy

As explained in Section 2.3.3, Lafaurie *et al.* [123] found the velocity magnitude of parasitic currents to be proportional to the ratio of surface tension coefficient σ and fluid viscosity μ, described as

$$|\boldsymbol{u}| = C\frac{\sigma}{\mu} \ , \tag{4.36}$$

where C is a constant depending on the interface capturing/tracking method. In fact, coefficient C represents the capillary number of the parasitic currents, since the capillary number is defined as $Ca = |\boldsymbol{u}|\,\mu/\sigma$ and, therefore, $|\boldsymbol{u}| = Ca\,\sigma/\mu$. Typical values of C range from 10^{-2} to 10^{-10} [123, 250]. The results of various studies presented in the literature are given in Table 4.3.

In order to validate the potential of the curvature evaluation method CELESTE, a viscous static drop in equilibrium is simulated. Assuming only moderate changes of the

Table 4.3.: Parasitic currents coefficients reported in the literature for various VOF methods.

Method		Grid resolution cells / radius	C
SURFER	[123]	-	$\approx 10^{-2}$
PLIC	[184]	$3.2 - 102.4$	$\approx 10^{-3}$
VOF-LS	[3]	$5 - 50$	$\approx 10^{-3}$
K_8	[154]	10	2.39×10^{-4}
K_8	[154]	20	6.58×10^{-5}
PROST	[154]	10	8.17×10^{-5}
PROST	[154]	20	1.31×10^{-5}
Advected normals	[193]	32	6.01×10^{-7}

interface shape as a result of parasitic currents, the kinetic energy induced by the parasitic currents and the viscous dissipation eventually reach an equilibrium. Similar to the drop considered in the previous sections, the simulated viscous drop has a radius of $r = 2\,m$ and is positioned at the centre of a cubical domain with edge length $8\,m$. The surface tension coefficient of the interface is $\sigma = 73\,N\,m^{-1}$ and no gravity is present. The density of both fluids is $\rho = 1\,kg\,m^{-3}$ and both fluids hold a viscosity of $\mu = 1\,Pa\,s$.

Table 4.4 presents coefficient C based on the maximum and mean velocity magnitude, and the total kinetic energy W_k for three equidistant Cartesian meshes and three tetrahedral meshes. Coefficient C based on the maximum and mean velocity is defined as

$$C_{max} = |\boldsymbol{u}|_{max} \frac{\mu}{\sigma} \tag{4.37}$$

and

$$C_{mean} = |\boldsymbol{u}|_{mean} \frac{\mu}{\sigma} \ , \tag{4.38}$$

respectively. The mean velocity in the domain is calculated as

$$|\boldsymbol{u}|_{mean} = \frac{1}{N} \sum_{P=1}^{N} \boldsymbol{u}_P \tag{4.39}$$

and the total kinetic energy in the domain is defined as

$$W_k = \sum_{P=1}^{N} \frac{1}{2} \rho_P \, |\boldsymbol{u}|_P^2 \, V_P \ , \tag{4.40}$$

where N is the number of cells in the domain.

The results presented in Table 4.4 show a decreasing velocity and kinetic energy for Cartesian meshes with increasing mesh resolution. The coefficients C obtained with CELESTE are comparable to the coefficients presented by Mencinger and Žun [154] using the piecewise parabolic reconstruction method PROST of Renardy and Renardy [197].

Table 4.4.: Mean and maximum parasitic currents and total kinetic energy on equidistant Cartesian and tetrahedral meshes for a static viscous drop in equilibrium using the CELESTE method to evaluate the interface curvature. The drop with surface tension coefficient $\sigma = 73\, N\, m^{-1}$ and radius $r = 2\, m$ is positioned at the centre of the $8\, m \times 8\, m \times 8\, m$ domain and the time-step is $10^{-3}\, s$. Both fluids have a density of $\rho = 1\, kg\, m^{-3}$ and a viscosity of $\mu = 1\, Pa\, s$.

Mesh	C_{mean}	C_{max}	$W_k\, [J]$
Equidistant Cartesian, 20^3 cells	1.27×10^{-5}	1.71×10^{-4}	8.085×10^{-4}
Equidistant Cartesian, 40^3 cells	1.40×10^{-6}	6.00×10^{-5}	2.413×10^{-5}
Equidistant Cartesian, 80^3 cells	4.10×10^{-7}	2.52×10^{-5}	1.267×10^{-6}
Tetrahedral, $\approx 6.0 \times 10^4$ cells	4.40×10^{-6}	1.47×10^{-4}	1.670×10^{-4}
Tetrahedral, $\approx 1.1 \times 10^5$ cells	4.10×10^{-6}	1.05×10^{-4}	7.990×10^{-5}
Tetrahedral, $\approx 2.7 \times 10^5$ cells	2.41×10^{-6}	1.45×10^{-4}	3.986×10^{-5}

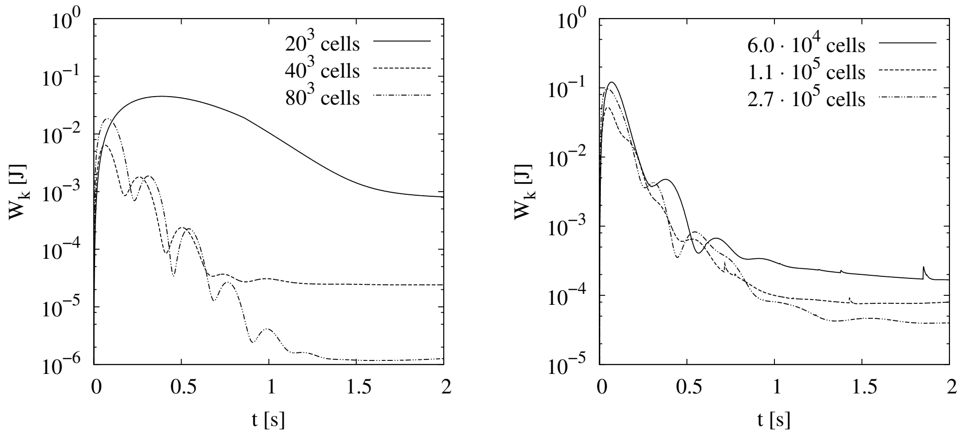
Moreover, the results presented here are better than the results obtained by Mencinger and Žun [154] using a similar convoluted colour function approach with K_8 convolution. It should be noted that Mencinger and Žun applied the K_8 convolution kernel with a significantly larger convolution length, *i.e.* convolution length equal to the fluid particle radius, which is expected to produce smaller parasitic currents. This issue is revisited in more detail in Section 4.5. On tetrahedral meshes, coefficients C of similar magnitude as for the Cartesian meshes are generally obtained. The C_{max} coefficient on the finest of the three tetrahedral meshes is higher than on the second finest tetrahedral mesh. The coefficient C_{mean} and the kinetic energy in the domain, however, decrease with increasing mesh resolution on the tetrahedral mesh as well, indicating that the higher C_{max} coefficient and the underlying maximum velocity on the finest tetrahedral mesh are not representative for the entire domain. Figure 4.13 shows the evolution of total kinetic energy as a function of time. In general, mesh refinement reduces terminal parasitic currents and the resulting kinetic energy. The evolution of the kinetic energy decreases rapidly with increasing mesh resolution, on Cartesian as well as on tetrahedral meshes.

4.4. Convolution of Fluid Properties and Surface Force

The content of this section has been published in:

[56] Denner, F. and van Wachem, B.G.M.: On the convolution of fluid properties and surface force for interface capturing methods. International Journal of Multiphase Flow, 54 (2013), pp. 61-64.

The convolution of fluid properties, *i.e.* density ρ and viscosity μ, and surface force $\boldsymbol{f}_s$ has previously been discussed in other studies, *e.g.* [19, 192, 240, 272], but remains a controversial and essentially unsolved issue. The convolution of density, viscosity and surface force facilitates a smooth transition of momentum across the interface, which improves the convergence of the numerical solver and reduces numerical oscillations. On



(a) Cartesian mesh (b) Tetrahedral mesh

Figure 4.13.: Evolution of the total kinetic energy W_k as a function of time for the viscous static drop in equilibrium using the CELESTE method. The drop with surface tension coefficient $\sigma = 73\,N\,m^{-1}$ and radius $r = 2\,m$ is positioned at the centre of the $8\,m \times 8\,m \times 8\,m$ domain and the time-step is $10^{-3}\,s$. Both fluids have a density of $\rho = 1\,kg\,m^{-3}$ and a viscosity of $\mu = 1\,Pa\,s$.

the other hand, keeping the fluid properties and the surface force as sharp as possible is desired from a physical viewpoint. Numerous studies [102, 123, 226, 240, 262, 263, 272] support the convolution of fluid properties and surface force as it diminishes numerical oscillations and, supposedly, reduces parasitic currents. Wang and Tong [256] and Hong *et al.* [98] argued that fluid properties should be distributed continuously, *i.e.* should be convoluted, since the velocity field is continuous across the interface, but that the surface force should not be convoluted to maintain a sharp pressure jump at the interface. Yet other studies either only convoluted the viscosity, keeping the density and the surface force sharp [29, 78, 243], or applied no convolution [147], applying density, viscosity and surface force as-is.

Eight convolution strategies with respect to fluid properties and surface force are validated and compared. The considered constellations, where superscript u denotes unconvoluted variables and superscript c denotes convoluted variables, are:

1. $\rho^u - \mu^u - f^u_{s,i}$ (no convolution),
2. $\rho^c - \mu^u - f^u_{s,i}$ (convolution of density),
3. $\rho^u - \mu^c - f^u_{s,i}$ (convolution of viscosity),
4. $\rho^c - \mu^c - f^u_{s,i}$ (convolution of fluid properties),
5. $\rho^u - \mu^u - f^c_{s,i}$ (convolution of surface force),
6. $\rho^c - \mu^u - f^c_{s,i}$ (convolution of density and surface force),
7. $\rho^u - \mu^c - f^c_{s,i}$ (convolution of viscosity and surface force),

8. $\rho^c - \mu^c - f^c_{s,i}$ (convolution of fluid properties and surface force).

For cases in which density is convoluted, the colour function γ in Eq. 2.44 is replaced by its convoluted counterpart γ^c. Therefore, the convoluted density follows as

$$\rho^c = \rho_A(1-\gamma^c) + \rho_B\gamma^c \ . \tag{4.41}$$

If viscosity is convoluted, the viscosity distribution is based on the convoluted colour function in a similar manner following Eq. 2.45, defined as

$$\mu^c = \mu_A(1-\gamma^c) + \mu_B\gamma^c \ . \tag{4.42}$$

The surface force is convoluted by replacing the colour function gradient in Eq. 2.48 with the gradient of the convoluted colour function. Thus, the convoluted surface force $f^c_{s,i}$ is defined as

$$f^c_{s,i} = \sigma\,\kappa\,\frac{\partial \gamma^c}{\partial x_i} \ . \tag{4.43}$$

The colour function γ is convoluted by means of the K_{cos} kernel defined in Eq. 4.8 with a convolution length of $\epsilon = 2\Delta x$. Convolution is performed immediately after the colour function advection. Dependent on the considered test case, the unconvoluted or convoluted properties are used in the discretised equations.

Two representative test cases, a static drop in equilibrium and a bubble rising due to buoyancy, are simulated to examine the considered convolution strategies. The static drop in equilibrium is used to assess the effect of convolution on surface-tension-dominated two-phase flows with high density and viscosity ratios. Examining the influence of convolution on viscous stresses and buoyancy is the focus of the rising bubble case. The interface curvature in both test cases is computed using the CELESTE method presented in Section 4.3.

4.4.1. Static Drop in Equilibrium

A static drop in equilibrium is considered with the aim of evaluating the evolution of parasitic currents induced by discretisation inaccuracies. Since the static system is initially in equilibrium, any velocities developing in the domain are regarded as parasitic currents. Assuming the parasitic currents are small enough so that they do not significantly affect the shape of the interface, the magnitude of parasitic currents should converge to an equilibrium value, where the kinetic energy induced by the parasitic currents is equal to the dissipation in the system. The drop with radius $r = 2\,m$ and surface tension coefficient $\sigma = 100\,N\,m^{-1}$ is positioned at the centre of an $8\,m \times 8\,m \times 8\,m$ domain and resolved with 10 cells per diameter. The considered density ratios ρ_i/ρ_o and viscosity ratios μ_i/μ_o are 10^3 and 10^6, where i and o denote fluid properties inside and outside the drop, respectively.

The results in Figure 4.14 demonstrate a considerable effect of the density and viscosity treatment on parasitic currents. In general, the effect of density and viscosity convolution on the parasitic currents increases with increasing density and viscosity ratios. In most cases, parasitic currents are significantly reduced by convoluting density and viscosity.

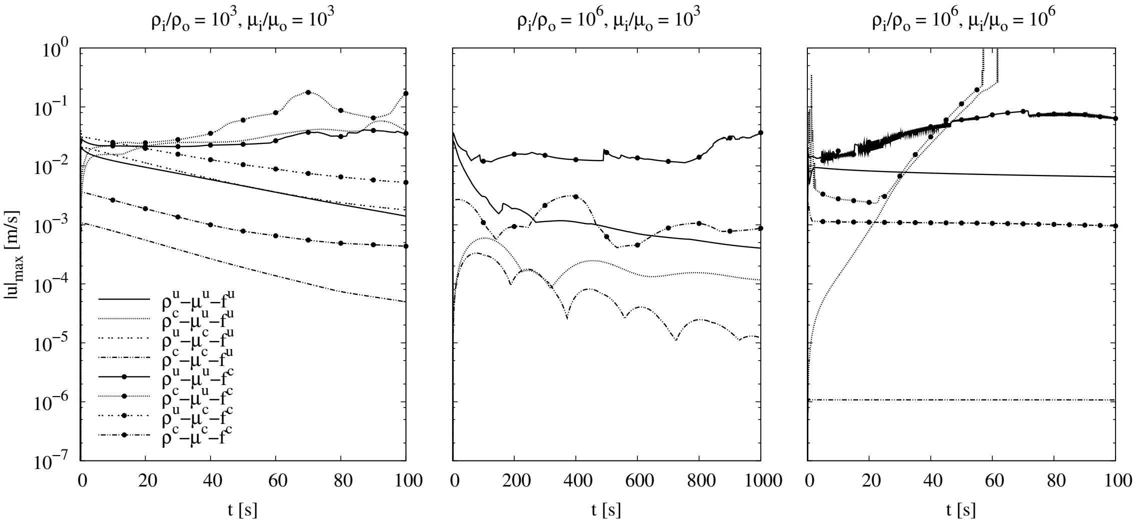


Figure 4.14.: Maximum parasitic currents as a function of time for a spherical drop in mechanical equilibrium with density ratios ρ_i/ρ_o and viscosity ratios μ_i/μ_o of 10^3 and 10^6, using different convolution strategies with respect to density ρ, viscosity μ and surface force $\boldsymbol{f}_s$. Superscript u denotes unconvoluted variables and superscript c denotes convoluted variables.

The cases in which viscosity is convoluted but density is not do not yield a stable result for a density ratio of 10^6. The interface in these cases is destroyed within the first five time-steps as a result of high parasitic currents. Therefore, these cases are not depicted in Figure 4.14b and 4.14c. Similar results can be observed in Figure 4.14c for convolution strategies in which density is convoluted but viscosity is not convoluted, where parasitic currents increase monotonically and the simulations eventually diverge. However, the simulations are stable if the convolution of density and viscosity is treated equally, with decreasing or constant parasitic currents for all tested density and viscosity ratios. This proves a critical enhancement of numerical stability if density and viscosity are treated equally, *i.e.* either both unconvoluted or both convoluted. Regarding the convolution of the surface force, the results in Figure 4.14 show a higher magnitude and a slower decay of parasitic currents for cases with convoluted surface force compared to the respective cases with unconvoluted surface force. The convolution of surface force does not improve the numerical stability for the performed test cases.

4.4.2. Spherical Cap Bubble Rising due to Buoyancy

The rise velocity u_r as a function of time for a bubble rising under the sole action of gravity, using the eight aforementioned convolution strategies, is depicted in Figure 4.15. The bubble is initially spherical with a diameter of $d_0 = 0.02\,m$, a Morton number of $Mo = g\,\mu_o^4/\rho_o\,\sigma^3 = 0.056$ and an Eötvös number of $Eo = \rho_o\,g\,d_0^2/\sigma = 40$. The density and viscosity ratios of the two fluids are $\rho_i/\rho_o = \mu_i/\mu_o = 10^{-2}$. The resolution of the equidistant Cartesian mesh corresponds to 20 cells per initial diameter d_0 and the domain has a width of $5d_0$. Empirical studies by Clift *et al.* [27, Fig. 2.5] suggest a terminal

Reynolds number of $Re_d = u_t \rho_o d_0 / \mu_o \approx 20.5 - 21.0$ for this bubble, representing a terminal Froude number of $Fr = u_t / \sqrt{d_0 g} \approx 0.626 - 0.642$, where u_t stands for the terminal rise velocity. Given the finite extend of the computational domain, the expected terminal rise velocity is corrected by means of the semi-empirical correlation [88]

$$\frac{u_t}{u_t^\infty} \approx 1 - \left(\frac{d_0}{L}\right)^2 , \tag{4.44}$$

where L denotes the domain extend perpendicular to the gravitational acceleration and u_t^∞ represents the terminal rise velocity in a domain of infinite extend. Thus, the expected terminal Froude number of the rising bubble in the given computational domain is $Fr \approx 0.601 - 0.616$.

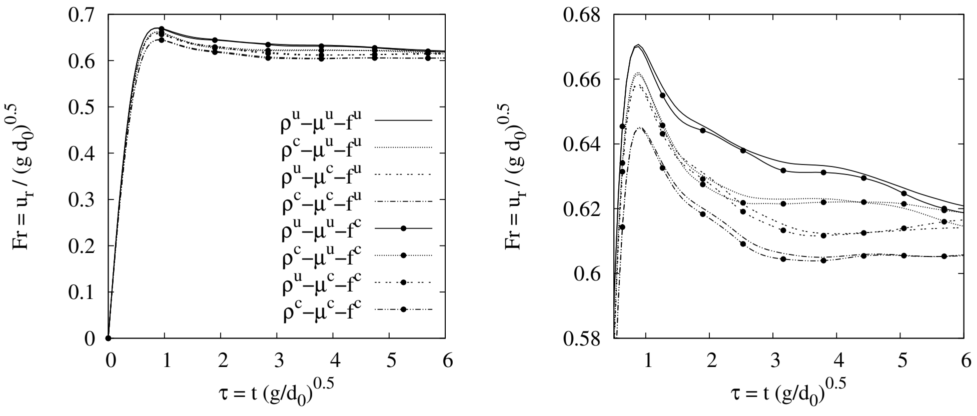


Figure 4.15.: Froude number of an initially spherical bubble rising due to buoyancy as a function of non-dimensional time $\tau = t\sqrt{g/d_0}$, using different convolution strategies with respect to density ρ, viscosity μ and surface force $\boldsymbol{f}_s$. Superscript u denotes unconvoluted variables and superscript c denotes convoluted variables.

At first glance, examining the left graph in Figure 4.15, all considered convolution strategies produce acceptable predictions of the rise velocity, with terminal Froude numbers ranging from 0.606 to 0.621. However, analysing the detailed presentation of the rise velocity evolution given in the right graph of Figure 4.15, considerable differences between the tested convolution strategies can be observed. Comparing cases for which only the convolution of the surface force differs show equivalent results, contrary to the significant differences observed for the static drop in the previous section. The dominating effect in case of the static drop is surface tension whereas the rising bubble is governed by buoyancy and viscous stresses, which greatly reduces the influence of the surface force on the fluid system. In contrast to the static drop in equilibrium where viscosity is merely responsible for the dissipation of parasitic currents, viscous stresses play an important role in the evolution of the bubble shape. This explains the large differences caused by the convolution of viscosity, as observed in Figure 4.15. The results also suggest that using the convoluted density field is superior to using the discontinuous density field, as the convolution strategies in which viscosity as well as density are convoluted predict the terminal rise velocities most accurate.

4.4.3. Conclusion

The analysis of the two representative test cases highlight the significant influence of convolution on the predictive quality of interfacial flow simulations. The presented results of the two test cases indicate that the convolution of density improves the results considerably, even though it is questionable from a strictly physical point of view. In cases where the accurate prediction of viscous stresses is essential, ensuring a continuous variation of viscous stresses across the interface by convoluting viscosity is important, as shown by the rising bubble case. Crucially, the equal treatment of density and viscosity with respect to convolution is essential to maintain numerical stability for high density ratios, as demonstrated by the static drop in equilibrium with density ratio 10^6. The test cases also comprehensively demonstrate that the convolution of the surface force adversely affects the development and dissipation of parasitic currents in surface-tension-dominated flows and, contrary to the generally accepted notion, does not improve the numerical stability.

4.5. Influence of Convolution Length

The content of this section has in parts been published in:

[53] Denner, F. and van Wachem, B.G.M.: Two-Phase Flow Modelling on Arbitrary Meshes: Superior VOF Curvature Estimation and the Issue of Convolution. International Conference on Numerical Methods in Multiphase Flows, 12 - 14 June 2012, State College, PA, USA.

[55] Denner, F. and van Wachem, B.G.M.: Fully-coupled balanced-force VOF framework for arbitrary meshes with least-squares curvature evaluation from volume fraction. Numerical Heat Transfer, Part B: Fundamentals, accepted for publication. DOI: 10.1080/10407790.2013.849996

The results presented in Section 4.3.5 show notable differences for different convolution lengths. Generally, a larger convolution length results in smaller parasitic currents and a more accurate estimate of the interface normal vector. Studies by Francois *et al.* [71], Williams *et al.* [263] and most notably Williams [262] and Cummins *et al.* [34] support these findings. Williams [262] studied various convolution kernels in detail and demonstrated a decreasing error when calculating interface normal vectors with increasing convolution length, as the interface region is resolved by an increasing number of mesh cells. Misleadingly, such results suggest that a large convolution stencil is preferable over a compact convolution stencil. Figure 4.16 shows the parasitic currents and the error in pressure jump across the interface, as defined in Eq. 3.145, for the static inviscid drop in equilibrium as a function of convolution length ϵ. The parasitic currents are reduced significantly with increasing convolution length on both meshes. However, the pressure error develops in the opposite direction, rising monotonically with increasing convolution length. Thus, evaluating only the parasitic currents, as the literature suggests is a popular practice, would lead to the false conclusion that a large convolution stencil is better than a compact convolution stencil.

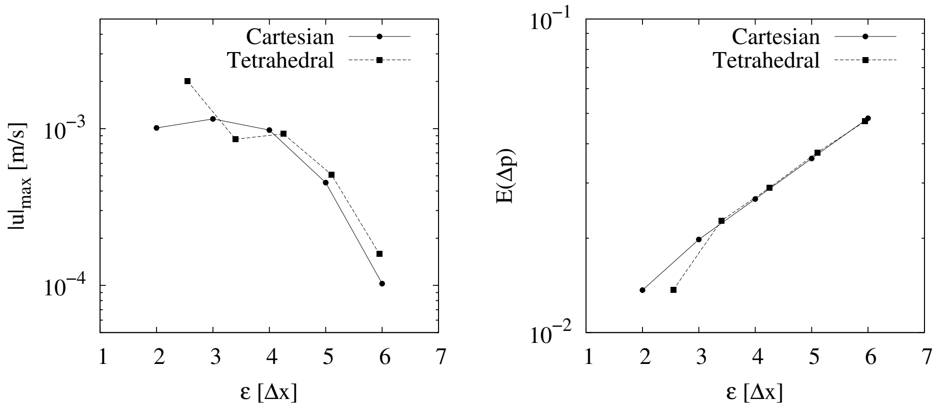


Figure 4.16.: Maximum parasitic currents magnitude $|\boldsymbol{u}|_{max}$ and pressure error $E(\Delta p)$ after one time-step as function of convolution length ϵ for the static inviscid drop in equilibrium used in Section 4.3.5 on an equidistant Cartesian (40^3 cells) and a tetrahedral ($\approx 6 \times 10^4$ cells) mesh.

As Gerlach *et al.* [76] pointed out correctly, a small convolution length ϵ may result in a noisy colour function field whereas a large convolution length may violate the physically feasible representation of the interface. Albadawi *et al.* [3] found steeply increasing errors in their simulations of detaching bubbles with increasing convolution length. Applying different convolution stencils to the rising bubble presented in Section 4.4.2 demonstrates why large convolution stencils are a poor choice for realistic applications of interfacial flows. The K_{cos} convolution kernel defined in Eq. 4.8 is applied to the rising bubble on the equidistant Cartesian mesh discussed in Section 4.4.2, using convolution lengths ranging from $\epsilon = 2\Delta x$ to $\epsilon = 8\Delta x$. The stencils for the evaluation of the interface normal vector ($l_{s,m} = 4\Delta x$) and the interface curvature evaluation ($l_{s,\kappa} = 2\Delta x$) remain unchanged. Density and viscosity are convoluted as described in Eqs. 4.41 and 4.42, whereas the surface force is calculated based on the unconvoluted colour function. The resulting bubble rise velocities of the four cases, illustrated in Figure 4.17, differ substantially. Examination of the bubble shape evolution, depicted in Figure 4.18, shows fundamental differences between the four cases. The results indicate that the different evolution of the bubble shape caused by the larger convolution length increases the drag of the bubble and, therefore, reduces the bubble rise velocity. Particularly, using a convolution stencil of $\epsilon = 8\Delta x$, the excessive convolution deteriorates the bubble in a significant and unphysical way, preventing the bubble to reach a terminal shape within the simulated time frame.

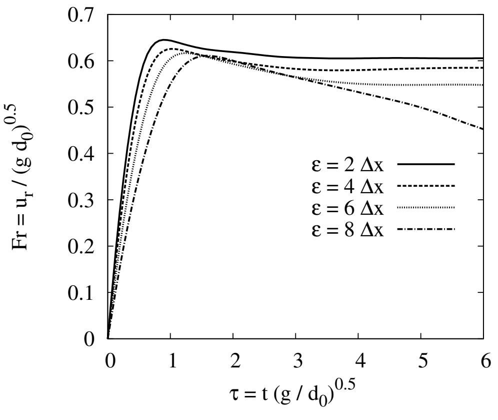
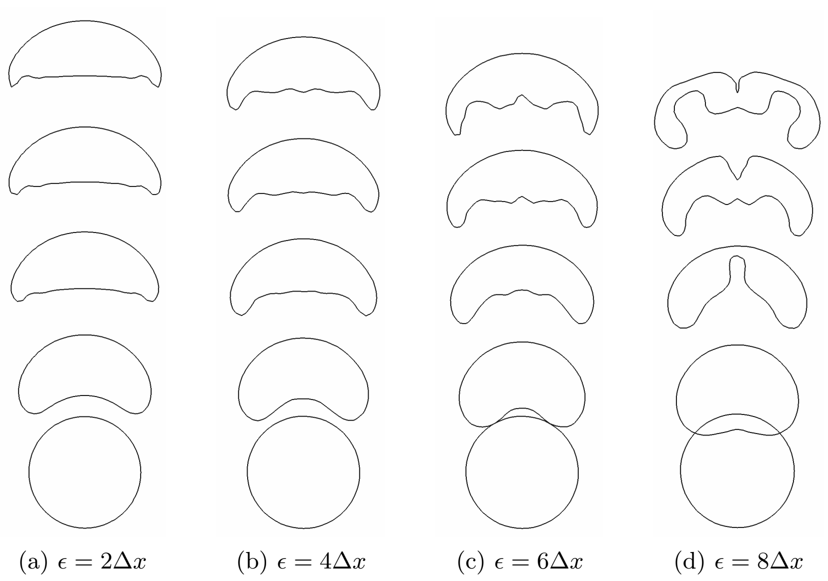


Figure 4.17.: Froude number of an initially spherical bubble rising due to buoyancy as a function of non-dimensional time $\tau = t\sqrt{g/d_0}$, applying different convolution lengths ϵ.



(a) $\epsilon = 2\Delta x$
(b) $\epsilon = 4\Delta x$
(c) $\epsilon = 6\Delta x$
(d) $\epsilon = 8\Delta x$

Figure 4.18.: Bubble shape evolution of an initially spherical bubble rising due to buoyancy, applying different convolution lengths ϵ. The bubble shapes are illustrated every $t = 0.07\,s$ ($\tau = 1.57$), starting from the initial position, in the x-y plane crossing through the centre of the domain.

4.6. Summary

In this chapter the numerical modelling of the interface and the evaluation of its geometry has been analysed and discussed, and a new method to evaluate the interface curvature has been presented. The new curvature evaluation method CELESTE is based on a second-order Taylor series expansion of the colour function and the resulting overdefined equation system is solved using a least-squares algorithm. CELESTE is applicable to both structured and unstructured meshes without difficulties and can be applied on computational stencils of various size. The curvature errors and parasitic currents obtained with CELESTE are as good as or better than the results presented in the literature deploying various other commonly used methods to approximate interface curvature directly or indirectly from the colour function, which are mostly limited to Cartesian meshes. Using CELESTE to evaluate the interface curvature, the results and performance on Cartesian and tetrahedral meshes are in very good agreement with each other.

The validation of the new curvature evaluation method CELESTE also included the comparison of three commonly used convolution kernels for the convolution of interface indicator functions in two-phase flows. Based on these results, the K_{cos} convolution kernel of Peskin [179] has been found to be best suited for the used numerical framework.

The test cases simulated in Sections 4.3.5 and 4.4 demonstrate that the application of the unconvoluted colour function, often described to be a discontinuous function, is not a problem in itself. In fact, every function is essentially discontinuous in a numerical framework due to the finite resolution. Thus, the unconvoluted colour function merely results in higher gradients and is more prone to errors upon differentiation than the convoluted colour function. At the other end of the spectrum, the numerical experiments presented in Section 4.5 show that the application of large convolution lengths adversely affect the outcome of the simulation.

Although the convolution of the colour function for the purpose of curvature evaluation has clear benefits if applied carefully, the convolution of fluid properties and surface force is a controversial issue. The presented test cases, comparing different convolution strategies with respect to density, viscosity and surface force, demonstrate the necessity of treating density and viscosity equally if large density ratios and viscosity ratios are applied. Furthermore, the test cases show that, contrary to the generally accepted notion, the surface force should not be convoluted as it increases parasitic currents and prohibits the accurate prediction of the pressure jump across the interface.

5. Adaptive Tetrahedral Mesh Refinement at Interfaces

The mesh resolution and quality are critical for the success of any numerical simulation and particularly important in CFD. The accuracy of the results, the convergence of the numerical solver as well as the computational requirements are directly dependent on the mesh. The mesh has to resolve relevant physical length scales occurring in the flow in order to perform physically sound numerical simulations and to minimise the numerical inaccuracies induced by the spatial discretisation. The mesh resolution plays a particularly important role with respect to two-phase flows and the interface separating the involved fluids. As previously mentioned in Sections 1.1 and 4.2.3, the interface position and the resulting forces acting at the interface (*i.e.* surface force and gravity) are only first-order accurate when using interface capturing methods [20, 61, 227], such as VOF or LS methods.

On a mesh of equal node density, such as an equidistant Cartesian mesh, resolution requirements derived from a particular flow feature, such as an interface, can lead to unaffordable computational requirements. In addition, the mesh resolution required to resolve all relevant flow structures is usually not known *a priori*. Adaptive mesh refinement (AMR) methods are frequently used to adapt the mesh in response to the flow, to overcome the problem of an accurate computational mesh of reasonable size. AMR methods aim at providing a fine mesh resolution in areas where small physical length scales occur or a high accuracy is desirable, and limit the mesh to a coarser resolution in areas which are not of great importance to the overall results or in which the physical length scales are considerably larger. Mesh adaption methods can be grouped into four major categories [107]:

1. *h-refinement*, where mesh nodes are inserted or removed and the mesh is reconnected accordingly,
2. *r-refinement*, where the mesh nodes are moved and the mesh connectivity remains the same,
3. *p-refinement*, where the local discretisation of the equations is adapted, and
4. hybrid methods, which are combinations of the above categories.

It is not guaranteed that an appropriate mesh resolution is achieved using r-refinement due to the fixed number of nodes. The adaption of the numerical discretisation, the so-called p-refinement, is especially interesting for finite element methods [107]. The h-refinement methodology is the most qualified and flexible approach for dynamic multiscale flow problems such as two-phase flows.

It is important to understand that a fluid-fluid interface represents a special case with respect to adaptive meshes. Information about the interface which is lost at any time during a simulation cannot be recovered at a later stage. Thus, it is crucial to maintain an adequate minimum mesh resolution in the vicinity of the interface at all times. In contrast, velocity information lost because of an inadequate mesh resolution can be interpolated or extrapolated from discrete neighbouring points with reasonable accuracy, due to its quasi-continuous distribution.

In this chapter, the following important issues regarding the application of unstructured adaptive meshes to two-phase flows are investigated:

- an implementation concept which is computationally efficient and can be easily extended to multi-processor computer systems,
- a suitable reference length scale or error measure to control the mesh resolution of adaptive tetrahedral meshes at interfaces,
- a methodology that satisfies continuity and preserves the conservation properties of compressive VOF methods, and
- accurate force-balancing at surface-tension-dominated interfaces on adaptive meshes.

Firstly, the present state-of-the-art concerning mesh adaption methods for two-phase flow simulations in general is reviewed in Section 5.1. Section 5.2 presents relevant methods to adapt tetrahedral meshes and Section 5.3 elaborates on the importance of mesh quality for fluid simulations using tetrahedral meshes. Subsequently, in Section 5.4, the implementation of a tetrahedral mesh adaption algorithm is presented. The presented algorithm is conceptually simple, computationally efficient and is readily extendable to multi-processor computer systems. In Section 5.5, the adequate interface resolution and related reference length scales for adaptive tetrahedral meshes are investigated. In Section 5.6, the advecting velocity evaluated at cell faces as derived in Section 3.3 is extended to adaptive meshes. The mesh properties, the conservation of flow properties and force-balancing are examined for a moving surface-tension-dominated interface on an adaptive tetrahedral mesh in Section 5.7. The findings of this chapter are summarised in Section 5.8.

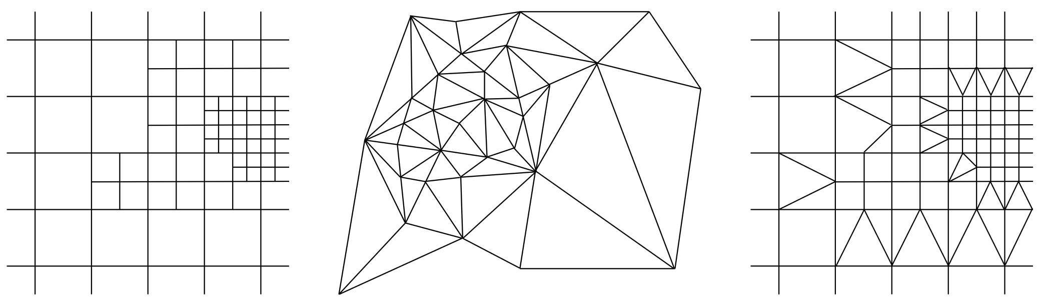
5.1. Adaptive Meshing and Interfaces

Even though two-phase flows are predicated for the application of AMR methods, the literature on this topic is relatively scarce. Most methods for two-phase flows are designed for Cartesian meshes, making their extension to adaptive meshes with different refinement levels difficult or in many instances impossible. On the other hand, adapting a structured mesh by redistributing its nodes (r-refinement), thereby abandoning a possibly existing initial Cartesian alignment, limits the mesh adaption substantially. Existing AMR methods developed for two-phase flows using interface capturing/tracking methods can generally be divided into three groups, illustrated for two-dimensional domains in Figure 5.1:

1. quadtree and octree meshes [83, 115, 139, 143, 183, 217, 226, 230, 255, 276],

2. adaptive triangular and tetrahedral meshes [28, 115, 275], and

3. adaptive hybrid meshes [47, 105].



(a) Quadtree mesh (b) Triangular mesh (c) Hybrid mesh

Figure 5.1.: Major refinement types of two-dimensional meshes used for two-phase flows.

The quadtree/octree refinement of typically Cartesian meshes has gained notable popularity in the two-phase flow research community over the past decade. The quadtree/octree refinement is conceptually easy to implement, the refinement is straightforward to control by its refinement levels and the preferable structured mesh arrangement is retained in most mesh areas. However, split cell faces are required to connect mesh cells of different size, as seen in Figure 5.1a. These split cell faces include so-called *hanging nodes*. A hanging node is defined as a mesh node on one side of the cell face but it is a point somewhere near the centre of the cell face for the opposing neighbour. Split cell faces require special treatment for the discretisation of face fluxes and pressure [83, 139]. Furthermore, the mesh is only able to change one refinement level per cell, prohibiting rapid resolution changes.

The limitations of most high-fidelity two-phase flow modelling methods to Cartesian meshes and the inherent complexity of adaption algorithms for unstructured meshes are the main reasons for the low popularity of adaptive triangular/tetrahedral meshes in the two-phase flow community. Although the numerical discretisation on stationary and adaptive triangular/tetrahedral meshes is identical, the mesh adaption algorithm is complex and not straightforward to implement. Furthermore, gaining a mesh of good quality is difficult and the control of the mesh adaption process is non-trivial, easily leading to underresolved or overresolved mesh regions.

Delage-Santacreu *et al.* [47] and Ito *et al.* [105] developed hybrid mesh refinement methods for two-dimensional meshes including triangular and quadrilateral cells, as illustrated in Figure 5.1c. In order to circumvent having hanging nodes between quadrilateral cells of different size, triangular cells are introduced to connect the different refinement levels. Tuković and Jasak [237] presented a moving mesh interface tracking method including an adaptive polyhedral mesh refinement. Most recently, Maric and co-workers [149, 150] proposed a geometrical VOF method applicable to arbitrary unstructured meshes, using the same mesh adaption framework as Tuković and Jasak [237]. Apart from that, the application of adaptive polyhedral meshes to two-phase flows has not been reported in the literature. Polyhedral meshes are equally flexible in terms of geometry as tetrahedral meshes but discretisation errors on polyhedral meshes are considerably smaller than on tetrahedral meshes [111]. However, similar to adaptive tetrahedral meshes, the implementation of polyhedral mesh adaption procedures is very complex.

Comparing quadtree/octree and unstructured triangular/tetrahedral meshes, the shape of the control volume (*i.e.* the mesh cell) is irrelevant for the application of source terms [112], such as surface force, since source terms are applied at cell centres and are not the sum of face values. Also, as a result of the abrupt, discontinuous pressure change, shocks do not benefit from face-pair error cancellation found in structured quadrilateral cells [112]. This is noteworthy in the context of two-phase flows because, from a numerical viewpoint, shocks are very similar to fluid-fluid interfaces with surface tension. Studies by Juretić [111] indicate a larger discretisation error for split-quadrilateral cells, found in quadtree meshes, than for triangular cells. On the other hand, Kim *et al.* [115] found no sizeable differences between the results of mold filling processes obtained on adaptive tetrahedral and octree meshes.

An important advantage of unstructured adaptive meshes is the possibility of a rapid resolution change, whereas quadtree/octree meshes are limited to a stepwise change of resolution. A rapid change of the resolution at an interface may save valuable computational resources but may also increase the errors induced by the numerical discretisation and may harm the stability of the numerical solver. The application of adaptive meshes with rapidly changing resolution has been successfully applied to capture shocks around aerofoils [41, 87, 127, 196]. Compere *et al.* [28] used strongly anisotropic adaptive tetrahedral meshes to model two-phase flows, though Compere *et al.* did not investigate the applicability of anisotropic meshes to surface-tension-dominated flows or two-phase flows with high density and viscosity ratios.

5.2. Methods for Tetrahedral Mesh Adaption

As tetrahedral meshes do not possess an inherent addressing structure, the adaption of tetrahedral meshes is very flexible. However, assuring a valid mesh of good quality is often difficult. This section provides an overview of the relevant methods for adaptive tetrahedral mesh refinement.

5.2.1. Delaunay Tetrahedralisation and Refinement

A *Delaunay triangulation* is a triangulation such that no mesh node is inside the circumcircle constituted by the three nodes of a given triangle, illustrated in Figure 5.2. The two-dimensional Delaunay triangulation extends straightforward to three dimensions, then called *Delaunay tetrahedralisation*. The Delaunay tetrahedralisation distributes the nodes such that no node is inside the circumsphere constituted by the four nodes of a given tetrahedron. In two-dimensions the Delaunay triangulation maximises the minimum angle between adjacent edges [43]. Although three-dimensional meshes generated by a Delaunay tetrahedralisation are generally of very good quality, dihedral angles are not maximised and the Delaunay tetrahedralisation fails to reliably identify so-called sliver cells [208], which are almost flat cells that fulfil the Delaunay condition. A derivative of the original Delaunay tetrahedralisation is the *constraint Delaunay tetrahedralisation* (CDT) [23, 216]. The CDT enforces additional requirements to the original Delaunay tetrahedralisation, such as a specific bounding geometry, so that the final tetrahedralisation might contain tetrahedrons which do not satisfy the Delaunay condition [214]. The Delaunay triangulation/tetrahedralisation and its derivatives represent the *de facto* standard for tetrahedral mesh generation and are applied in various meshing algorithms, most notably [60, 80, 82, 207, 211, 213, 268]. For a more detailed introduction to Delaunay tetrahedralisation in general and constraint Delaunay tetrahedralisation in particular, the interested reader is advised to refer to the works of Shewchuk [207] and Si [214].

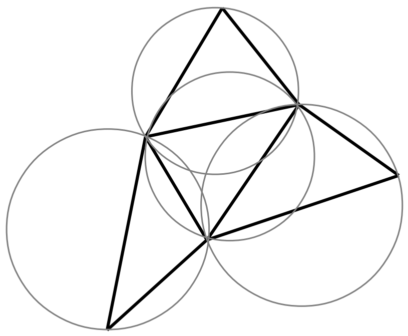


Figure 5.2.: Example of a triangular mesh which satisfies the Delaunay condition, including the circumcircles of the individual triangles.

The Delaunay method can also be used for the refinement of triangular and tetrahedral meshes without major algorithm changes [44, 45, 59, 122, 208]. Once an element is chosen for refinement, illustrated in Figure 5.3a, a new node is inserted at the centre of the circumcircle of a given triangle in two-dimensional meshes, shown in Figure 5.3b, or the centre of the circumsphere constituted by the four nodes of a given tetrahedron in three-dimensional meshes. Subsequently, the new node is connected to the nodes of the neighbouring elements, which are then replaced by the newly created elements, as shown in Figure 5.3c. The key challenge is to choose the best combination of elements to replace.

5.2.2. Edge Bisection

Edge bisection is a widely used refinement method for adaption algorithms of tetrahedral meshes [8, 24, 44, 80, 87, 181]. Edges which have to be refined are chosen according to an error estimate and new mesh nodes are introduced at the midpoints of these edges, as illustrated in Figure 5.4. The new nodes are connected to the neighbour nodes building new faces and elements. If the refinement is only applied to one cell adjacent to an edge, it leads to hanging nodes and generates an invalid mesh. Hence, in order to avoid the creation of hanging nodes and to ensure a valid mesh, all elements and faces adjacent to a split edge must be refined. The edge bisection method is relatively easy to implement and the number of new elements is bounded, since each redundant element is replaced by exactly two new elements.

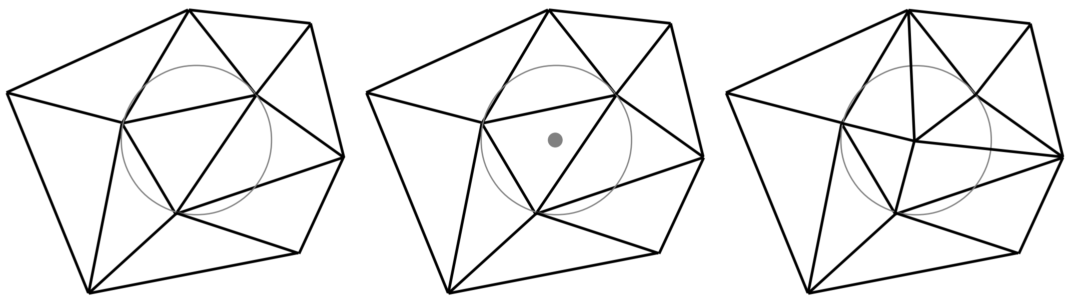
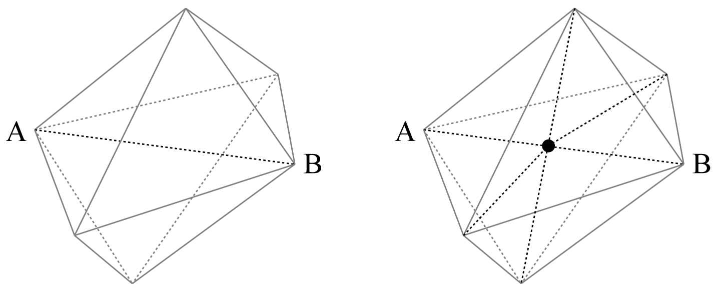

(a) Delaunay condition (b) Node insertion (c) Retriangulation

Figure 5.3.: Two-dimensional example of the Delaunay refinement. The circumcircle of a given element is constructed, as shown in (a), at the centre of which a new mesh node is inserted, see in (b). The new node is then reconnected with the neighbouring mesh nodes as illustrated in (c).



(a) Before edge bisection (b) After edge bisection

Figure 5.4.: Edge bisection of edge $\overline{AB}$ as part of an adaptive tetrahedral mesh refinement. A new mesh node is inserted at the centre of edge $\overline{AB}$ and reconnected with the neighbouring mesh nodes.

There are various algorithms to decide which edge of a given cell should be split [44, 273]. Using the longest edge of a given cell, called *longest-edge bisection* or *generalised bisection*, maximises the dihedral angles of the new elements, since the largest dihedral angle of the parent-element is split. In *non-generalised bisection* approaches, the refinement edge is chosen based on various other measures, for instance weighted error estimates for anisotropic mesh adaption [252]. The *newest vertex* approach, also called *newest node* approach, splits the edge opposite to the newest node of a given element. This method is straightforward to implement for two-dimensional meshes but is difficult to implement for three-dimensional meshes [273].

Numerous examples of the successful application of edge bisection methods to engineering problems have been reported in the literature. For instance, Sahni *et al.* [204] applied edge bisection for the adaptive meshing of boundary layers. An unstructured AMR algorithm with edge bisection was used for free surface flows by Dai and Schmidt [35], for aerodynamic simulation over a wing and in a gas turbine combustor by Tam *et al.* [229], for biomedical flow simulations and other applications by Li *et al.* [128] and for general interface problems by Wang *et al.* [252], to name just a few.

5.2.3. Edge Collapsing

Edge collapsing, also called *edge contraction*, illustrated in Figure 5.5, is a commonly used method to coarsen tetrahedral meshes and numerous application examples can be found in the literature [6, 10, 35, 44, 128, 204, 229].

An edge which is too short according to the local error measure or reference length scale is collapsed by merging its two constituting nodes. Thus, all elements and faces adjacent to the collapsed edge become redundant and are removed from the mesh. Positioning the mesh node resulting from edge collapsing at the midpoint of the collapsed edge is generally preferable [229]. Processing edges flagged for collapsing in order of their length, starting with the shortest edge, minimises the local impact on the mesh caused by edge collapsing [127].

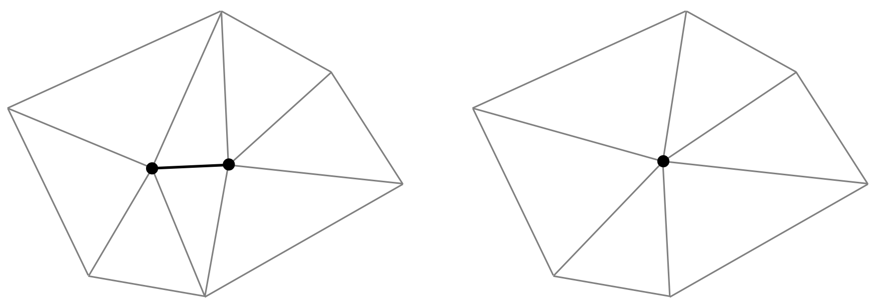

(a) Before edge collapsing (b) After edge collapsing

Figure 5.5.: Two-dimensional mesh coarsening using edge collapsing. The nodes constituting the collapsed edge (indicated in black) are merged and adjacent mesh entities are removed.

A major difficulty of the application of edge collapsing is to assure the collapsing procedure does not violate the validity of the mesh. For example, the integrity of the geometry is violated if a boundary node is moved. When edges are collapsed, elements in close vicinity may be inverted or unintentionally collapsed [35, 121], as depicted in Figure 5.6, resulting in elements with zero or negative volume.

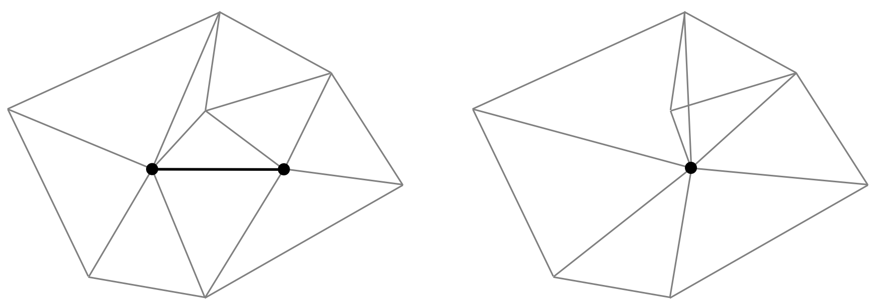

(a) Before edge collapsing (b) After edge collapsing

Figure 5.6.: Two-dimensional example of an invalid mesh resulting from edge collapsing.

5.2.4. Face and Edge Swapping

Mesh reconnection by means of *face swapping* and *edge swapping* is an important method to improve local and global mesh quality [74, 209]. In general, mesh reconnection means replacing mesh elements and faces by newly built elements and faces, using the same set of mesh nodes and occupying the same volume. Face swapping reconnects two tetrahedral elements adjacent to a single face, whereas edge swapping reconnects N tetrahedral elements adjacent to one common edge by removing the chosen edge and replacing the N elements by $2N - 4$ new elements [74].

The *2-3 flip* [74, 209], illustrated in Figure 5.7, deletes the shared face of the two tetrahedral elements and connects the two nodes which have not been adjacent to the deleted face. As a result, three new faces and three new elements are created. The reverse procedure is known as the *3-2 flip*. The *4-4 flip* [209], shown in Figure 5.8, replaces the common edge of four tetrahedral elements by its cross-sectional counterpart. The 2-3/3-2 flip as well as the 4-4 flip are easy to implement and very fast, since there are only two possible configurations which have to be compared with respect to mesh quality.

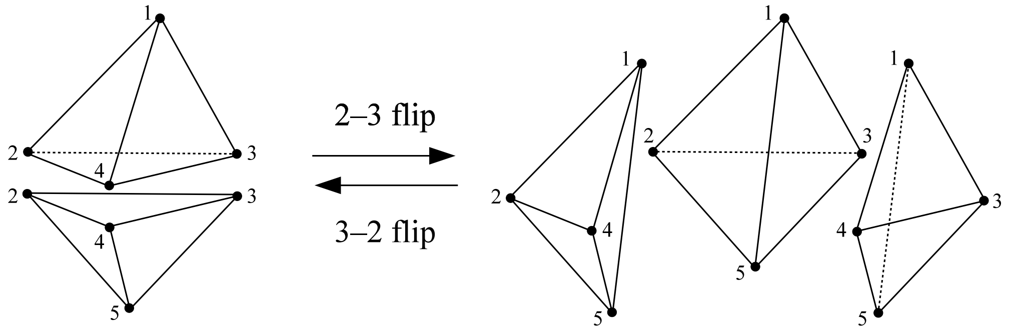


Figure 5.7.: Example of a *2-3/3-2 flip* tetrahedral mesh reconnection. The mesh face $\{2, 3, 4\}$ is replaced by faces $\{1, 2, 5\}$, $\{1, 3, 5\}$ and $\{1, 4, 5\}$.

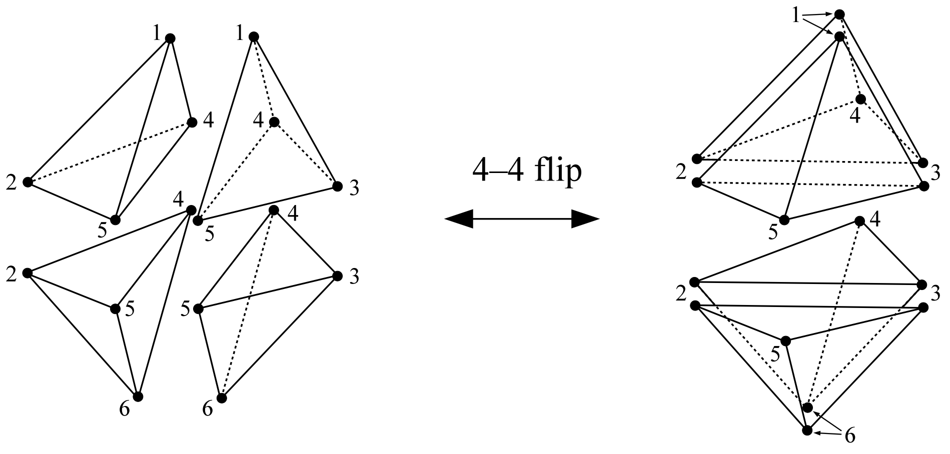


Figure 5.8.: Example of a *4-4 flip* tetrahedral mesh reconnection. The mesh edge $\{4, 5\}$ is replaced by edge $\{2, 3\}$.

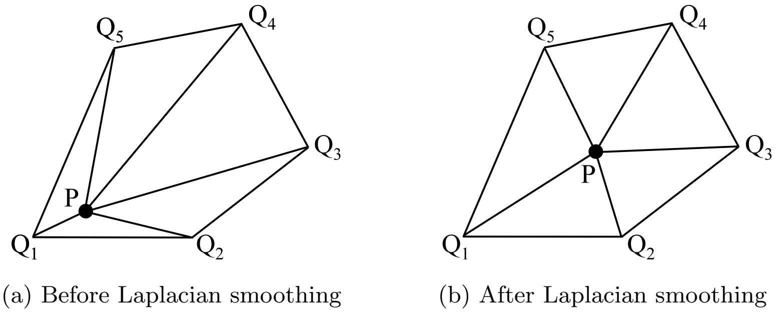
5.2.5. Node Redistribution

Redistributing mesh nodes, also referred to as r-refinement, is used to improve the mesh quality, to adapt the mesh to the flow field without changing the mesh topology and to avoid sudden resolution changes by smoothing the node distribution.

Laplacian smoothing is a frequently used mesh smoothing method [35, 69, 74, 178], since it is straightforward to implement and computationally efficient. Laplacian smoothing relocates the free mesh node P at the arithmetic centre of its neighbour nodes Q, as shown in Figure 5.9. The mesh nodes are relocated in an iterative procedure, defined as

$$\boldsymbol{x}_P^i = (1-\zeta)\,\boldsymbol{x}_P^{i-1} + \frac{\zeta}{N_Q}\sum_{Q=1}^{N_Q} \boldsymbol{x}_Q^{i-1} \ , \tag{5.1}$$

where superscript i represents the iteration and N_Q is the number of neighbours cells Q of cell P. The underrelaxation factor ζ in Eq. 5.1 is necessary to avoid inverted elements as a result of the Laplacian smoothing. With regards to tetrahedral meshes, Laplacian smoothing is not directly related to a good mesh quality, which is a substantial drawback of this method [35, 73].



(a) Before Laplacian smoothing (b) After Laplacian smoothing

Figure 5.9.: General principle of the Laplacian smoothing in two dimensions. Mesh node P is moved to the arithmetic centre of its neighbour nodes Q.

Anderson *et al.* [6] presented a node redistribution algorithm which minimises the energy of fictitious springs representing the mesh edges. The redistribution method removes sudden element volume changes and, according to the findings of Anderson *et al.* [6], results in a sufficient element quality. Each mesh edge is treated as a massless overdamped linear spring with tension

$$\sigma_e = l_e - l_{ref} \ , \tag{5.2}$$

where l_e is the edge length and l_{ref} is the target length scale. The redistribution method aims on reducing the equivalent of the global spring energy E_s (equivalent as the spring constant is not defined) of the mesh, defined as [6]

$$E_s = \sum_{e=1}^{N_e} \sigma_e^2 \ . \tag{5.3}$$

Based on the spring tension σ_e, the nodes are dynamically repositioned with velocity [6]

$$\dot{\boldsymbol{x}}_n = \sum_{e}^{N_e} \frac{\boldsymbol{x}_m - \boldsymbol{x}_n}{|\boldsymbol{x}_m - \boldsymbol{x}_n|} \sigma_e \ , \tag{5.4}$$

where subscript e denotes all edges connected to node n, subscript m denotes the second node constituting edge e and N_e represents the number of edges connected to node n. The redistribution algorithm terminates as soon as the spring tension reaches an equilibrium within a predefined tolerance. Anderson *et al.* [6] applied a Runge-Kutta solver with pseudo time-stepping to perform the dynamical displacement. In order to avoid the inversion of cells as a result of the node displacement, Anderson *et al.* limited the maximal movement per time-step and found that the equilibrium criterion is usually satisfied within 10 iterations. Similar algorithms have also been reported in other publications [30, 87, 165].

5.2.6. Other Methods

An alternative to edge collapsing for the coarsening of a previously refined mesh region is *de-refinement*. In a de-refinement procedure, the local mesh nodes introduced most recently are removed. Hence, de-refinement aims to restore a previous, coarser local mesh. De-refinement is very restrictive for two reasons: a) it is not possible to coarsen beyond the initial mesh and b) deletion of one node can have contingencies, *i.e.* requires the removal of other nodes to ensure a valid mesh. A two-dimensional example of a de-refinement displaying the issue of contingencies is illustrated in Figure 5.10. Integers 0 to 2 refer to the order in which nodes have been previously introduced as part of the mesh refinement, *e.g.* by means of an edge bisection algorithm. Removing the node of order 1 to de-refine the mesh also requires to remove the node of order 2 to retain a valid mesh.

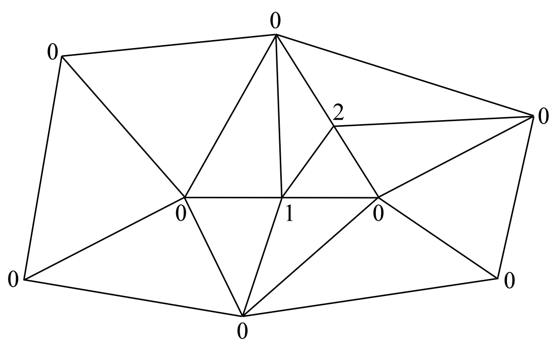


Figure 5.10.: Problems associated with contingencies of a mesh de-refinement algorithm. The integers denote the refinement level of each mesh node (0 = initial mesh). Removing the node of refinement level 1 also requires the node of refinement level 2 to be removed, otherwise the mesh becomes invalid.

Mesh adaption can also be performed by means of a *multigrid approach*, as presented by Plaza *et al.* [181]. A multigrid approach stores each mesh of each refinement step in memory. When a set of elements or a mesh region is chosen to be coarsened, the next coarser mesh representation for this mesh region is recalled from memory. This approach is computationally fast but memory expensive, as it is necessary to store multiple mesh representations. A major drawback is that it is not possible to coarsen past the initial mesh with this method.

5.3. Tetrahedral Element Shape and Quality

The quality of the computational mesh is of great importance to conduct accurate numerical simulations and is particularly interesting with respect to adaptive meshes. As briefly explained in Section 2.2.1, mesh errors resulting from a mesh with poor quality considerably deteriorate the accuracy of the discretisation and may severely challenge the stability of the numerical framework. Various measures to quantify the shape of a tetrahedral element and its quality can be found in the relevant literature [70, 74, 116, 131, 210]. Widely deemed to be the most universally applicable quality measures for a tetrahedral element are the dihedral angle of its cell faces [73], its aspect ratio [15] and its radius-edge ratio [214].

The *dihedral angle* is defined as the angle between two adjacent mesh faces. Small and large dihedral angles, *i.e.* close to $0°$ and $180°$, cause significant interpolation errors because of increased face skewness and face non-orthogonality. The evaluation of gradients for an element with very small or large dihedral angles is additionally challenged as the cell volume approaches zero, thus, making gradients unbounded. This is particularly problematic with respect to discontinuous variables with high gradients, such as the VOF colour function and pressure in two-phase flows with surface tension. The *aspect ratio* measures the "roundness" of the element and is either defined as the ratio between the longest edge and the shortest height of the tetrahedron [216], ranging from $\sqrt{2}/\sqrt{3}$ to $+\infty$, or as the ratio between 3 times the inradius and the circumradius, ranging from 1 to $+\infty$, also called *radius ratio*. The objective in isotropic meshes is to minimise the aspect ratio, as the reachable accuracy with a given element is inversely proportional to its longest edge [116]. The *radius-edge ratio* of a tetrahedron is defined as the ratio between the radius of its circumsphere and its shortest edge [216], ranging from $\sqrt{6}/4$ to $+\infty$. Research suggest that the nodes of a tetrahedral mesh are well spaced if the radius-edge ratio of the mesh is bounded for all its tetrahedral elements [214]. The radius-edge ratio reliably identifies tetrahedrons of bad quality apart from sliver cells, see example in Figure 5.11, which can have a radius-edge ratio as good as $\sqrt{2}/2$.

Typical shapes of tetrahedral elements are shown in Figure 5.11, a figure reprinted from Si [214, page 27, fig. 3.3]. The numerical stability of the computational simulation is dictated by the worst shaped element, rendering mesh quality essential. The optimal element shape for a tetrahedral mesh is the *regular tetrahedron*, constituted by four equilateral triangles. A regular tetrahedron has an optimal aspect ratio of $\sqrt{2}/\sqrt{3}$ based on its edge length and height, and all dihedral angles are $70.53°$. *Needles* occur commonly in areas with large resolution changes and are not by default harmful to the simulation outcome. *Sliver cells*, on the other hand, are an example for a particularly bad shaped tetrahedral element, causing major errors and instabilities. The smallest dihedral angles of a sliver cell are close to 0°, whereas its largest angles are typically close to 180°. The aspect ratio of sliver cells is large compared to most other tetrahedral element types and the volume of sliver cells is approaching zero. Most crucially, the resulting face skewness and face non-orthogonality are significant. Despite considerable research efforts [15, 122, 127, 128, 207, 210, 214], the reliable detection of sliver cells has proven to be difficult, since non of its edges is significantly longer than the others and because its radius-edge ratio is typically very good.

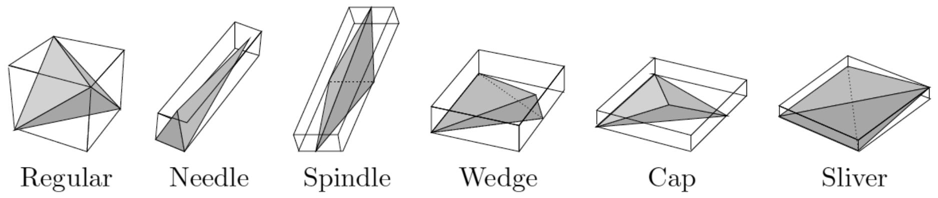


Figure 5.11.: Tetrahedral elements of different shapes as classified by Bern *et al.* [15]. Figure reprinted from Si [214, page 27, fig. 3.3].

The ultimate goal of maintaining a good mesh quality is minimising discretisation errors and, in return, maximising the accuracy of the discretisation as well as the stability of the solving algorithm. As previously mentioned, specific mesh errors influence the accuracy of the discretisation and the numerical stability. Most important with respect to the discretisation error are the face skewness and the face non-orthogonality, previously discussed in Section 2.2.1. The face skewness of a given mesh face f is defined as

$$\varphi = \frac{|\boldsymbol{r}_f|}{\Delta s} \ , \tag{5.5}$$

where $\boldsymbol{r}_f$ is the vector connecting the interpolation point of face f with the actual centre of face f, as defined in Section 2.2.2, and Δs is the distance between the cell centres adjacent to face f, as defined in Section 2.2.4. As previously discussed in Section 2.2.2, a gradient based skewness-correction is required to maintain a second-order accurate interpolation. Because the gradient based skewness-correction also has to be used in the evaluation of spatial gradient itself, interpolation errors caused by face skewness can propagate quickly and become self-amplifying and unbounded for severely skewed meshes. As a result, the numerical stability of the solving algorithm is fundamentally compromised and no convergent result can be reached. The non-orthogonality of a given mesh face is defined by the angle between its normal vector and the vector connecting the two adjacent cell faces and is, therefore, calculated as

$$\alpha^* = arccos\left(\boldsymbol{n}_f \cdot \boldsymbol{s}_f\right) \ , \tag{5.6}$$

where $\boldsymbol{n}_f$ is the outward-pointing unit normal vector of face f and $\boldsymbol{s}_f$ is the unit vector connecting the cell centres adjacent to face f. Minimising non-orthogonality is crucial for an accurate and stable interpolation of spatial gradients, as explained in Section 2.2.4. In an ideal mesh, such as a Cartesian mesh, face skewness and face non-orthogonality are $\varphi = 0$ and $\alpha^* = 0°$, respectively. Face skewness and face non-orthogonality are not strictly correlated with any shape measure of tetrahedral elements, since skewness and non-orthogonality primarily depend on the arrangement of elements in relation to each other. However, elements of good quality typically lead to a mesh with small skewness and non-orthogonality.

Minimising both face skewness and face non-orthogonality is essential to obtain accurate results and the minimisation of face skewness in particular is important to avoid unbounded gradients and to assure a robust numerical algorithm. Although tetrahedrons with large aspect ratio but good dihedral angles, such as the *needle* element depicted in Figure 5.11, are not desirable because of the potentially significant mesh non-uniformity and insufficient resolution associated with them, they are not critical to the stability of the numerical solving algorithm and can, therefore, be tolerated. Cells with very small or very large dihedral angles, on the other hand, such as *slivers* or *spindles*, are hazardous to the numerical stability and may lead to divergence of the solving algorithm. Thus, cells with dihedral angles $\vartheta \to 0°$ and $\vartheta \to 180°$ must be avoided. The adaption algorithm presented in Section 5.4 enforces limits on minimum and maximum dihedral angle explicitly by not accepting new mesh cells if they contain a dihedral angle smaller or larger than predefined limits. Preliminary studies performed during the course of the development of the adaption algorithm found $\vartheta = 13°$ and $\vartheta = 167°$ to be suitable lower and upper bounds to ensure a stable result. However, further research on dihedral angle limits is required to gain a deeper understanding of the underlying mechanisms.

5.4. Mesh Adaption Algorithm

In this section a mesh adaption algorithm for tetrahedral meshes and its implementation are presented. The mesh adaption algorithm first computes the reference length scale for each mesh cell. Subsequently, the mesh adaption algorithm, described in the following sections, is applied based on the local reference mesh resolution. After the mesh has been adapted, the solution stored on the old mesh is mapped to the new mesh, the data structures of the flow variables are reallocated and the new spatial gradients of the primitive variables and the colour function are determined. Lastly, the new advecting velocity at face centres is evaluated.

5.4.1. Setwise-Local Implementation Concept

A computationally efficient implementation is important for the applicability of a mesh adaption algorithm. The presented mesh adaption algorithm is implemented following a *setwise-local* implementation approach. As the name suggests, the setwise-local implementation limits any mesh adaption to a small subset of the local mesh, providing an efficient and flexible implementation. In this context, *local mesh* refers to the original mesh assigned to a given processor[8]. The basic concept of the setwise-local implementation is constituted by four distinct steps:

[8] *Processor* refers to a CPU and its associated memory

1. selecting a subset of the mesh, from hereon called *mesh-set*,
2. remesh the mesh-set with an appropriate adaption method,
3. establish internal connectivity of the new mesh-set, and
4. reintegrate the new mesh-set into the original mesh by updating the local connectivity.

The setwise-local principle can be used on single-processor as well as multi-processor computer systems and for any kind of remeshing operation, *e.g.* refinement, coarsening, reconnection or local retetrahedralisation.

The workflow is schematically illustrated in Figure 5.12 using the example of a three-dimensional edge bisection. A mesh-set is selected around the mesh edge flagged for refinement, containing the cells adjacent to that edge as well as the faces, edges and nodes constituting these cells. In the given example the mesh-set consists of four cells, four internal faces, eight bounding faces and the adjacent edges and nodes, as shown in Figure 5.12a. Subsequently, the edge bisection algorithm is applied to the mesh-set, inserting a node that splits the flagged edge, as illustrated in Figure 5.12b. The new edges, faces and elements are created and the internal connectivity is established. Lastly, the retetrahedralised mesh-set, shown in Figure 5.12c, is reintegrated in the original mesh by updating the connectivity between the new mesh-set and the original mesh. The old, now redundant mesh entities are flagged to be removed. The new mesh-set occupies exactly the same volume as was previously occupied by the old mesh-set. It is crucial that the mesh entities, *i.e.* nodes, edges and faces, which constitute the boundary of the mesh-set are not changed, otherwise the connectivity and validity of the resulting mesh is violated. The setwise-local implementation is applicable for any kind of mesh adaption and is not bounded to a specific size of the mesh-set. Crucially, the implementation principle remains unchanged regardless of the performed adaption procedure and is independent of the number of processors involved.

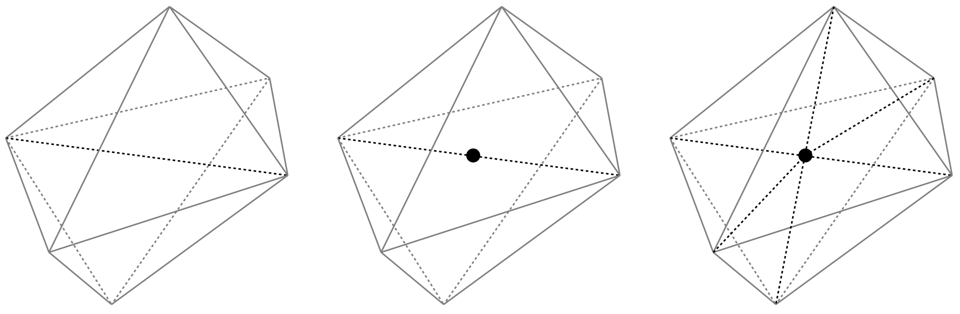
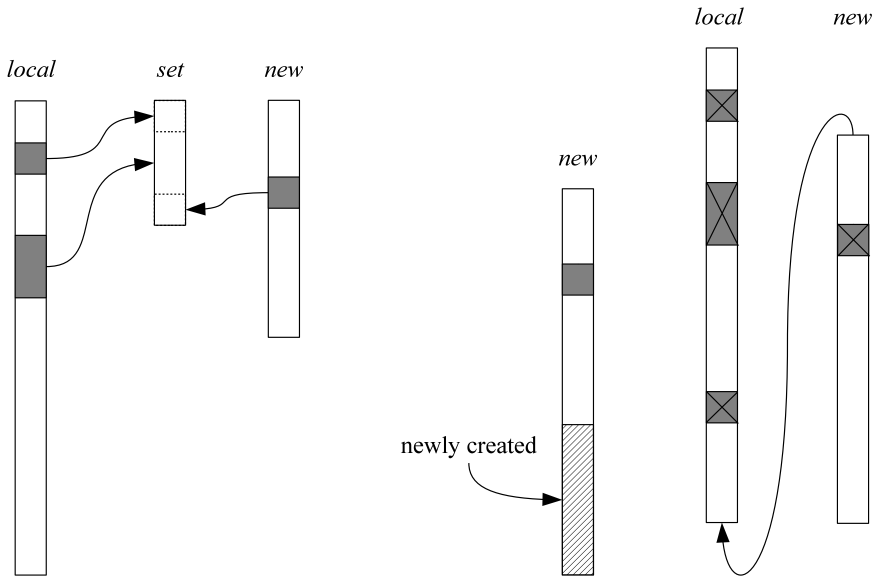

(a) Selected mesh-set (b) Node insertion (c) Retetrahedralised mesh-set

Figure 5.12.: Basic principle of the setwise-local implementation, exemplified by a three-dimensional edge-bisection.

The implementation is comprised by three separate data structures: the *local*, *set* and *new* data structures. The *local* data structure contains the original mesh on a given processor and remains fully intact during the adaption process. The only amendments made to the *local* data structure during the adaption process is flagging entities which become redundant and updating the connectivity information where applicable. The mesh-set selected for refinement or coarsening is stored in the *set* data structure. The set data structure only holds copies of relevant data from the *local* and *new* data structures, schematically shown in Figure 5.13a, and is overwritten after every adaption step with the present mesh-set. The *new* data structure contains mesh entities created by the mesh adaption algorithm, illustrated in Figure 5.13b. The remaining mesh entities of the *local* and *new* data structures are merged into a reallocated *local* data structure after the entire mesh has been adapted, as illustrated in Figure 5.13c, and the redundant mesh entities of both data structures are discarded. The separation of the three data structures provides high computational efficiency and simplifies the extension to a multi-processor computer architecture. Additionally, it is simple to control memory requirements, as the size of the *new* data structures and arrays can be fixed, and it is possible to readapt certain mesh areas multiple times without algorithm changes or performance loss.



(a) Mesh-set selection (b) Mesh adaption (c) Merging data structures

Figure 5.13.: Data structures of the setwise-local implementation and workflow of the mesh adaption procedure. The relevant information of the mesh-set selected for mesh adaption is copied from the *local* and *new* data structures to the *set* structure, as illustrated by the arrows in (a). New mesh entities created by the mesh adaption algorithm are appended to the *new* data structures, depicted in (b). As shown in (c), the *local* and *new* data structures are merged and redundant mesh entities, crossed out in (c), are deleted after the mesh adaption has been completed.

Two situations have to be considered when performing parallel, multi-processor simulations: a) all entities of the mesh-set are located on the same processor or b) the entities of the mesh-set are located on two or more processors. Situation a), where all entities are located on a single processor, does not require any special treatment and the implementation concept as described above can be applied. If the mesh entities of a given mesh-set are distributed over two or more processors, situation b), special measures are required to retain a valid and correctly connected mesh. The implementation across processor boundaries follows a *master-slave* principle. Firstly, it is determined on which processors the individual entities of the mesh-set are located. Among the involved processors, one processor is assigned to be the *master* and the other processors are labelled as *slaves*. The master is host to the *set* data structure, in which the mesh-set is stored, and to the mesh adaption operations. The slaves send the relevant data to the master and wait for the remeshing process to be completed. After the master has received the relevant mesh data from its slaves, the mesh adaption commences as a local procedure on the master processor. Subsequently, the internal connectivity of the new mesh-set is established. The newly created mesh data is communicated back to the slaves and the local connectivity is reestablished on the adjacent processors (*i.e.* the master and its slaves). Lastly, the ghost information is updated on the involved processors. Ghost entities are mesh entities which are located adjacent to a processor boundary on a neighbouring processor and are required for the numerical discretisation across processor boundaries (see Appendix A.2 for details).

When the mesh is adapted across processor boundaries, it is necessary to determine which processor will own the new mesh entities. Two rules may be put in place, either a) the new mesh entities all belong to the master and the connectivity information and ghost patterns are updated accordingly or b) the new mesh entities are distributed over the involved processor by means of a parentage rule. Communication between processors is considerably reduced if the master processor owns the new mesh entities, rule a), but it may lead to a substantial load imbalance, which then has to be resolved by a costly repartitioning of the mesh. Redistributing the new mesh entities back to the relevant processors, rule b), is desirable with respect to load balancing but increases the interprocessor communication. Furthermore, dependent on the performed mesh adaption operation, *e.g.* retetrahedralisation by means of a Delaunay tetrahedralisation, no unique assignment to a specific processor can be passed on from an old cell, requiring an overwriting processor assignment. Studies of Gross and Reusken [85] advocate storing the children on the same processor as their parent, *i.e.* rule b).

5.4.2. Error Measure

Applying adaptive meshes to any given numerical problem requires a suitable error measure, such as a reference length scale, to control the mesh adaption. Frequently used error measures are based on first or second derivatives of the primitive variables, the Kolmogorov length scale in turbulent flows or the length scale of other relevant flow features. Menzies [156] identified the following criteria for the definition of error measures:

- accurate reflection of the desired mesh resolution,
- reliably computable from the available data, and
- inexpensive to compute.

In two-phase flows, the interface is typically the dominating flow feature and its resolution is crucial for the accurate prediction of two-phase flows. Suitable error measures to define the mesh resolution in the vicinity of interfaces by means of reference length scales are examined in Section 5.5. The interested reader is referred to the papers of Habashi *et al.* [87] and Cristini *et al.* [30] or the thesis of Menzies [156] for an overview of general error measures used in mesh adaption algorithms.

The mesh adaption algorithm presented in this section uses error measures in the form of reference length scales to determine the mesh resolution. Since the flow variables in the presented numerical framework are stored at cell centres, calculating the reference length scale at cell centres suggests itself. The mesh adaption algorithm, however, is based on mesh edges rather than mesh cells. Thus, the cell-centred reference length scale has to be transferred to its adjacent edges. Enforcing the local minimal reference length scale strictly would require that every mesh edge is as long as its shortest adjacent cell-centred reference length scale, within a predefined tolerance. In terms of implementation, this requires a floating point comparison for each adjacent element of every mesh edge, a computationally expensive exercise for large meshes. To circumvent the evaluation of a large number of floating point comparisons while transferring the cell-centred reference length scale to mesh edges, it is proposed to transfer the reference length scale from cell centres to adjacent edges by means of a harmonic average, defined for edge e as

$$l_e = \frac{N_Q}{\sum_Q l_{ref,Q}^{-1}} \, , \tag{5.7}$$

where subscript Q denotes all cells adjacent to edge e, $l_{ref,Q}$ represents the reference length scale at cell Q and N_Q is the number of adjacent cells. The harmonic average emphasises small length scales and bypasses expensive floating point comparisons.

5.4.3. Mesh Coarsening

The applied mesh coarsening algorithm is based on a local retetrahedralisation using a setwise-local implementation. An edge identified to be too short compared to the local reference length scale is removed together with all its adjacent mesh entities, leaving only the empty hull of the mesh-set constituted by the bounding faces, edges and nodes. This empty hull is then filled by a new tetrahedralisation. The local retetrahedralisation has been chosen over edge collapsing to avoid problems with inverted or invalid cells and to increase the flexibility of the algorithm. A local retetrahedralisation can be applied to coarsen the mesh as well as to refine the mesh or improve its quality. The basic principle of the algorithm outlined below is, therefore, not limited to mesh coarsening.

A threshold is defined based on the local reference length scale, edges shorter than this threshold should be removed. The threshold follows from the local reference length scale of all adjacent elements and a factor representing the adaption tolerance. The adaption tolerance is necessary to avoid a substantial overlap of mesh coarsening and refinement, which could lead to a deadlock. The index and length of edges flagged to be removed are stored in a doubly-linked list, schematically illustrated in Figure 5.14, in the order of their length, starting with the shortest. A doubly-linked list is a chain of structures, where each structure is linked to the preceding and succeeding structures by a pointer containing the respective memory address. The linked list provides a storage format which can be dynamically changed while the mesh is adapted and, hence, the linked list enables to remove edges or move edges to a different position in the coarsening order. In the presented algorithm, always the first edge stored in the linked list at any given time through the coarsening procedure, called the *head*, is chosen to be removed. Therefore, the edges are reliably removed in order of their length, beginning with the shortest edge, as suggested by Li [127].

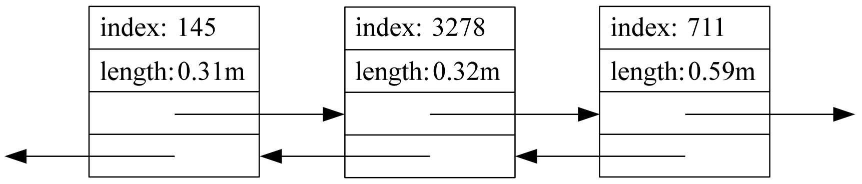


Figure 5.14.: Schematical illustration of mesh edges stored in a doubly-linked list for mesh coarsening.

The coarsening algorithm, applied to every flagged edge, is constituted by the following eight steps:

1. define and copy the mesh-set around the flagged edge,
2. translate the relevant data of the mesh-set into a form suitable for the retetrahedralisation algorithm,
3. perform the retetrahedralisation,
4. define the new internal mesh entities of the mesh-set,
5. establish the internal connectivity of the new mesh-set,
6. reconnect the new mesh-set with the original mesh,
7. map the flow data from the old mesh-set to the new mesh-set, as described in Section 5.4.6, and
8. update the list of flagged edges.

The open-source software *TetGen* [214, 215] is used as part of the presented algorithm to retetrahedralise a given mesh-set. TetGen is able to generate tetrahedral meshes for any three-dimensional polyhedral domain using a CDT algorithm and can be integrated into existing software projects as an external library. The tetrahedralisation is controlled by means of input flags upon calling TetGen with a given set of faces and nodes, which represent the empty hull defined by the bounding faces of the mesh-set. It it worth mentioning again that it is absolutely essential that the hull of the mesh-set, *i.e.* the bounding faces of the mesh-set, is not changed or split by the retetrahedralisation to avoid the creation of hanging nodes and to ensure a valid new mesh.

After all edges flagged for coarsening and its adjacent mesh entities have been replaced, the *local* and *new* data structures are merged, the obsolete mesh entities are discarded and the indexing (see Appendix A.1 for details) of the mesh entities is updated.

5.4.4. Mesh Refinement

The mesh refinement is based on a longest-edge bisection approach, as presented in Section 5.2.2. Similar to the threshold defined for mesh coarsening in Section 5.4.3, a threshold is defined for the mesh refinement based on the local reference length scale and a factor representing the adaption tolerance. The index and length of the flagged edges are then stored in a linked list as illustrated in Figure 5.14 in order of descending edge length and the edge stored at the *head* of the list is processed by the longest-edge bisection algorithm.

The longest-edge bisection algorithm applied to every flagged edge is constituted by the following nine steps:

1. define and copy the mesh-set adjacent to the flagged edge (see Figure 5.12a),
2. introduce a new mesh node at the centre of the flagged edge (see Figure 5.12b),
3. connect the new node to the other nodes of the mesh-set (build new edges),
4. define the new faces adjacent to the new node,
5. define the new elements inside the mesh-set based on their host face (bounding face of the mesh-set),
6. establish the internal connectivity of the new mesh-set,
7. reconnect the new mesh-set with the original mesh by updating the relevant connectivity information,
8. map the flow data from the old mesh-set to the new mesh-set, as described in Section 5.4.6, and
9. update the list of flagged edges.

In addition, before proceeding to the next edge flagged for refinement, the algorithm identifies any new edges which do not satisfy the local reference length scale. New edges which do not satisfy the refinement threshold are flagged for refinement and added to the linked list holding the flagged edges. It is worth mentioning again that all cells adjacent to the edge flagged for refinement have to be processed to ensure that no hanging nodes are created.

The update of the edge-adjacent element information is treated in a special way, as the number of elements adjacent to an edge may change during the refinement process. Therefore, the connectivity information of edge-adjacent element information is only updated once, after all edges have been processed, not after every individual edge bisection. This minimises the time intensive array reallocation and updating of the edge-adjacent element information. Instead, the old connectivity information is kept and, if necessary, the actually edge-adjacent elements are determined by a recursive algorithm, marching from descendant to descendant to the finest local mesh level.

Similar to the mesh coarsening algorithm presented in Section 5.4.3, the *local* and *new* data structures are merged after all edges flagged for refinement have been processed. Furthermore, the obsolete mesh entities are discarded, the edge-adjacent element connectivity is updated and the indexing of all mesh entities is updated. Details about connectivity information and indexing can be found in Appendix A.1.

5.4.5. Initialisation

The initialisation step is particularly important for interfacial flows to assure an adequate mesh resolution and, thus, an accurate simulation. Although velocity information lost in the adaption process can be retrieved with good accuracy from the velocity data and its gradients at neighbour cells, interface information lost during the mesh adaption process is lost permanently. Therefore, the result of a two-phase flow computation can only be as accurate as the initial interface representation.

The initialisation of the flow field is performed iteratively in conjunction with the mesh adaption algorithm. At first, the relevant flow variables, *i.e.* velocity, pressure and colour function, are initialised and the reference length scale is evaluated. Subsequently, the mesh is adapted according to the reference length scale. The relevant flow variables are then reinitialised on the new mesh, the new reference length scale for each cell is computed and the mesh is adapted again. This procedure continues until the mesh has converged to a steady solution or until a predefined number of iterations is reached. Figure 5.15 shows an example of the iterative initialisation procedure for a spherical interface, including the initial mesh in Figure 5.15a, an intermediate mesh in Figure 5.15b and the final mesh after five iterations in Figure 5.15c.

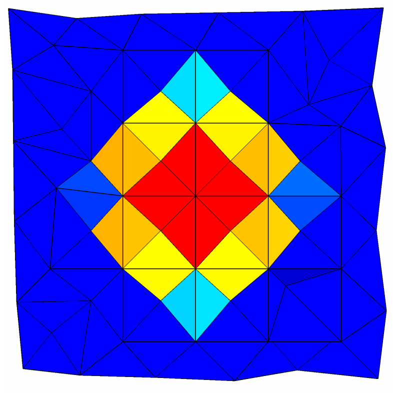
5.4.6. Solution Mapping Between Meshes

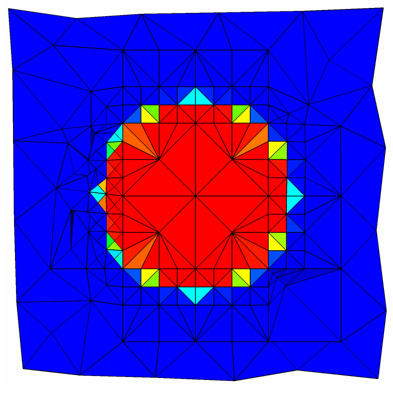
The mapping of the solution obtained on the old mesh to new and moved mesh cells is important to assure an efficient and accurate application of mesh adaption algorithms and to maximise the accuracy of the results. The aim is to obtain a mapping methodology which is accurate, preserves the governing conservation laws, in particular continuity, and assures the balance between body forces and pressure gradient.

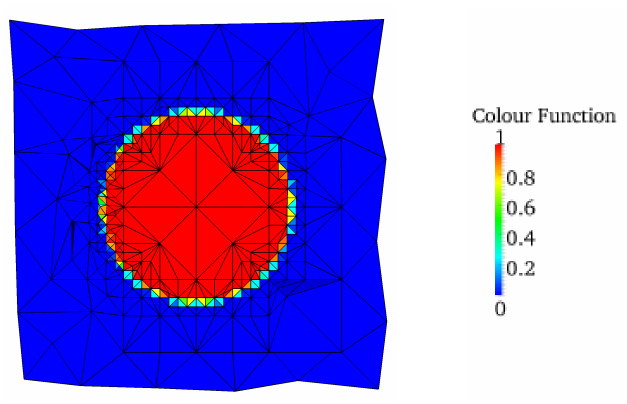
For continuously varying variables, such as velocity, a piecewise-linear mapping is applied by means of extrapolating the variable from the parent cell to its children cells. In general, the parent is defined as the cell of the old mesh closest to the new cell under consideration. Following Jasak and Gosman [107] and Juretić [111], the value of a flow variable at the cell centre of the new cell can be extrapolated as

$$\phi^n = \phi^p + (\boldsymbol{x}^n - \boldsymbol{x}^p) \cdot \nabla \phi^p \ , \tag{5.8}$$

where superscript n denotes the new cell and superscript p denotes the parent cell. The position vector of the new and parent cell centres are represented by $\boldsymbol{x}^n$ and $\boldsymbol{x}^p$, respec-


(a) Initial mesh


(b) After two iterations of initialisation



(c) After five iterations of initialisation

Figure 5.15.: Example of the iterative initialisation procedure for a spherical interface. The initial mesh, shown in (a), is refined based on the initialised flow field and after the mesh has been adapted, the flow field is reinitialised. This iterative procedure continues until a predefined convergence criteria is reached.

tively.

A different treatment is required to map discontinuous variables, such as the colour function or the pressure at interfaces, because, firstly, the exact position of the discontinuity is not known and, secondly, the large, abruptly changing spatial gradient of the discontinuous variable may lead to an unbounded solution upon extrapolation. Using the mapping of the interface position (*i.e.* the discontinuity of the colour function) after mesh refinement as an example, it is neither explicitly known in which of the new cells the interface is located nor is the magnitude of the colour function in each new cell known. Furthermore, the distribution of the colour function is known with only first-order accuracy. Therefore, a piecewise-constant mapping of the colour function, defined as

$$\gamma^n = \gamma^p \ , \tag{5.9}$$

is applied. Since the colour function is known with only first-order accuracy regardless of the applied mapping method, a piecewise-constant mapping does not decrease the formal order of accuracy of the interface position. However, it is unlikely that this mapping is strictly conservative, although the results regarding the conservation of continuity and mass presented in Section 5.7.2 are excellent.

With respect to isothermal, incompressible two-phase flows and the numerical framework presented in Chapter 3, mapping the velocity and the colour function to the new mesh is sufficient. The pressure distribution as well as the density and viscosity distribution can be calculated from the velocity and colour function data in conjunction with boundary conditions. However, mapping the pressure to the adapted mesh as well improves the convergence and stability of the numerical solver substantially, as it serves as an initial guess for the iterative solving procedure. To avoid an unbounded pressure field near the interface, the pressure is mapped following Eq. 5.9.

5.5. Interface Resolution

Despite considerable research efforts directed towards the application of mesh adaption methods to two-phase flows in the past two decades, the literature on suitable reference length scales is scarce. The focus on quadtree/octree meshes in the two-phase flow community is presumably one reason why well-defined reference length scales have been discounted in many studies on mesh adaption methods. The control of octree meshes is considerably simpler compared to adaptive tetrahedral or polyhedral meshes, as the number of refinement levels is easily specified. Unstructured adaptive meshes, however, require a sophisticated definition of error measures or reference length scales, in order to resolve flow features adequately while minimising computation time. Another reason why well-defined error measures have not caught notable attention with respect to interfacial flows might be the common notion of *"the more cells the merrier"*. In principle this is, of course, true because of the discontinuous nature of the interface but it does not maximise the potential of mesh adaption. It is worth recalling at this point that any interface information lost after initialisation cannot be recovered. Thus, the reference length scale applied in the vicinity of the interface must assure an adequate mesh resolution throughout the simulation and the interface must not be directly affected by the applied mesh adaption procedure after initialisation. Moreover, the reference length scale must assure that the total number of cells is reasonable with respect to the available computational resources.

In the following sections, the resolution at interfaces is examined and a set of suitable reference length scales is defined. The influence of the different parameters and length scales is demonstrated with a spherical interface in a cubical domain, illustrated in Figure 5.16a. The initial mesh contains approximately 1500 cells and is depicted in Figure 5.16b. Only the initialisation stage of the simulation is conducted, as described in Section 5.4.5.

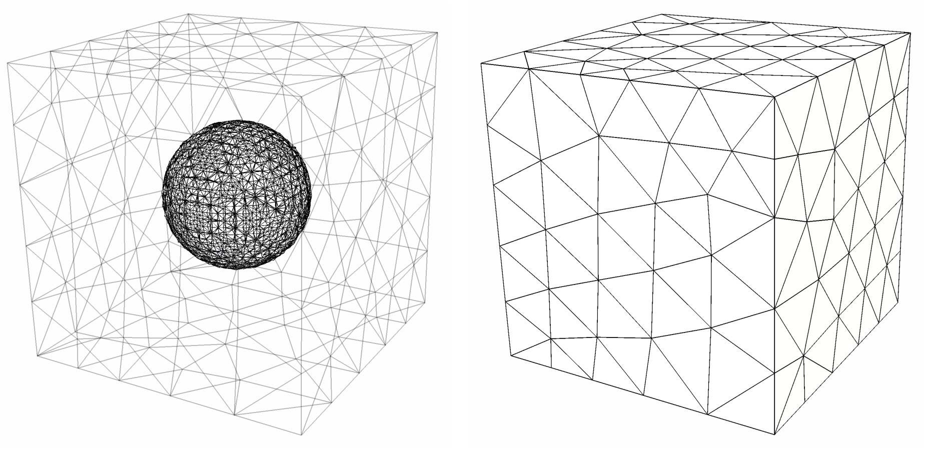

(a) Domain with spherical interface (b) Initial mesh

Figure 5.16.: Computational domain including the spherical interface and its initial tetrahedral mesh.

5.5.1. Defining the Interface Region

A minimum resolution in the interface region regardless of the shape of the interface is important to keep the interface sharp and, with respect to adaptive meshes, to optimise the application of mesh adaption procedures [28]. Various ways of capturing the interface and defining an interface region for the purpose of determining reference length scales for adaptive meshing are available. The aim is to define a band of finite width around the interface in which a minimum mesh resolution is guaranteed and in which the mesh size is defined based on interface properties, such as interface curvature or interface thickness. Moreover, an interface region of adequate size can be used to assure that the interface is always resolved by a required minimum mesh resolution, in order to avoid the loss of interface information. The width of the interface region is also important with respect to the frequency with which the mesh adaption is applied. Mesh adaption has to be applied more frequent to interface regions of smaller width than to interface regions of larger width.

Otsu [170] presented a method to binarise a scalar field by defining an optimal threshold that separates two reference values by maximising the variance between those values. Otsu's method has initially been developed to binarise gray-level histograms in picture processing. Remaki and Habashi [196] successfully applied Otsu's threshold binarisation method to capture weak shocks (typically with Mach number > 1 behind the shock) by the adaptive mesh refinements, which would otherwise be underresolved in instances where they occur together with strong shocks (Mach number behind shock < 1). However, since interfaces practically do not have different "strength", the weak-strong problematic does not occur in incompressible interfacial flows.

Alternatively, a band defining the interface region can be constructed based on the distance from an interface. Compere *et al.* [28], for instance, called such a band *proximity zone* for their level-set framework with adaptive meshes. The difficulty of this approach lies in finding the closest interface-cell for every given cell. Thus, this approach requires considerable computational resources for large meshes and if multiple interfaces are present in the domain. Herrmann [92] constructed a band around the interface as part of a localised level-set method by means of a band growth algorithm. The band growth algorithm constructs the band by marching from neighbour to neighbour, starting from each cell containing an interface, and terminates when no more cells within the width of the band are found.

Defining a band around the interface based on the first derivative of the colour function is a straightforward and computationally inexpensive approach. Because the colour function is by definition abruptly varying at the interface, the magnitude of its first derivative is non-zero only in the cells containing an interface and in its direct neighbours. Thus, it is proposed to define the interface band as the region in which the magnitude of the first derivative of a reference colour function γ_{ref} is larger than a specified threshold. The reference colour function can be the unconvoluted colour function or a convoluted colour function, whichever deemed to be most suitable for the definition of the interface region. Applying a convoluted colour function increases the number of cells with a non-zero gradient magnitude and, therefore, allows for a wider and better resolved interface band. Thus, any cell P that satisfies the condition

$$|\nabla \gamma_{ref}|_P \geq C_{\nabla\gamma} \cdot |\nabla \gamma_{ref}|_{max} \ , \tag{5.10}$$

where $C_{\nabla\gamma}$ is a predefined constant defining the width of the band, is part of the interface band. In Eq. 5.10, $|\nabla \gamma_{ref}|_{max}$ represents the maximum discrete magnitude of the reference colour function gradient in the entire domain. $C_{\nabla\gamma}$ is defined within the range $0 \leq C_{\nabla\gamma} \leq 1$, where $C_{\nabla\gamma} = 0$ classifies the entire domain as interface region and $C_{\nabla\gamma} = 1$ would only include the cell(s) holding the maximum colour function gradient magnitude in the interface region. The extension to parallel computer systems and unstructured meshes is straightforward.

Figure 5.17 shows the resulting mesh using two different interface region widths, $C_{\nabla\gamma} = 10^{-3}$ and $C_{\nabla\gamma} = 0.2$. The adaptive refinement in this example is limited to five iterations of initialisation, as described in Section 5.4.5, and the mesh resolution is determined based on the interface curvature and the interface thickness, discussed in Sections 5.5.2 and 5.5.3. As expected, the larger constant $C_{\nabla\gamma}$ results in a thinner interface region than the smaller constant. The relative volume error with respect to the geometrically exact volume of the spherical interface in both cases is approximately 0.03%, proving that the volume error is not dependent on the band width. Since the interface thickness is only one cell, the volume error should not depend on the thickness of the interface region.

5.5.2. Resolution of Interface Curvature

An adequate resolution of the interface curvature, *i.e.* the resolution tangential to the interface, is vital for accurate interfacial flow simulations, as an underresolved interface curvature results in a deficient local surface force. As discussed in Section 4.2.3, various

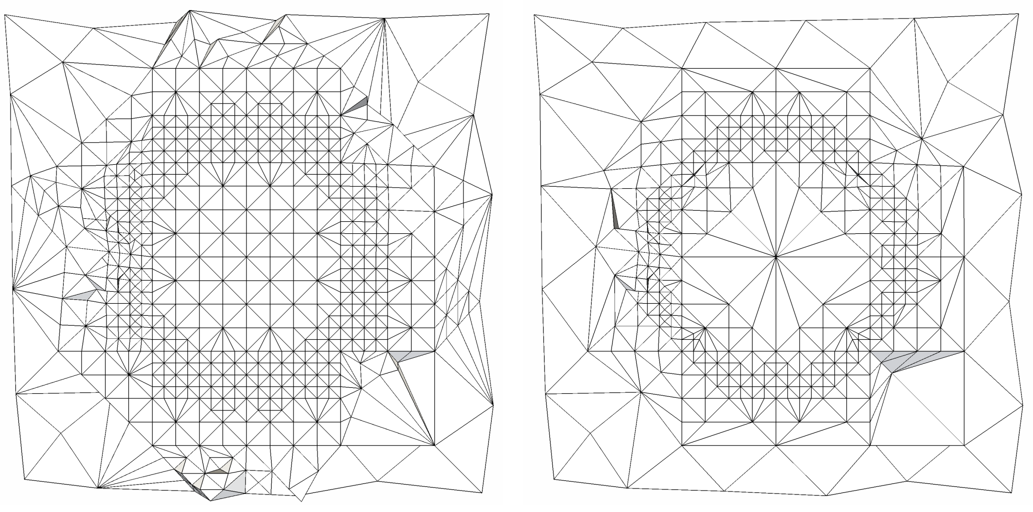

(a) $C_{\nabla\gamma} = 10^{-3}$ (b) $C_{\nabla\gamma} = 0.2$

Figure 5.17.: Tetrahedral mesh refined in the interface region based on the condition defined in Eq. 5.10. The figures show the mesh cells with an x-coordinate of the cell-centre larger than half the domain size.

recommendations for a suitable curvature resolution have been made in previous publications. Cristini *et al.* [30] proposed to use a harmonic average to define a reference length scale based on the local minimum and maximum curvature as

$$l_\kappa = \sqrt{\frac{2}{\kappa_{min}^2 + \kappa_{max}^2}} \ , \tag{5.11}$$

whereas others, most notably [19, 143, 183, 194, 275], defined the curvature length scale to be

$$l_\kappa \propto \frac{1}{\kappa} \ . \tag{5.12}$$

The evaluation of the length scale following Cristini *et al.* [30], given in Eq. 5.11, includes a considerable number of floating point comparisons to find the local minimum and maximum curvatures. Thus, the length scale definition given in Eq. 5.12 is preferable from a computational standpoint.

The reference length scale derived from the interface curvature used in the presented algorithm follows the relationship defined in Eq. 5.12 and is given as

$$l_\kappa = \frac{1}{C_\kappa \, \kappa} \ , \tag{5.13}$$

where C_κ is a predefined constant defining the mesh resolution and its bias towards interface curvature. Results of Raessi *et al.* [194] indicate useful results for a curvature resolution of

$$l_\kappa \leq \frac{1}{3\kappa} \ . \tag{5.14}$$

Malik *et al.* [143] found nearly curvature independent solutions for a variety of interface geometries for a curvature resolution of

$$l_\kappa \lessapprox \frac{1}{20\kappa} \ . \tag{5.15}$$

Figure 5.18 shows meshes for a spherical interface obtained with different values of C_κ, applied in an interface region defined by Eq. 5.10 with $C_{\nabla\kappa} = 0.1$. Figure 5.19 shows the relative volume error of the spherical interface as a function of C_κ. The volume error successively decreases for larger curvature coefficients C_κ as a result of the resulting higher mesh resolution. The qualitative assessment of Figure 5.18 indicates that C_κ should not be smaller than 3 to assure a decent representation of the curvature.

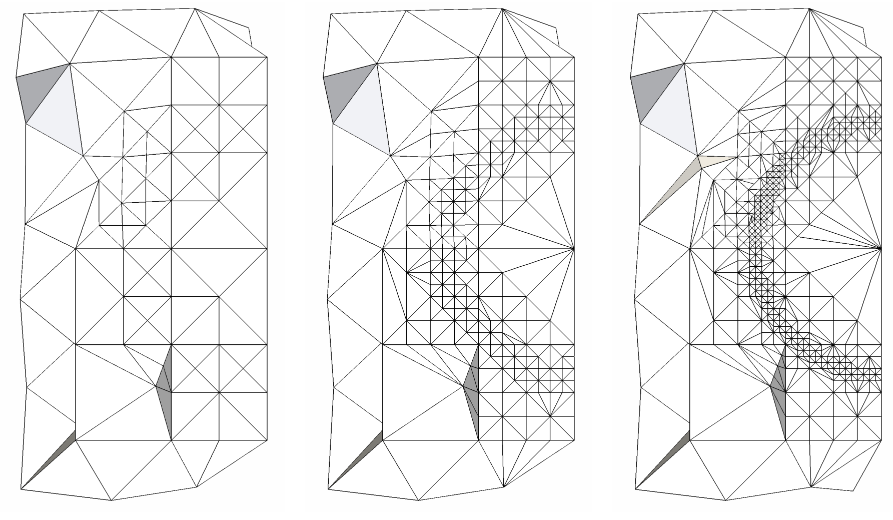

(a) $C_\kappa = 1$ (b) $C_\kappa = 3$ (c) $C_\kappa = 8$

Figure 5.18.: Tetrahedral mesh refined based on the interface curvature of a spherical interface for different values of C_κ, as defined in Eq. 5.13. The figures show the mesh cells with an x-coordinate of the cell-centre larger than half the domain size.

For the above length scale definitions as well as for the recommendations reviewed in Section 4.2.3 the mesh resolution refers to the distance between computational nodes, *i.e.* the distance between cell centres. On unstructured meshes the distance between cell centres is, however, not equal to the edge length. For two adjacent regular tetrahedrons, the distance of its cell centres is twice the insphere radius and is given as

$$d^* = \frac{l_e}{\sqrt{6}} \ , \tag{5.16}$$

where l_e is the edge length of the regular tetrahedrons. Hence, assuming a mesh built of regular tetrahedrons, a reference length scale based on the interface curvature defined with $C_\kappa = 1$ represents a mesh spacing in the interface region of $d^* \approx 5/2\kappa$ and $C_\kappa = 8$ corresponds to $d^* \approx 2/39\kappa$. Thus, assuming the cells are regular tetrahedrons, the interface curvature resolution recommended by Raessi *et al.* [194] can theoretically be achieved with $C_\kappa = 1.225$ and the recommendation of Malik *et al.* [143] with $C_\kappa = 8.165$. This must be

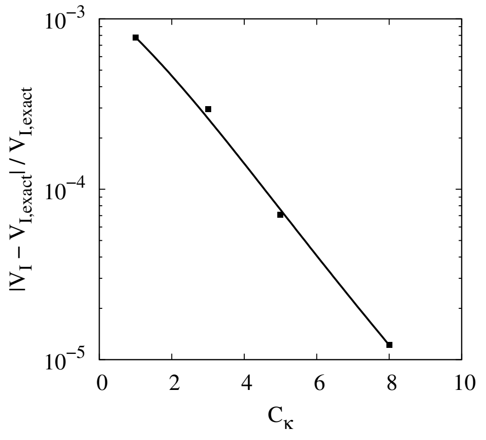


Figure 5.19.: Relative volume error of the spherical interface on meshes adapted with different values of C_κ, as defined in Eq. 5.13.

kept in mind when deciding on an adequate reference length scale definition.

5.5.3. Interface Thickness

In reality the interface between to immiscible fluids is infinitesimally thin with respect to continuum mechanics and, thus, strictly speaking, cannot be resolved within a finite volume framework. Instead, the interface is represented with a finite thickness. Following the results of the study of different convolution strategies presented in Section 4.4, the fluid properties are distributed smoothly across the interface, *i.e.* convoluted, whereas the surface force is calculated based on the unconvoluted colour function. Therefore, the interface thickness affects the surface force and the resulting pressure jump across the interface directly. Treating the interface thickness separate from the interface curvature for the purpose of defining an adequate mesh resolution is necessary for two reasons. Firstly, defining the interface resolution solely based on the interface curvature and assuming the interface has a thickness of one mesh cell results in a thinner interface in regions of high curvature than in regions where the local curvature is small. Secondly, determining the mesh resolution based on the interface thickness assures that interface movements are captured accurately even if $\kappa \to 0$, *i.e.* for a completely flat interface.

The first derivative of the colour function, which is readily available, seems to provide a meaningful indication for the mesh resolution at interfaces, because the first derivative of the colour function is non-zero only at interfaces and by definition spatially coincides with the surface force (see Eq. 2.48). Darwish *et al.* [41] published results applying a gradient-based indicator for adaptive mesh refinement to capture shocks, which are numerically similar to fluid-fluid interfaces, in supersonic flows. However, a high first derivative itself does not *per se* require a high mesh resolution and first derivatives only provide efficient indication for the mesh resolution if the variation of the quantity under consideration is considerably non-linear. For this reason, previous studies concerned with general error measures for mesh adaption proved the second derivative to be a more reliable and efficient indicator for the mesh resolution [87, 196, 228, 229]. For instance, Tam [228] defined an error estimate from the second derivative of a given flow quantity, where edge length l_e has to satisfy

$$l_e^2 \left| \frac{\partial^2 \phi}{\partial x^2} \right|_e = C_l \ , \tag{5.17}$$

with C_l representing a predefined coefficient. Other methods to control mesh adaption include the equidistribution principle [63, 156] and the spring analogy [6, 87, 156, 165]. Nakahashi and Deiwert [165] recommend a spring analogy in form of a tension spring to determine the mesh resolution and a torsion spring to define the face skewness.

However, because the interface, represented implicitly by the colour function, should retain a thickness of one mesh cell, applying one of the error measures explained above effectively creates a deadlock. In theory, the refinement would continue with an ever decreasing interface thickness until, eventually, the interface thickness approaches zero and the number of mesh cells becomes infinite. Because a gradient-based indicator can only initiate mesh adaption but not terminate it for the specific case of a fluid-fluid interface, an upper bound l_{max} and a lower bound l_{min} for the interface thickness have to be determined. Defining explicit bounds assures a reasonably thin interface and guarantees a termination of the mesh refinement algorithm. The proposed length scale defining the resolution of the interface thickness is given as

$$l_\gamma = \min \left\{ \max \left\{ \frac{C_\gamma}{|\nabla \gamma_{ref}|}, l_{min} \right\}, l_{max} \right\} \ , \tag{5.18}$$

where coefficient C_γ defines the target resolution of the interface thickness based on the colour function gradient. The upper and lower bound of the length scale for the resolution of the interface thickness should be suitable for the size of the problem and the dominating length scales of the expected flow field, in order to keep the interface thin at a reasonable computational cost.

Meshes obtained with different values of C_γ and different lower and upper bounds are shown in Figure 5.20. Comparing Figures 5.20a and 5.20b only the value of C_γ differs. The two meshes show substantial differences in mesh resolution and a strong impact of the weighting of the colour function gradient in determining the reference length scale. The relative volume error of the spherical interface is 3×10^{-4} for $C_\gamma = 10$ compared to 5×10^{-5} for $C_\gamma = 1$, emphasising the strong impact of C_γ on the mesh observed in Figures 5.20a and 5.20b. The differences resulting from the definition of the lower bound of the interface thickness can be examined by comparing Figures 5.20b and 5.20c. In Figure 5.20c the lower bound of $l_{min} = r/10$ effectively limits the mesh resolution. The volume error difference between the two cases with $C_\gamma = 1$ is smaller than the qualitative assessment of Figures 5.20b and 5.20c suggests, with a relative volume error of 7×10^{-5} for $l_{min} = r/10$ compared to 5×10^{-5} for $l_{min} = r/40$.

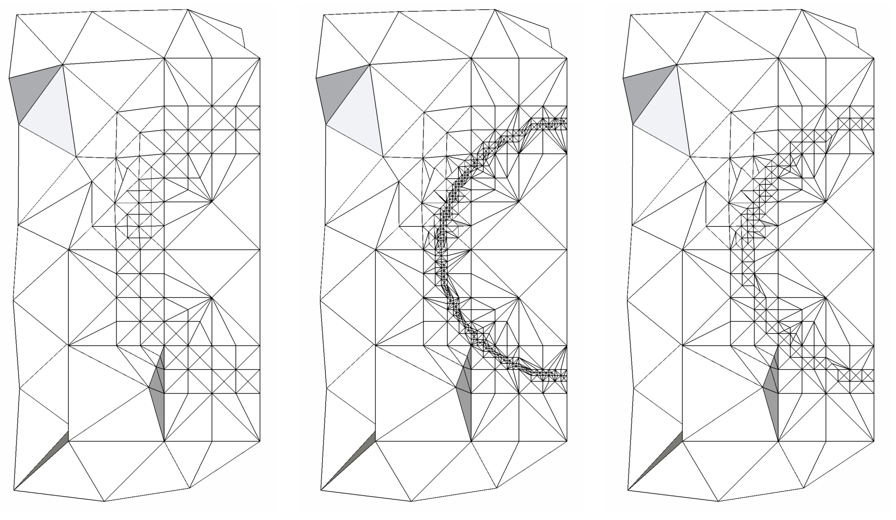

(a) $C_\gamma = 10$, $l_{min} = r/40$, $l_{max} = r/4$ (b) $C_\gamma = 1$, $l_{min} = r/40$, $l_{max} = r/4$ (c) $C_\gamma = 1$, $l_{min} = r/10$, $l_{max} = r/4$

Figure 5.20.: Tetrahedral mesh refined based on the colour function gradient of a spherical interface, as defined in Eq. 5.18. C_γ defines the target mesh resolution with respect to the interface thickness and l_{min} and l_{max} are predefined lower and upper bounds for the interface thickness. The figures show the mesh cells with an x-coordinate of the cell-centre larger than half the domain size.

5.6. Advecting Velocity on Adaptive Meshes

The content of this section has in parts been published in:

[54] Denner, F. and van Wachem, B.G.M.: Force-balancing at moving surface-tension-dominated interfaces on collocated unstructured meshes. 8th International Conference on Multiphase Flow (ICMF 2013), 26 - 31 May 2013, Jeju, Korea.

The adaption of the mesh poses two viable problems with respect to the advecting velocity and, thus, requires separate attention. Firstly, the transient term of the advecting velocity as given in Eq. 3.61 cannot be evaluated accurately at newly created mesh faces. Secondly, inconsistencies are introduced to the evaluation of the advecting velocity using Eq. 3.61 by applying a piecewise-linear mapping to the velocity and a piecewise-constant mapping to the colour function, as described in Section 5.4.6. Because this chapter is concerned with the application of adaptive tetrahedral meshes at interfaces, only surface force is considered in the following discussion. Other body forces, such as gravity, are omitted for the sake of brevity. Nonetheless, the proposed methodology is equally applicable if other or additional body forces are acting on the fluid.

In the solution procedure outlined in Section 3.7, the advective velocity is evaluated twice, implicitly and explicitly. The implicit evaluation of the advecting velocity is embedded in the continuity constraint (Eq. 3.25), which is an essential part of the fully-coupled flow equation system. The advecting velocity is also calculated explicitly at the end of each non-linear iteration and used to update the mass fluxes (Eq. 3.26) for the convective term of the momentum equation of the succeeding non-linear iteration. The advecting velocity calculated explicitly after the last non-linear iteration of a given time-step is also used to update the transient term of the continuity constraint and the volume flux for the VOF advection (Eq. 3.67) of the succeeding time-step.

After the mesh has been adapted and the solution has been mapped to the new mesh, the explicit advecting velocity has to be evaluated. It is important to understand that this explicitly calculated advecting velocity is only required for the first iteration of the succeeding time-step. On the old mesh, by virtue of Eq. 3.25, the velocity field resulting from the last non-linear iteration satisfies continuity and, as given by the balanced-force numerical framework presented in Chapter 3, the pressure gradient is in balance with the acting body forces. Thus, assuming the velocity field mapped to the new mesh also satisfies continuity, it is proposed to explicitly calculate the advecting velocity at face centres as

$$u_f^{n,t_a} = \boldsymbol{u}_f^{t_a} \cdot \boldsymbol{n}_f \ , \tag{5.19}$$

where superscript t_a denotes values at the time instant of mesh adaption on the new mesh. Calculating the advecting velocity in the proposed way circumvents the issue of unreliable data of the previous time-step and the implicit advecting velocity, implemented as part of the fully-coupled flow equation system, of the succeeding time instant $t_a + \Delta t$ becomes

$$\begin{aligned} u_f^{n,t_a+\Delta t} &= \boldsymbol{u}_f \boldsymbol{n}_f - \alpha_f \, \hat{d}_f \left[\frac{p_Q - p_P}{|\boldsymbol{s}_f} - \frac{\rho_f}{2} \left(\frac{\nabla p|_P}{\rho_P} + \frac{\nabla p|_Q}{\rho_Q} \right) \boldsymbol{s}_f \right] \\ &+ \alpha_f \, \hat{d}_f \, \sigma \left[\kappa_f \frac{\gamma_Q - \gamma_P}{\Delta s} - \frac{\rho_f}{2} \left(\frac{\kappa_P \nabla\gamma|_P}{\rho_P} + \frac{\kappa_Q \nabla\gamma|_Q}{\rho_Q} \right) \boldsymbol{s}_f \right] \\ &+ c_f \, \hat{d}_f \underbrace{\left[u_f^{n,t_a} - \boldsymbol{u}_f^{t_a} \boldsymbol{n}_f \right]}_{=0} \ . \end{aligned} \tag{5.20}$$

Hence, the problematic transient term of the advecting velocity vanishes for the first iteration at the time instant following mesh adaption. Given pressure and colour function are mapped from the old mesh to the new mesh in the same way and applying the same methodology to evaluate the gradients of these variables, the pressure gradient and surface force are in discrete balance on the new mesh. Moreover, using the advecting velocity as defined in Eq. 5.19 to update the mass flux at cell faces the mass flux is given by Eq. 2.53 and, thus, the convective term of the momentum equations is consistently defined. In the subsequent non-linear iterations the advecting velocity is then again calculated in its original form as defined in Eq. 3.60.

Any remaining errors resulting from the solution mapping and the assumptions made in Eq. 5.19 should vanish during the course of the non-linear iterations within the first time-step after mesh adaption, given that velocity, pressure and colour function are consistently defined. Errors caused by mapping can be further diminished by reducing the size of the first time-step following mesh adaption, as it decreases coefficient $\hat{d}$ as well as the difference between the implicit and explicit pressure gradient and surface force terms of Eq. 5.20.

5.7. Moving Interface with Surface Tension on Adaptive Tetrahedral Mesh

A surface-tension-dominated interface moving through a rectangular domain at a constant velocity, the test case presented in Section 3.8.4.2, is used to test the tetrahedral mesh adaption methodology at interfaces and to verify the conservation of mass and the applicability of the balanced-force framework presented in Chapter 3. The considered density ratios between the fluids inside and outside the interface are $\rho_i/\rho_o = 10^0$ and $\rho_i/\rho_o = 10^6$ and the viscosity ratio between the two immiscible, incompressible fluids is unity. The applied Reynolds number $Re_d = |\boldsymbol{u}|\, \rho_i\, d/\mu$ and capillary number $Ca = |\boldsymbol{u}|\, \mu/\sigma$ are 0.01, and the geometrically exact curvature is explicitly imposed at the interface.

5.7.1. Properties of the Adapted Mesh

The initial mesh consists of approximately 2.1×10^4 cells and is adapted in the interface region. The reference length scale, as discussed in Section 5.5, is determined by a minimum interface thickness $l_{min} = r/5$, curvature factor $C_\kappa = 2.5$ and interface band $C_{\nabla\gamma} = 0.1$ based on a reference colour function convoluted applying the Laplacian filter (see Eq. 4.1) for 5 iterations. The dihedral angles of new and modified elements is explicitly bounded to $13° \leq \vartheta \leq 167°$. An example of the adapted mesh at time $t = 1.907\,s$ is given in Figure 5.21.

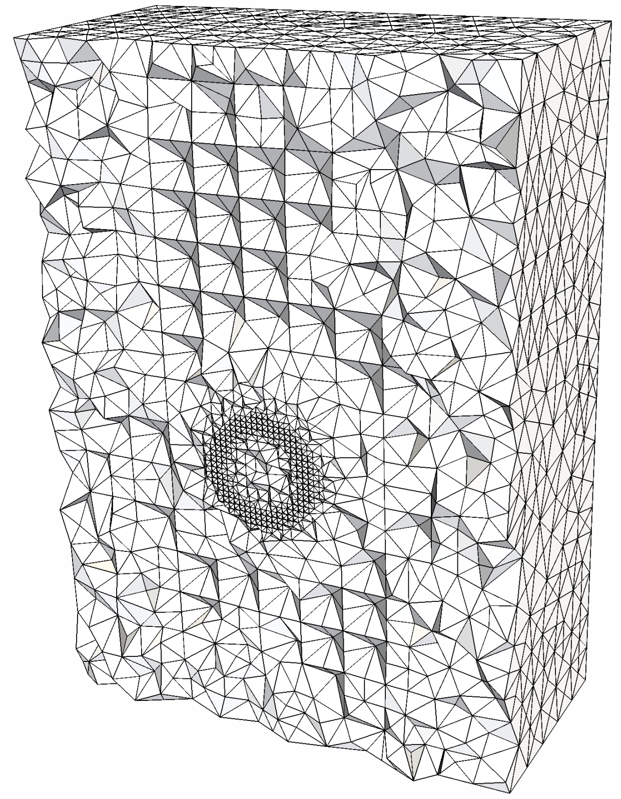
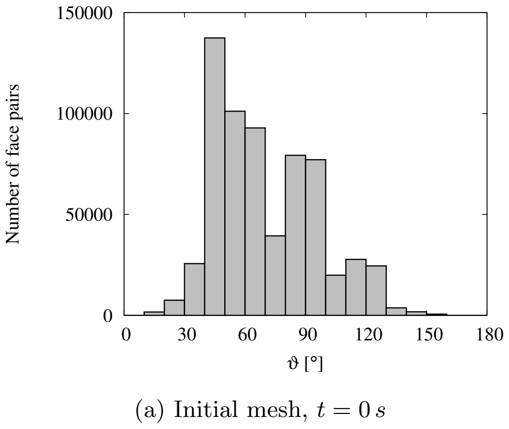
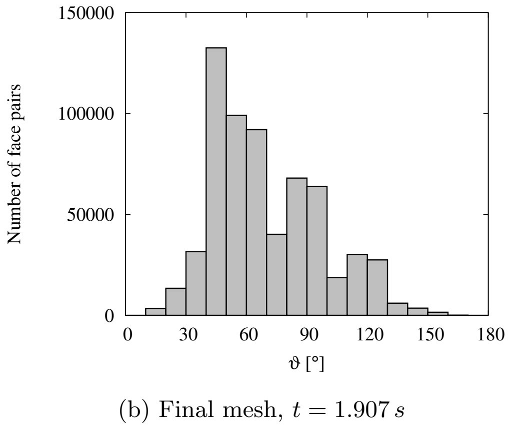


Figure 5.21.: The adapted tetrahedral mesh at $t = 1.907\,s$ used to simulate the moving surface-tension-dominated interface.

As discussed in Section 5.3, the quality of the mesh is paramount in performing accurate numerical simulations. Particular attention is given to the dihedral angle ϑ, the face skewness φ as defined in Eq. 5.5 and the face non-orthogonality α^* as defined in Eq. 5.6. Figure 5.22 shows the histograms of the dihedral angle for every face pair of the mesh at two different time instants: the adapted mesh after initialisation in Figure 5.22a and the mesh after the last adaption step at $t = 1.907\,s$ in Figure 5.22b. In both meshes, the largest share of dihedral angles lies between 40° and 70°. Comparing the histograms with the mesh shown in Figure 5.21, it appears that these are the angles of the large clusters of somewhat regularly distributed elements remaining from the initial mesh. At the later time instant more dihedral angles are located near the lower and upper bounds of 13° and 167°, respectively.



(a) Initial mesh, $t = 0\,s$

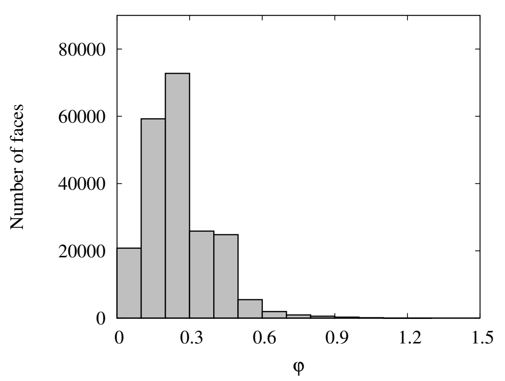


(b) Final mesh, $t = 1.907\,s$

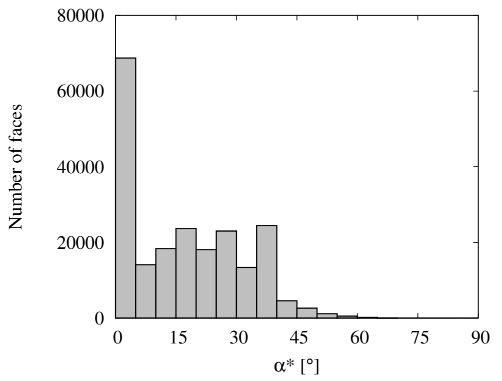
Figure 5.22.: Histogram of the dihedral angle ϑ of all face pairs of the adaptive tetrahedral mesh.

The skewness of each individual mesh face of the final mesh is given in Figure 5.23a. A large number of faces have a skewness in the range $0.1 \leq \varphi < 0.3$, originating from the previously mentioned large area of regularly distributed elements. However, the histogram also shows a significant number of faces with a skewness of $\varphi \geq 0.5$ and, in particular, 455 faces with a skewness of $\varphi \geq 1$, posing a potential problem for the numerical solving algorithm and, as explained in Section 5.3, for the explicit evaluation of spatial gradients.

Figure 5.23b shows the histogram of the non-orthogonality of the mesh faces at $t = 1.907\,s$, quantified by the angle between the normal vector of a given cell and the vector connecting the cell centres adjacent to that face. As seen for the dihedral angle and the skewness, the regularly distributed elements in large parts of the domain can be identified in the histogram of the non-orthogonality as their non-orthogonality angle is $\alpha^* \approx 0°$, as observed in Figure 5.21. More interesting, however, is the considerable number of faces which have a non-orthogonality angle of $\alpha^* \geq 38°$, as this is the angle identified in studies of Ahipo and Traoré [2] and Traoré *et al.* [234] above which the orthogonal correction of the deferred correction method becomes numerically unstable. It is worth reminding, that the deferred correction methodology is used in the presented framework for the viscous



(a) Skewness φ



(b) Non-orthogonality α^*

Figure 5.23.: Histogram of the skewness φ and the non-orthogonality α^* of the adaptive tetrahedral mesh at time $t = 1.907\,s$.

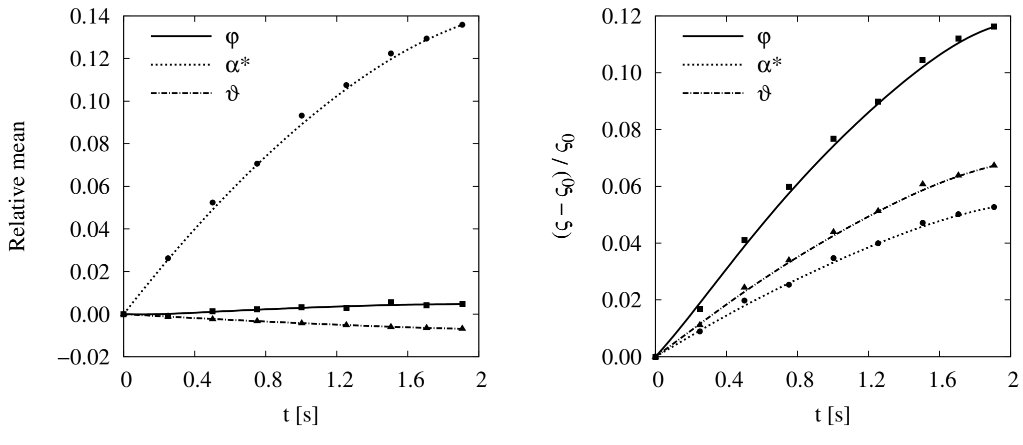
stresses of the momentum equations as well as the pressure gradient and body force terms of the continuity constraint. Since an overrelaxed correction is used in the presented framework, as described in Section 2.2.4, the numerical solving algorithm is robust despite the large non-orthogonality, however, at the price of a reduced accuracy [106].

The change of the mean skewness, shown in Figure 5.24a, is negligible, being $< 1\,\%$. The rise of the standard deviation ς of the face skewness shown in Figure 5.24b, on the other hand, is significant. Given that the skewness is defined in the interval $0 \leq \varphi < \infty$, the increasing standard deviation in conjunction with the almost constant mean indicates a considerable increase in the number of faces with large skewness. Similar conclusions can be drawn for the rising standard deviation of the dihedral angle ϑ with practically constant mean. This corresponds well with the observations made in comparing the histograms of the dihedral angle of the initial mesh and the final mesh, showing an overall unchanged distribution of the dihedral angle but more face pairs with a dihedral angle close to the lower and upper bounds in the final mesh. The mean of the face non-orthogonality rises significantly during the course of the simulation, as observed in Figure 5.24a, with a comparably moderate rise of the standard deviation. This suggests a general increase of the non-orthogonality of a large number of mesh faces.

In order to examine potential dependencies of the considered mesh properties, Table 5.1 presents the *Pearson product-moment correlation coefficient*, defined as

$$\varrho(X,Y) = \frac{\sum_{i=1}^{N}(X_i - \overline{X}) \cdot (Y_i - \overline{Y})}{(N-1)\,\varsigma_X\,\varsigma_Y}\ , \tag{5.21}$$

for combinations of the examined mesh properties. In Eq. 5.21, ς represents the standard deviation and X and Y are the examined data sets. The correlation coefficient ranges from -1 to 1, with $\varrho = 0$ representing no correlation between the data sets and $\varrho = -1$ and $\varrho = 1$ representing a perfect negative or positive correlation of the data sets, respectively. Face skewness φ and face non-orthogonality α^* are basically uncorrelated but the results show a notable correlation between face skewness and dihedral angle as well as non-orthogonality



(a) Relative mean (b) Relative standard deviation

Figure 5.24.: Evolution of the relative mean and relative standard deviation of skewness φ, non-orthogonality α^* and dihedral angle ϑ of the adaptive tetrahedral mesh.

and dihedral angle. Face skewness and face non-orthogonality both have a significant negative correlation with the minimum adjoining dihedral angle of the respective mesh face and a similarly positive correlation with the largest adjoining dihedral angle adjacent of the respective mesh face. Thus, if the minimum dihedral angle decreases or the maximum dihedral angle increases, the face skewness and face orthogonality are likely to rise. This observation is supported by intuition and the generally advocated notion that an element with bad dihedral angles, *i.e.* very small and very large angles, typically causes higher discretisation errors. It should be noted that, although both meshes consist of more than 2×10^5 faces, the distribution and orientation of the mesh faces are, of course, not perfectly independent, particularly in the large regions where the elements are distributed somewhat regularly.

Table 5.1.: Correlation coefficients ϱ of skewness φ, non-orthogonality α^*, minimum dihedral angle $\vartheta_{f,min}$ and maximum dihedral angle $\vartheta_{f,max}$ for all faces f of the initial and final mesh.

Correlation coefficient	$t = 0\,s$	$t = 1.907\,s$
$\varrho(\varphi_f, \alpha^*)$	-0.113	-0.111
$\varrho(\varphi_f, \vartheta_{f,min})$	-0.524	-0.561
$\varrho(\varphi_f, \vartheta_{f,max})$	$+0.554$	$+0.602$
$\varrho(\alpha^*_f, \vartheta_{f,min})$	-0.472	-0.507
$\varrho(\alpha^*_f, \vartheta_{f,max})$	$+0.544$	$+0.548$

5.7.2. Continuity and Mass Conservation

Continuity and the conservation of the colour function has been discussed and demonstrated in Section 3.8.5 for simulations on stationary meshes. As described in Section 5.6, the advecting velocity is extended for the application on adaptive meshes. Thus, it is important to verify that the made assumptions do not impair the basic integrity of the numerical framework.

The cumulative continuity error of the moving interface on the adaptive tetrahedral mesh examined in the previous section, shown in Figure 5.25a, is negligible for both considered density ratios. Occasional small perturbations induced by the adaption of the mesh are evident, however, they are insignificant and vanish quickly. The volume error of the volume inside the interface relative to its initial volume, shown in Figure 5.25b, is negligible as well, being $3 \times 10^{-9}\,\%$ at the end of the simulation. What is more, the results are in excellent agreement with the results obtained on stationary Cartesian and tetrahedral meshes presented in Section 3.8.5. Hence, the results verify that the numerical framework presented in Chapter 3 together with the extension made in Section 5.6 satisfies continuity and conserves the colour function accurately on adaptive meshes.

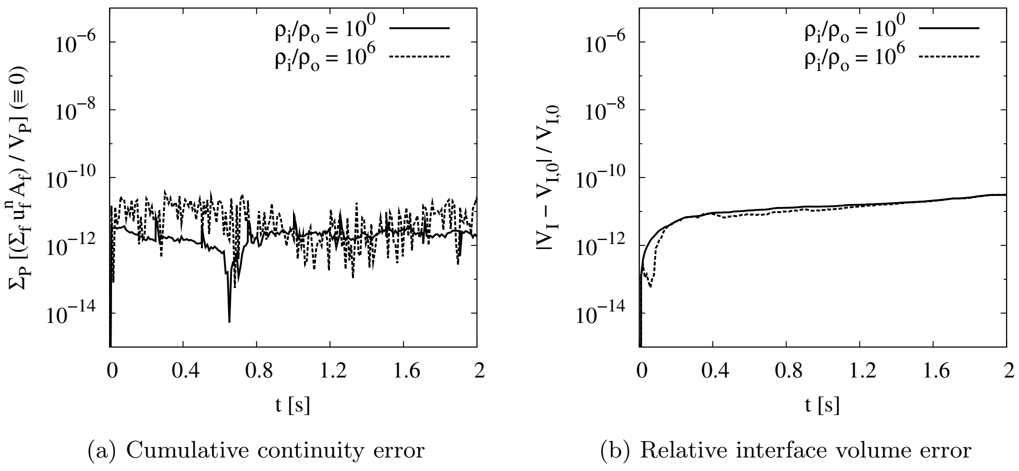


Figure 5.25.: Cumulative continuity error and relative interface volume error of the surface-tension-dominated moving interface presented in Section 3.8.4.2 on an adaptive tetrahedral mesh.

5.7.3. Verification of Force-Balancing

Similar as for continuity and the conservation of the colour function, the impact of mesh adaption on the force-balancing as well as the corresponding extension of the presented momentum interpolation method to adaptive meshes in Section 5.6 may affect the force-balancing methodology of the numerical framework and, therefore, requires additional verification. Velocity gradients of the considered test case are negligible, due to the constant velocity of the interface in conjunction with the very low Reynolds number ($Re_d = 0.01$). Furthermore, given the geometrically exact curvature explicitly imposed at the spherical interface and the low capillary number ($Ca = 0.01$), the interface remains in mechanical equilibrium. Thus, any imbalance between surface force and pressure gradient manifests itself in parasitic currents and in an erroneous prediction of the pressure jump across the interface, as previously discussed in Section 3.8.4.

Figure 5.26a shows the maximum velocity error and Figure 5.26b the maximum pressure error as a function of time on the adaptive tetrahedral mesh. The parasitic velocity magnitude as well as the pressure error are both negligible and of the same order of magnitude as on the stationary tetrahedral mesh presented in Figures 3.13b and 3.14b. Moreover, the impact of the mesh adaption on the results is insignificant and the velocity error as well as the pressure error show only minor differences between the two applied density ratios. This proves the applicability of the presented balanced-force numerical framework and the extension to it made in Eq. 5.19 to adaptive meshes.

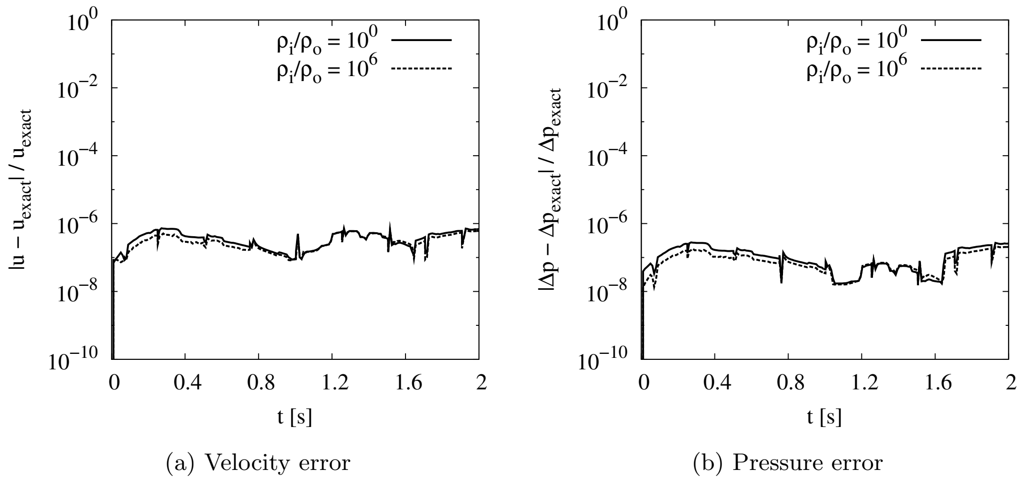


Figure 5.26.: Velocity error and pressure error for a surface-tension-dominated interface moving at a constant velocity and using the exact curvature to determine surface force on an adaptive tetrahedral mesh.

5.8. Summary

In this chapter the basics of the application of adaptive tetrahedral meshes to interfacial flows have been studied. The difficulties surrounding adaptive meshing at interfaces have been highlighted and the available adaption methods for tetrahedral meshes have been reviewed.

Based on the outlined difficulties and the available methods, an adaption algorithm and its implementation concept have been presented. The implementation concept limits mesh adaption to a local set of mesh entities and, thus, considerably simplifies the data management and the extension to parallel computer architectures. Furthermore, the solution mapping between meshes for different variables has been assessed, concluding that a piecewise-constant mapping for the colour function is preferable to a piecewise-linear mapping, as for instance applied to velocity.

A particular focus of this chapter has been the adequate mesh resolution at interfaces. As mentioned previously, the literature on suitable reference length scales and other error measures to define the mesh resolution at interfaces is very limited. In Section 5.5 the mesh resolution at interfaces has been discussed and reference length scales have been proposed, designed to control an adaptive tetrahedral mesh refinement in the vicinity of the interface. The presented numerical experiments demonstrate the applicability and impact of the proposed reference length scale.

The examination of the sample adaptive tetrahedral mesh in Section 5.7.1 has found a statistically notable correlation between the face skewness and face non-orthogonality with the adjoining dihedral angles of a given face. Furthermore, the analysis of the mesh showed that mesh adaption should be applied with great care as it tends to include elements with inferior properties in regions where the mesh resolution varies considerably, as for instance close to the interface. Further work is required in the future to improve the understanding of the underlying mechanisms leading to an increased face skewness and face non-orthogonality.

As previously discussed in Section 2.3.3, a local imbalance between body forces, in particular surface force, and the pressure gradient leads to substantial parasitic currents and is a major source of errors in two-phase flow simulations. Up to date, however, the applicability of a balanced-force framework on unstructured adaptive meshes has not been demonstrated in the literature. The extension to the advecting velocity (Eq. 3.61) proposed in Section 5.6 assures force-balancing on unstructured adaptive meshes as demonstrated in Section 5.7.3. Thus, the numerical framework presented in Chapter 3 is fully applicable to adaptive meshes. In this regard, it has also been successfully verified that the numerical framework including its extension to adaptive meshes satisfies continuity and conserves the mass of both fluids. It is worth mentioning that, although demonstrated on tetrahedral meshes, the theory of the balanced-force numerical framework is equally applicable to other unstructured adaptive meshes.

6. Further Case Studies

In this section additional case studies are presented and discussed to further validate and assess the presented numerical framework for incompressible two-phase flows. The test cases are chosen because of their particular informative value in order to scrutinise the numerical framework in its entirety. Firstly, Section 6.1 studies the influence of interface properties on parasitic currents. A better understanding of the involved error sources helps to perform more accurate two-phase flow simulations. The rise of a bubble due to buoyancy is presented in Section 6.2, demonstrating the correct interaction of surface force, gravity force and viscous stresses. Section 6.3 presents the rise of liquid inclusions at low Reynolds number, used to investigate the accurate prediction of capillary instabilities. Subsequently, the numerical framework is assessed with respect to interface topology changes in Section 6.4, simulating coaxial and oblique coalescence of two rising bubbles. Lastly, the numerical diffusion of the interface advection on unstructured meshes is revisited in Section 6.5. The major findings of the case studies are summarised in Section 6.6.

The interface normal vector and the interface curvature are evaluated using the CELESTE method, presented in Section 4.3. Unless otherwise stated, the applied convolution length is $\epsilon = 2d^*$ and the stencil size for the evaluation of the interface normal vector and the interface curvature are $l_{s,m} = 4d^*$ and $l_{s,\kappa} = 2d^*$, respectively.

6.1. Influence of Interface Properties on Parasitic Currents

The content of this section has been published in:

[55] Denner, F. and van Wachem, B.G.M.: Fully-coupled balanced-force VOF framework for arbitrary meshes with least-squares curvature evaluation from volume fraction. Numerical Heat Transfer, Part B: Fundamentals, accepted for publication. DOI: 10.1080/10407790.2013.849996

Parasitic currents present a limiting factor for the applicability of VOF methods. Under specific circumstances, parasitic currents can become large enough to destroy the interface and, therefore, prohibit a feasible solution to a given problem. Using the CSF method [19] discussed in Section 2.3.3 to model the surface force, the surface tension coefficient σ and the interface curvature κ are the key interface properties.

In order to verify the influence of interface properties on the outcome of VOF simulations, different surface tension coefficients are applied to the inviscid static drop in equilibrium discussed in Section 4.3.5. Moreover, the size of the drop and its surrounding domain is varied to examine the influence of the interface curvature on the simulation results. The tests are performed on an equidistant Cartesian mesh of 40^3 cells, shown in Figure 3.9a, and on a tetrahedral mesh with approximately 6.0×10^4 cells, shown in Figure

3.9c. The CELESTE method is used to evaluate the interface curvature with a stencil size of $l_s = 4\Delta x$ on the Cartesian mesh and $l_s = 6d^*$ on the tetrahedral mesh. The applied convolution length is $\epsilon = 2\Delta x$ on the Cartesian mesh and $\epsilon = 3d^*$ on the tetrahedral mesh.

A linear relationship between surface tension coefficient σ and parasitic currents for a viscous static drop in equilibrium has been demonstrated by Francois *et al.* [71], applying a convolution approach as well as a height function technique, and Ubbink [238], using a convolution approach. The results presented in Figure 6.1 also show a linear relationship between surface tension coefficient and parasitic currents. Furthermore, Figure 6.1 shows a predominantly second-order rise of parasitic currents with increasing curvature. The mean curvature error is constant with changing surface tension but increases mainly linear with increasing interface curvature, as depicted in Figure 6.2.

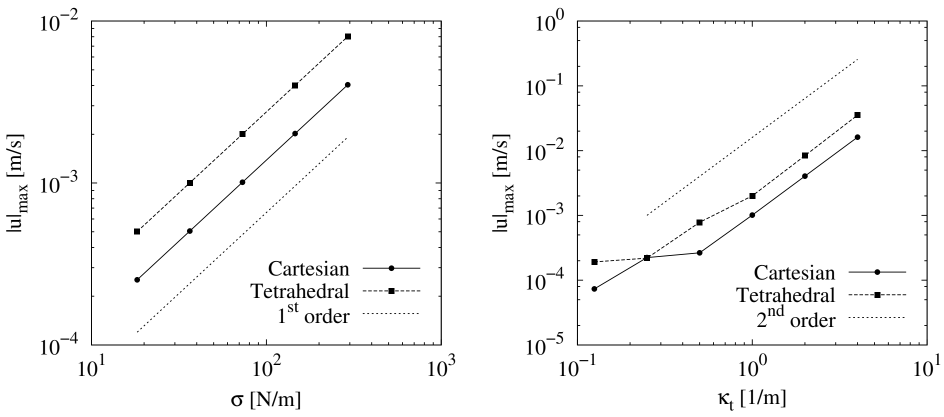


Figure 6.1.: Maximum velocity magnitude after one time-step as a function of surface tension coefficient σ and theoretical interface curvature κ_t for a static inviscid drop in equilibrium. The drop with surface tension coefficient $\sigma = 73\,N\,m^{-1}$ and radius $r = 2\,m$ is positioned at the centre of an $8\,m \times 8\,m \times 8\,m$ domain and is simulated with a time-step of $10^{-3}\,s$.

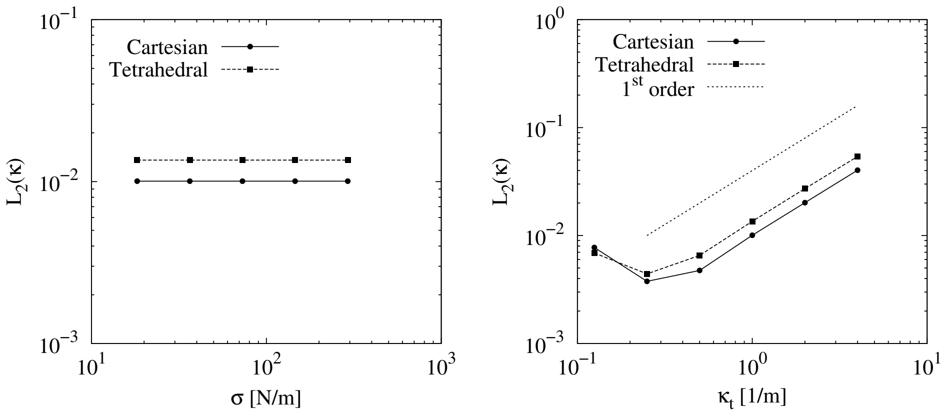


Figure 6.2.: L_2 error norm of interface curvature κ as a function of surface tension coefficient σ and theoretical interface curvature κ_t for a spherical drop.

The different impact of surface tension coefficient and interface curvature on parasitic currents, despite the first-order influence of both parameters on the surface force according to the definition in Eq. 2.48, can be explained by examining the presented results in more detail. In case of an inviscid two-phase flow without gravity, the momentum equation presented in Eq. 2.52 reduces to

$$\frac{\partial u_i}{\partial t} + u_j \frac{\partial u_i}{\partial x_j} = -\frac{1}{\rho}\frac{\partial p}{\partial x_i} + \frac{1}{\rho}\,\sigma\,\kappa\,\frac{\partial \gamma}{\partial x_i} \; . \tag{6.1}$$

The local interface curvature κ includes errors arising from its approximation and may be split up as $\kappa = \overline{\kappa} + \kappa'$, where $\overline{\kappa}$ represents the base value, *i.e.* the average curvature, and κ' represents the absolute local error. The absolute local error is defined as $\kappa' = \xi\,\overline{\kappa}$ with the relative local curvature error being $\xi = (\kappa - \overline{\kappa})/\overline{\kappa}$. The results presented in Section 3.8.4 as well as the results of other studies [71, 154] indicate that the pressure gradient at the interface reflects the base surface force $\overline{\boldsymbol{f}}_s = \sigma\,\overline{\kappa}\,\nabla\gamma$. Assuming a perfect correlation between base surface force and pressure gradient means that for a stationary problem both terms cancel each other out and Eq. 6.1 becomes

$$\frac{\partial u_i}{\partial t} + u_j \frac{\partial u_i}{\partial x_j} = \frac{1}{\rho}\,\sigma\,\kappa'\,\frac{\partial \gamma}{\partial x_i} \; . \tag{6.2}$$

Hence, parasitic currents arise due to the local surface force error $\boldsymbol{f}'_s = \sigma\,\kappa'\,\nabla\gamma$ and, therefore, due to the local error of the interface curvature $\kappa' = \xi\,\overline{\kappa}$. Changing the surface tension coefficient σ only changes the magnitude of the local surface force error $\boldsymbol{f}'_s$ but not its distribution. Accordingly, parasitic currents have a linear relation to a changing surface tension coefficient. A similar conclusion can be drawn for a changing interface curvature, which changes the magnitude of the surface force error and, as a result, changes the resulting parasitic currents. However, the results shown in Figure 6.2 also show a changing curvature error, suggesting a higher relative error ξ for higher curvature values. The higher relative curvature error originates from larger aliasing errors when the interface curvature is evaluated on a smaller mesh spacing, resulting from the smaller domain. As a result, the local error in surface force $\boldsymbol{f}'_s$ is not only affected by the changing base curvature $\overline{\kappa}$, but also by the changing relative error ξ, leading to a predominantly second-order change of parasitic currents, as seen in Figure 6.1. It is worth emphasising at this point that the parasitic currents are not considerably affected by the mesh type, neither the source of parasitic currents nor the general relationship to interface properties.

Considering static fluid particles in equilibrium with equal pressure jump across the interface $\Delta p = \sigma\kappa$, the presented results indicate different parasitic currents for different combinations of surface tension coefficient σ and interface curvature κ. The interface properties of a fluid particle with an accelerating force acting on it, such as the rising bubble in Sections 4.4.2, 4.5 and 6.2, can be adjusted with the aim of diminishing the impact of parasitic currents without changing its Eötvös number $Eo = \rho_o\,g\,d_0^2/\sigma$. Thus, the presented findings can provide guidance in choosing fluid properties and spatial dimensions which diminish parasitic currents and curvature errors, and the findings can help to analyse simulation results.

According to Scardovelli and Zaleski [205], computational studies of two-phase flows become difficult for Laplace number $La_r = \sigma\,\rho\,r/\mu^2 \gtrapprox 10^6$ because of the resulting parasitic currents, which can become large enough to destroy the interface. This assumption is supported by the results presented in Figure 6.1, showing that parasitic currents increase linearly with increasing surface tension coefficient σ. However, the presented results also suggest that the Laplace number is not a reliable indicator for the applicability of computational methods to two-phase flow problems. As shown in Figure 6.1, the parasitic currents predominantly increase with second-order when increasing the interface curvature κ. Replacing the fluid particle radius r with the equivalent interface curvature given in Eq. 3.144, the Laplace number is defined as $La_\kappa = 2\,\sigma\,\rho/\kappa\,\mu^2$, meaning that an increasing interface curvature results in a decreasing Laplace number. Thus, parasitic currents can theoretically become large enough to break up the interface at any Laplace number.

6.2. Spherical Cap Bubble Rising due to Buoyancy

The content of this section has in parts been published in:

[53] Denner, F. and van Wachem, B.G.M.: Two-Phase Flow Modelling on Arbitrary Meshes: Superior VOF Curvature Estimation and the Issue of Convolution. International Conference on Numerical Methods in Multiphase Flows, 12 - 14 June 2012, State College, PA, USA.

[55] Denner, F. and van Wachem, B.G.M.: Fully-coupled balanced-force VOF framework for arbitrary meshes with least-squares curvature evaluation from volume fraction. Numerical Heat Transfer, Part B: Fundamentals, accepted for publication. DOI: 10.1080/10407790.2013.849996

The content of this section has in parts been submitted for publication in:

[57] Denner, F., van der Heul, D., Oud, G.T., Martins Villar, M., da Silveira Neto, A. and van Wachem, B.G.M.: Comparative study of mass-conserving interface capturing frameworks for two-phase flows with surface tension. Submitted to International Journal of Multiphase Flow on 27 June 2013.

A bubble rising in a heavier fluid due to buoyancy is simulated, as previously presented in Sections 4.4.2 and 4.5. The precise calculation and interaction of viscous stresses and the forces due to buoyancy and surface tension are essential for the accurate prediction of the bubble shape and the bubble rise velocity. In particular the overshoot of the rise velocity shortly after the bubble starts ascending and the terminal rise velocity are key characteristics. The finite size of the fluid domain, however, affects the rise velocity of the bubble [88], which is important for the correct interpretation of the simulation results. The expected rise velocity is corrected by the semi-empirical correlation proposed by Harmathy [88], given in Eq. 4.44. The accuracy and applicability of this correlation in numerical studies has been demonstrated by van Sint Annaland *et al.* [245], using a VOF method, and by Hua *et al.* [99], using a front-tracking method.

Following the test case proposed by Lebaigue *et al.* [124], the bubble is initially spherical with a diameter of $d_0 = 0.02\,m$, a Morton number $Mo = g\,\mu_o^4/\rho_o\,\sigma^3 = 0.056$ and an Eötvös number $Eo = \rho_o\,g\,d_0^2/\sigma = 40$, where subscript o denotes the fluid properties outside the bubble and subscript i denotes properties inside the bubble, and g is the gravitational acceleration. The fluid of the continuous phase has a density of $\rho_o = 1000\,kg\,m^{-3}$ and a viscosity of $\mu_o = 2.73556 \times 10^{-1}\,Pa\,s$, and the fluid of the dispersed phase has a density of

$\rho_o = 10\,kg\,m^{-3}$ and a viscosity of $\mu_o = 2.73556 \times 10^{-3}\,Pa\,s$. The surface tension coefficient is $\sigma = 0.1\,N\,m^{-1}$ and the gravity of $g = 10\,m\,s^{-2}$ is acting in negative y-direction. The domain dimensions are $5d_0 \times 7d_0 \times 5d_0$. Two different meshes are applied, an equidistant Cartesian mesh with $100 \times 140 \times 100$ cells, shown in Figure 6.3a, and a tetrahedral mesh with approximately 1.4×10^6 cells and an average cell-to-cell distance of $d^* \approx 8.07 \times 10^{-4}\,m$, shown in Figure 6.3b. The capillary time-step constraint, following Eq. 3.104, for this case is $t_c \leq 8.97 \times 10^{-4}\,s$ on the Cartesian mesh and $t_c \leq 6.5 \times 10^{-4}\,s$ on the tetrahedral mesh. Both fluids are initially at rest and the motion of the bubble is induced by buoyancy only. The boundary at the top of the domain is considered to be an outlet boundary, all other boundaries are free-slip walls.

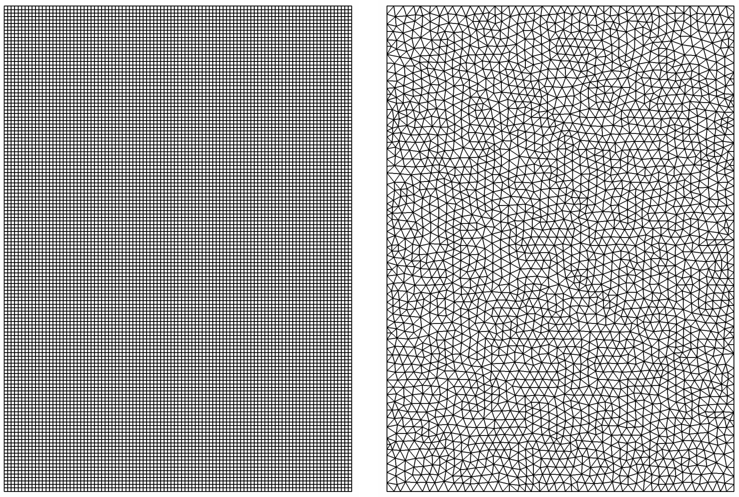
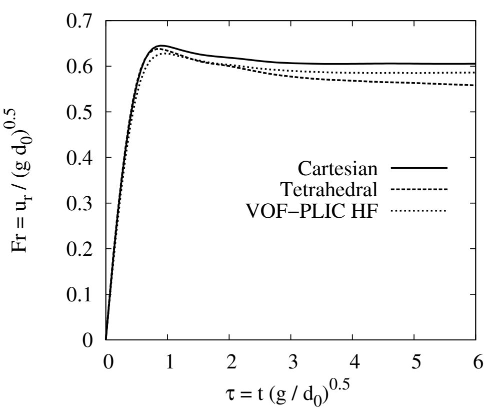
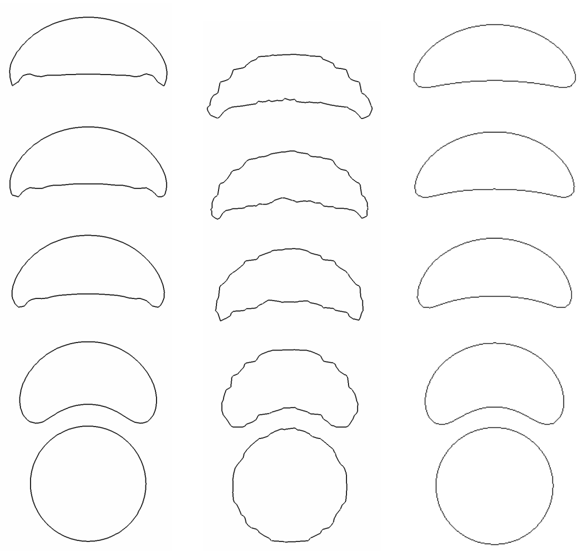

(a) Cartesian mesh (b) Tetrahedral mesh

Figure 6.3.: The two meshes applied to simulate the spherical cap bubble rising due to buoyancy. The equidistant Cartesian mesh consists of $100 \times 140 \times 100$ cells and the tetrahedral mesh has approximately 1.4×10^6 cells.

As previously discussed in Section 4.4.2, empirical studies by Clift *et al.* [27, Fig 2.5] suggest a terminal Reynolds number of $Re_d = |\boldsymbol{u}|_r\,\rho_o\,d_0/\mu_o \approx 20.5 - 21.0$ for a bubble with $Eo = 40$ and $Mo = 0.056$. This Reynolds number represents a Froude number of $Fr = |\boldsymbol{u}_r|/\sqrt{d_0\,g} \approx 0.626 - 0.642$ with respect to the fluid properties defined above. Given the domain extend of $5d_0$ perpendicular to the gravitational acceleration, the rise velocity in the finite computational domain is approximately 96% of the rise velocity observed in a domain of infinite extend. Thus, the expected terminal rise velocity corresponds to a Froude number of $Fr \approx 0.601 - 0.616$. Numerical studies of Blanco-Alvarez [17] and Lubin *et al.* [140], using a computational domain of unspecified extend, obtained terminal Froude numbers of approximately 0.63 and 0.60, respectively. Results for this test case obtained by Martins Villar [151] and reported in [57] on the equidistant Cartesian mesh using a VOF-PLIC interface reconstruction method to advect the interface and a height function technique to evaluate the interface curvature, abbreviated herein as *VOF-PLIC HF*, are included in Figure 6.4 as a reference.



(a) Rise velocity



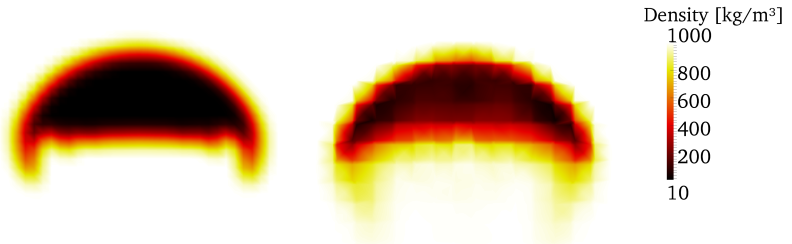
(b) Cartesian mesh (c) Tetrahedral mesh (d) VOF-PLIC HF

Figure 6.4.: Rise velocity and shape evolution of the rising bubble on a Cartesian mesh and a tetrahedral mesh. Reference results of Martins Villar [151] using a VOF-PLIC method to track the interface and a height function (HF) technique to evaluate the interface curvature, abbreviated VOF-PLIC HF, are shown as well. The bubble shapes are illustrated every 0.07 s ($\tau = 1.57$), starting from the initial position, in the x-y plane crossing through the centre of the domain.

The rise velocity of the bubble as a function of time, presented in Figure 6.4a, shows similar results on the Cartesian and the tetrahedral mesh. Predicting equal rise velocities during the initial acceleration phase of the bubble on both meshes, the rise velocities differ increasingly while the bubble assumes its terminal shape. The terminal Froude number at time $\tau = 6$ obtained on the Cartesian mesh is $Fr = |\boldsymbol{u}_r|/\sqrt{d_0\, g} = 0.606$, being in excellent agreement with the literature references given above, whereas on the tetrahedral mesh the resulting Froude number is 0.557. The rise velocity predicted on the Cartesian mesh is in very good agreement with the reference results of Martins Villar [151] obtained with a VOF-PLIC HF framework.

The bubble shapes obtained on both meshes, shown in Figures 6.4b and 6.4c, as well as the bubble shape obtained with the VOF-PLIC HF framework, shown in Figure 6.4d, are in very good agreement with each other. The predicted terminal rise velocity is equivalent to a Reynolds number of $Re_d = 19.81$ on the Cartesian mesh and $Re_d = 18.2$ on the tetrahedral mesh. The aspect ratio of the bubble on the Cartesian mesh is 0.403, with an eccentricity of 0.69. Both values are in very good agreement with the experimental results of Bhaga and Weber [16], suggesting an aspect ratio of approximately 0.395 and an eccentricity of roughly 0.64 for $Re_d = 19.81$, as well as with the results obtained using the VOF-PLIC HF framework of Martins Villar [151], predicting an aspect ratio of 0.408 and an eccentricity of 0.7. On the tetrahedral mesh, the bubble has an aspect ratio of 0.375 and an eccentricity of 0.71, which is in reasonably good agreement with the experimental results of Bhaga and Weber [16], suggesting an aspect ratio of approximately 0.4 and an eccentricity of approximately 0.64 for $Re_d = 18.2$. The shape of the rising bubble on both meshes, depicted in Figure 6.4, is also in excellent agreement with boundary-element simulations of Ryskin and Leal [201]. The drag force on the bubble as a potential reason for the different rise velocities can, therefore, be eliminated. The inherent mass conservation of the applied VOF methodology further excludes mass conservation errors as a potential reason for the discrepancy in rise velocity. On the tetrahedral mesh the relative error of the bubble volume is $6.1 \times 10^{-2}\,\%$ at the end of the simulation.

The colour function suffers from notable numerical diffusion on the tetrahedral mesh, reducing the effective density difference between the two fluids from initially $990\,kg\,m^{-3}$ to approximately $890-920\,kg\,m^{-3}$ at time $\tau = 6$, as observed in Figure 6.5. This represents a reduction of the effective buoyancy force acting on the bubble of $7-10\,\%$, which correlates well with the $8.1\,\%$ lower rise velocity at $\tau = 6$ on the tetrahedral mesh compared to the rise velocity on the Cartesian mesh. The issue of numerical diffusion of the colour function on tetrahedral meshes is further examined in Section 6.5. The comparison of the results obtained on the equidistant Cartesian with the numerical framework presented in this thesis and with the VOF-PLIC HF framework show no significant differences. This demonstrates the competitiveness of the compressive VOF method used in the presented numerical framework compared to advanced interface reconstruction methods that are typically limited to Cartesian meshes.



(a) Cartesian mesh (b) Tetrahedral mesh

Figure 6.5.: Density distribution of the rising bubble at time $\tau = 6$ on the equidistant Cartesian mesh and the tetrahedral mesh.

6.3. Stability of Liquid Inclusions at Low Reynolds Number

The content of this section has been submitted for publication in:

[57] Denner, F., van der Heul, D., Oud, G.T., Martins Villar, M., da Silveira Neto, A. and van Wachem, B.G.M.: Comparative study of mass-conserving interface capturing frameworks for two-phase flows with surface tension. Submitted to International Journal of Multiphase Flow on 27 June 2013.

The solely buoyancy driven free rise of a viscous inclusion, initially of ellipsoidal shape, at low Reynolds number ($Re < 0.1$) in another viscous fluid is simulated. As a result of the very low Reynolds number, the material derivative of the velocity can be neglected

[13, 202]. Therefore, the momentum equation becomes effectively linear and the problem reduces to balancing the acting body forces, *i.e.* surface force and buoyancy force, and the viscous stresses. The particular challenge of this test case is the accurate prediction of the critical capillary number beyond which the initially ellipsoidal inclusion does not restore its equilibrium (*i.e.* spherical) shape. The rising inclusion becomes unstable when the critical capillary number is exceeded and high curvatures occur due to the resulting shape of the inclusion. Because the shape and stability of the liquid inclusion are strongly dependent on the correct prediction of the surface tension effects and the viscous stresses in proximity of the interface, this test case represents a more rigorous validation of interface capturing/tracking methods than the test case presented in Section 6.2.

Following boundary-element simulations of Koh and Leal [117], reprinted in Figures 6.6 and 6.7, Lemmonier and Hervieu [125] proposed six different test cases. Each test case is characterised by the capillary number $Ca = U\,\mu_o/\sigma$, where U is the characteristic velocity, and the ratio between the inclusions length along its symmetry axis a_y and its radial axis a_r. The characteristic velocity for a spherical inclusion rising under the sole action of gravity within the Stokes approximation ($Re < 1$) is [13, 27]

$$U = \frac{2\,g\,R^2\,|\rho_o - \rho_i|}{3\,\mu_o} \frac{1 + \frac{\mu_i}{\mu_o}}{2 + 3\frac{\mu_i}{\mu_o}} \,, \tag{6.3}$$

where subscripts i and o denote fluid properties inside and outside the inclusion, and R is the equivalent radius of the inclusion $R = \sqrt[3]{a_y\,a_r^2}$. The characteristic time scale of the problem is defined as

$$\tau = \frac{R}{U} \,. \tag{6.4}$$

In all six test cases (A-F) the continuous phase has a density of $\rho_o = 2000\,kg\,m^{-3}$ and a viscosity of $\mu_o = 100\,Pa\,s$, and the liquid inclusion has a density of $\rho_i = 1980\,kg\,m^{-3}$ and a viscosity of $\mu_i = 50\,Pa\,s$. The gravity acting in negative y-direction is $g = 9.81\,m\,s^{-2}$. In cases A-C the liquid inclusion has initially a prolate ellipsoidal shape, as seen in Figure 6.6. The properties of test cases A-C are presented in Table 6.1. The liquid inclusion is expected to restore its equilibrium shape for test case A ($Ca = 1.25$), whereas in test cases B ($Ca = 1.5$) and C ($Ca = 2.0$) an instability develops at the tail of the liquid inclusion and the equilibrium shape is not fully restored, as boundary-element simulations of Koh and Leal [117] depicted in Figure 6.6 indicate. In test cases D-F the liquid inclusion has initially an oblate ellipsoidal shape, as seen in Figure 6.7, for which the properties are given in Table 6.2. According to Koh and Leal [117], the liquid inclusion is able to restore its equilibrium shape in test case D ($Ca = 3.5$) but develops a cavity in test cases E ($Ca = 4.0$) and F ($Ca = 10.0$), as shown in Figure 6.7. Experimental and numerical studies by Koh and Leal [118], Stone [221] and Manga and Stone [145] support these findings. Test cases A-C are simulated on an equidistant Cartesian mesh of $76 \times 220 \times 76$ cells and test cases D-F are simulated on an equidistant Cartesian mesh of $96 \times 260 \times 96$ cells. Both meshes have a mesh spacing of $\Delta x = 0.01\,m$.

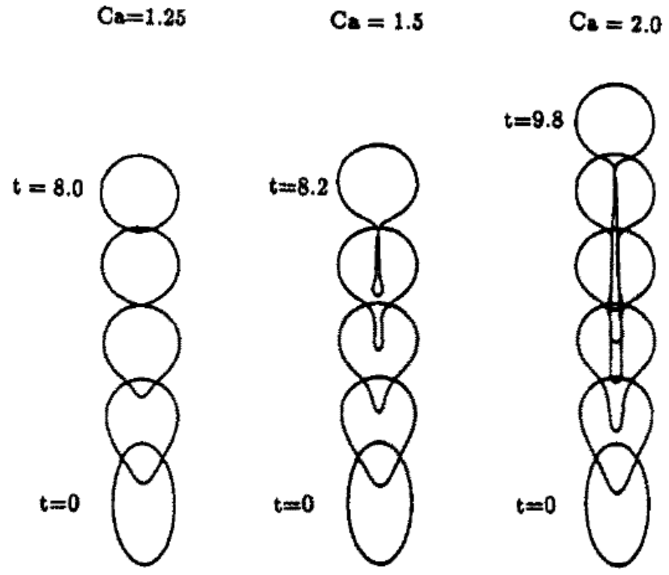


Figure 6.6.: Reference results of Koh and Leal [117] for the liquid inclusion test cases A-C ($Ca = 1.25 - 2.0$). The individual inclusion shapes are depicted at $t = 2\tau$ intervals as well as the time at which the inclusion has restored its equilibrium shape or at which the instability is fully developed, respectively. Reprinted with permission from Koh and Leal [117]. Copyright 1989, AIP Publishing LLC.

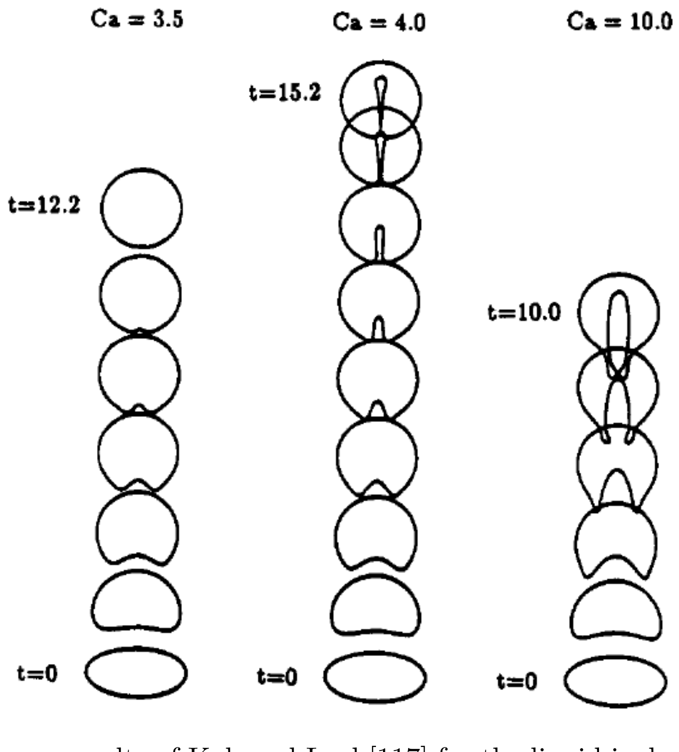


Figure 6.7.: Reference results of Koh and Leal [117] for the liquid inclusion test cases D-F ($Ca = 3.5 - 10.0$). The individual inclusion shapes are depicted at $t = 2\tau$ intervals as well as the time at which the inclusion has restored its equilibrium shape or at which the instability is fully developed, respectively. Reprinted with permission from Koh and Leal [117]. Copyright 1989, AIP Publishing LLC.

Table 6.1.: Properties of the prolate liquid inclusion test cases.

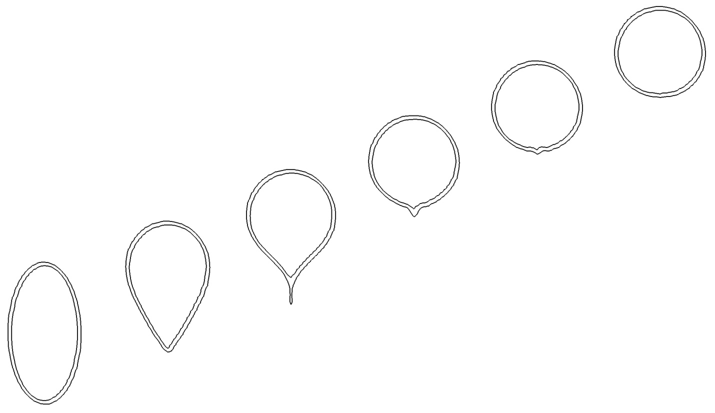
		Case A	Case B	Case C
Ca		1.25	1.5	2.0
σ		$0.711881\,N\,m^{-1}$	$0.593235\,N\,m^{-1}$	$0.444926\,N\,m^{-1}$
Δt_c	(Eq. 3.104)	$0.02109\,s$	$0.02311\,s$	$0.02668\,s$
a_r		$0.1\,m$	$0.1\,m$	$0.1\,m$
a_y		$0.2\,m$	$0.2\,m$	$0.2\,m$
U	(Eq. 6.3)	$8.8985 \times 10^{-3}\,m\,s^{-1}$	$8.8985 \times 10^{-3}\,m\,s^{-1}$	$8.8985 \times 10^{-3}\,m\,s^{-1}$
τ	(Eq. 6.4)	$14.1588\,s$	$14.1588\,s$	$14.1588\,s$

Table 6.2.: Properties of the oblate liquid inclusion test cases.

		Case D	Case E	Case F
Ca		3.5	4.0	10.0
σ		$0.403586\,N\,m^{-1}$	$0.353138\,N\,m^{-1}$	$0.141255\,N\,m^{-1}$
Δt_c	(Eq. 3.104)	$0.02801\,s$	$0.02995\,s$	$0.04735\,s$
a_r		$0.2\,m$	$0.2\,m$	$0.2\,m$
a_y		$0.1\,m$	$0.1\,m$	$0.1\,m$
U	(Eq. 6.3)	$1.4126 \times 10^{-2}\,m\,s^{-1}$	$1.4126 \times 10^{-2}\,m\,s^{-1}$	$1.4126 \times 10^{-2}\,m\,s^{-1}$
τ	(Eq. 6.4)	$11.2378\,s$	$11.2378\,s$	$11.2378\,s$

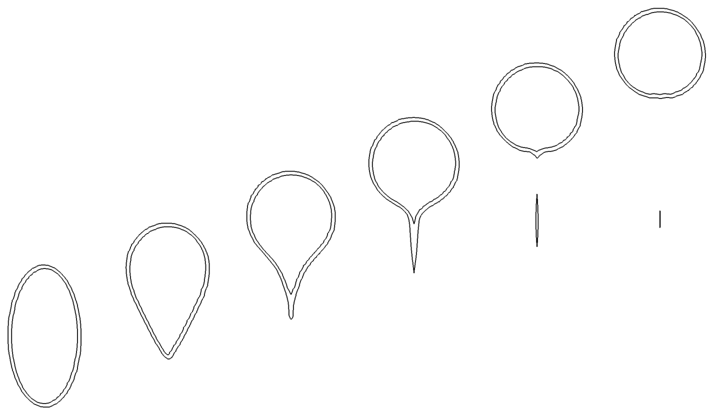
Figures 6.8 - 6.10 show the evolution of the initially prolate liquid inclusions (cases A-C) at different time instants. In case A, shown in Figure 6.8, the inclusion restores its equilibrium shape and predicts the evolution of the liquid inclusion shape with very high accuracy. Case B, shown in Figure 6.9, predicts the evolution of the liquid inclusion shape very accurate until $t = 6\tau$. However, the liquid inclusion does not develop a proper instability, apart from a small detachment, and restores its equilibrium shape eventually. In case C, on the other hand, the development of the instability at the tail of the liquid inclusion is predicted accurately, as shown in Figure 6.10. Apart from the minor disagreement regarding the instability in case B, the computational results obtained with the presented numerical framework are in excellent agreement with the reference results of Koh and Leal [117].

The results for the initially oblate liquid inclusions, test cases D-F, are presented in Figures 6.11 - 6.13. In case D, the liquid inclusion resumes its equilibrium shape slowly but predicts the final result accurately, as shown in Figure 6.11. Similar to case D, the evolution of the shape of the liquid inclusion in case E is slower than predicted by Koh and Leal [117], as observed in Figure 6.12. Moreover, the numerical framework underestimates the developing cavity. In case F, shown in Figure 6.13, the size of the cavity of the liquid inclusion is also underestimated but the overall prediction is in very good agreement with the reference results of Koh and Leal [117].



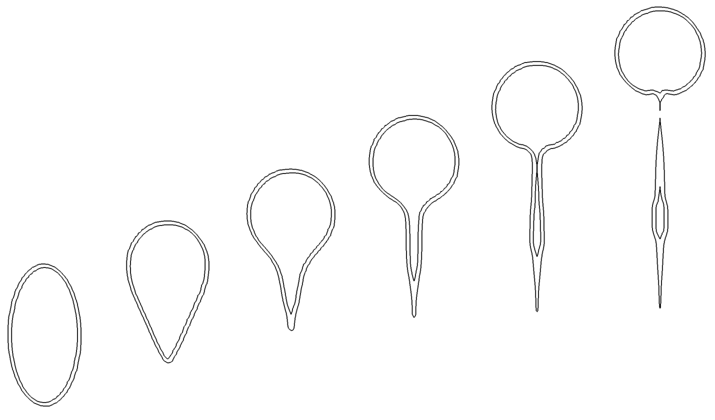
(a) $t = 0\tau$ (b) $t = 2\tau$ (c) $t = 4\tau$ (d) $t = 6\tau$ (e) $t = 8\tau$ (f) $t = 10\tau$

Figure 6.8.: Shape evolution of the liquid inclusion of test case A ($Ca = 1.25$) at different time instants. The interface region of the liquid inclusion is illustrated by isocontours for VOF colour function values $\gamma = 0.25$ and $\gamma = 0.75$ in the x-y plane crossing through the centre of the domain.



(a) $t = 0\tau$ (b) $t = 2\tau$ (c) $t = 4\tau$ (d) $t = 6\tau$ (e) $t = 8\tau$ (f) $t = 10\tau$

Figure 6.9.: Shape evolution of the liquid inclusion of test case B ($Ca = 1.5$) at different time instants. The interface region of the liquid inclusion is illustrated by isocontours for VOF colour function values $\gamma = 0.25$ and $\gamma = 0.75$ in the x-y plane crossing through the centre of the domain.



(a) $t = 0\tau$ (b) $t = 2\tau$ (c) $t = 4\tau$ (d) $t = 6\tau$ (e) $t = 8\tau$ (f) $t = 10\tau$

Figure 6.10.: Shape evolution of the liquid inclusion of test case C ($Ca = 2.0$) at different time instants. The interface region of the liquid inclusion is illustrated by isocontours for VOF colour function values $\gamma = 0.25$ and $\gamma = 0.75$ in the x-y plane crossing through the centre of the domain.

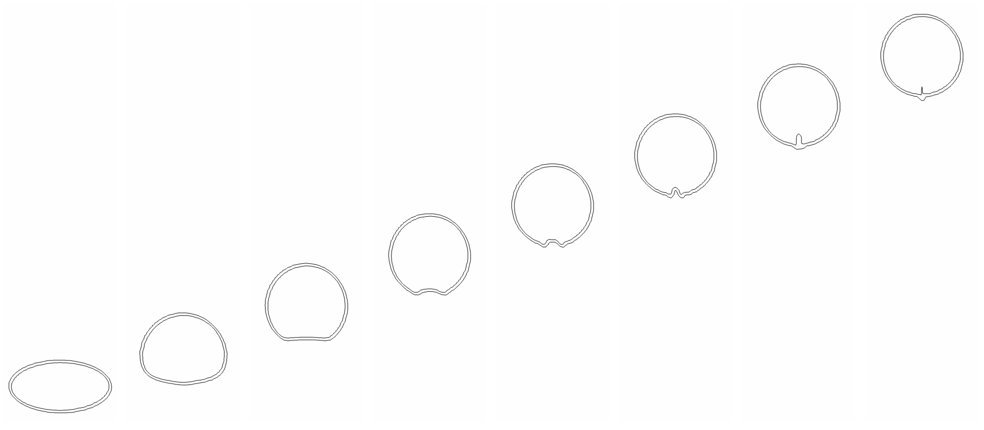

(a) $t = 0\tau$ (b) $t = 2\tau$ (c) $t = 4\tau$ (d) $t = 6\tau$ (e) $t = 8\tau$ (f) $t = 10\tau$ (g) $t = 12\tau$ (h) $t = 14\tau$

Figure 6.11.: Shape evolution of the liquid inclusion of test case D ($Ca = 3.5$) at different time instants. The interface region of the liquid inclusion is illustrated by isocontours for VOF colour function values $\gamma = 0.25$ and $\gamma = 0.75$ in the x-y plane crossing through the centre of the domain.

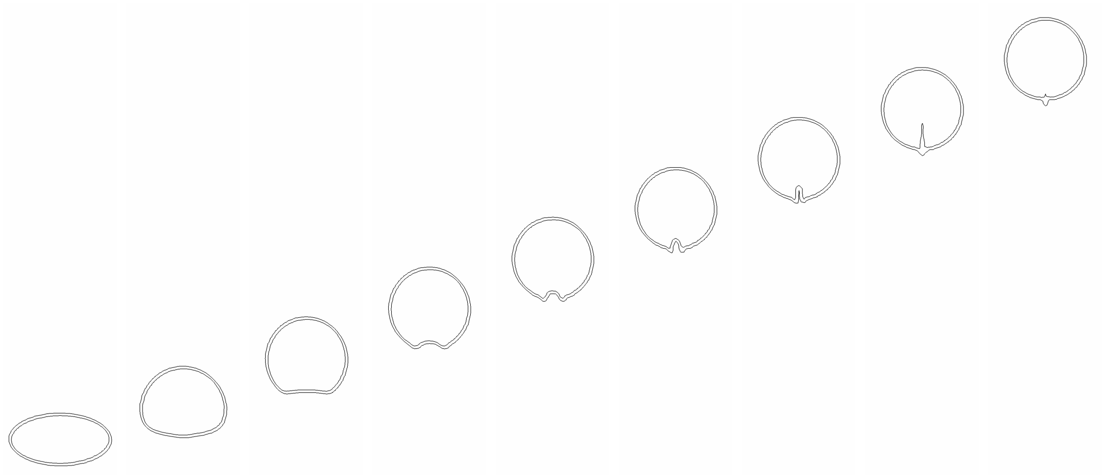

(a) $t = 0\tau$ (b) $t = 2\tau$ (c) $t = 4\tau$ (d) $t = 6\tau$ (e) $t = 8\tau$ (f) $t = 10\tau$ (g) $t = 12\tau$ (h) $t = 14\tau$ (i) $t = 16\tau$

Figure 6.12.: Shape evolution of the liquid inclusion of test case E ($Ca = 4.0$) at different time instants. The interface region of the liquid inclusion is illustrated by isocontours for VOF colour function values $\gamma = 0.25$ and $\gamma = 0.75$ in the x-y plane crossing through the centre of the domain.

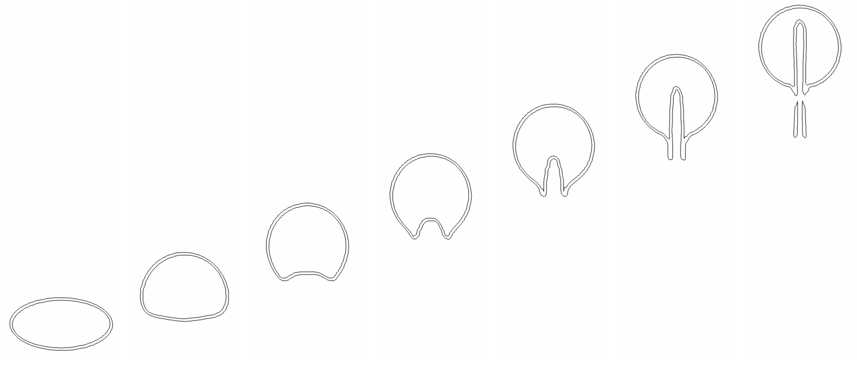
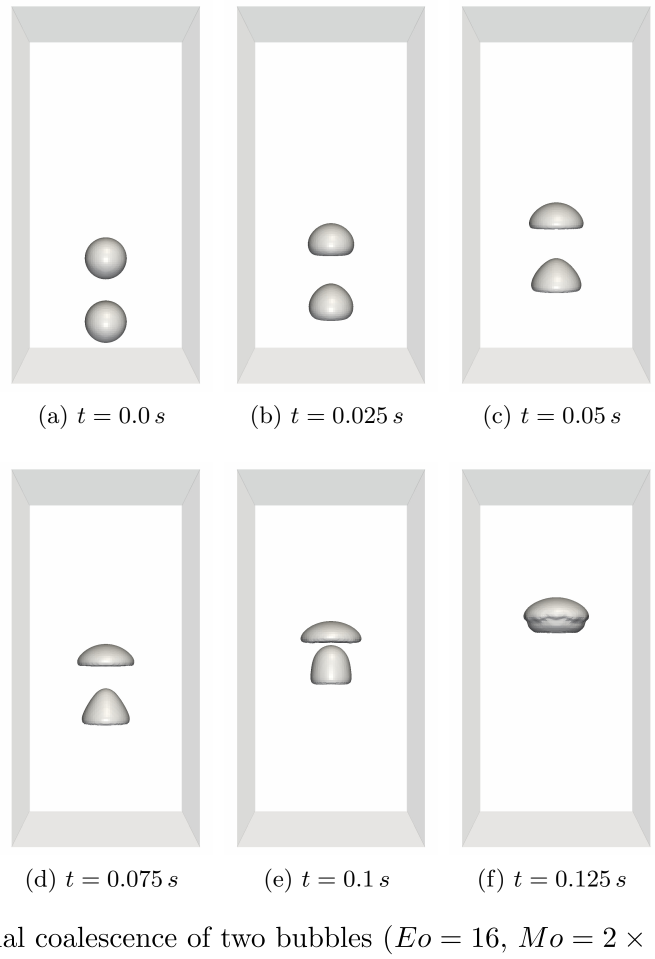

(a) $t = 0\tau$ (b) $t = 2\tau$ (c) $t = 4\tau$ (d) $t = 6\tau$ (e) $t = 8\tau$ (f) $t = 10\tau$ (g) $t = 12\tau$

Figure 6.13.: Shape evolution of the liquid inclusion of test case F ($Ca = 10.0$) at different time instants. The interface region of the liquid inclusion is illustrated by isocontours for VOF colour function values $\gamma = 0.25$ and $\gamma = 0.75$ in the x-y plane crossing through the centre of the domain.

6.4. Bubble Coalescence

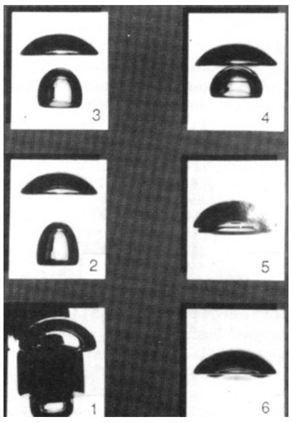
The coalescence of two bubbles of equal properties is simulated, following experiments conducted by Brereton and Korotney [21], reprinted in Figure 6.15. Both bubbles are initially spherical with a Morton number $Mo = 2 \times 10^{-4}$ and an Eötvös number $Eo = 16$. The bubbles are arranged either coaxially or obliquely and are driven by buoyancy only. The fluids are initially at rest and the domain is resolved by an equidistant Cartesian mesh with 20 cells per initial diameter. The numerical results of van Sint Annaland *et al.* [245] are reprinted in Figure 6.16 as an additional reference.

In the coaxial case, both bubbles are initially positioned coaxially with respect to the direction of gravity and the bubble centres are situated 1.5 diameter apart from each other. Figure 6.14 shows the evolution of both bubbles at different time instants. The trailing bubble quickly reduces the gap to the leading bubble as a result of the slip stream, whereas the leading bubble is not notably affected by the presence of the trailing bubble. The presented results are in excellent agreement with the experiments of Brereton and Korotney [21], reprinted in Figure 6.15a, and with the simulations of van Sint Annaland *et al.* [245], shown in Figure 6.16a, and Farhangi *et al.* [64]. It has to be noted, that in the experiment the initial shape of the bubbles is evidently not perfectly spherical.

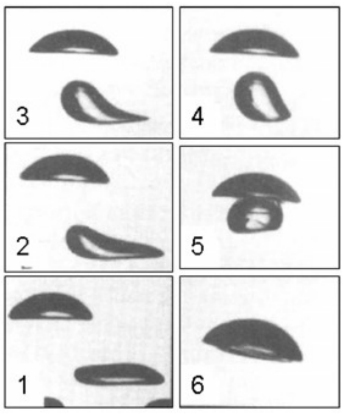

(a) $t = 0.0\,s$ (b) $t = 0.025\,s$ (c) $t = 0.05\,s$

(d) $t = 0.075\,s$ (e) $t = 0.1\,s$ (f) $t = 0.125\,s$

Figure 6.14.: Coaxial coalescence of two bubbles ($Eo = 16$, $Mo = 2 \times 10^{-4}$) rising due to gravity on an equidistant Cartesian mesh with a mesh resolution of 20 cells per initial bubble diameter.

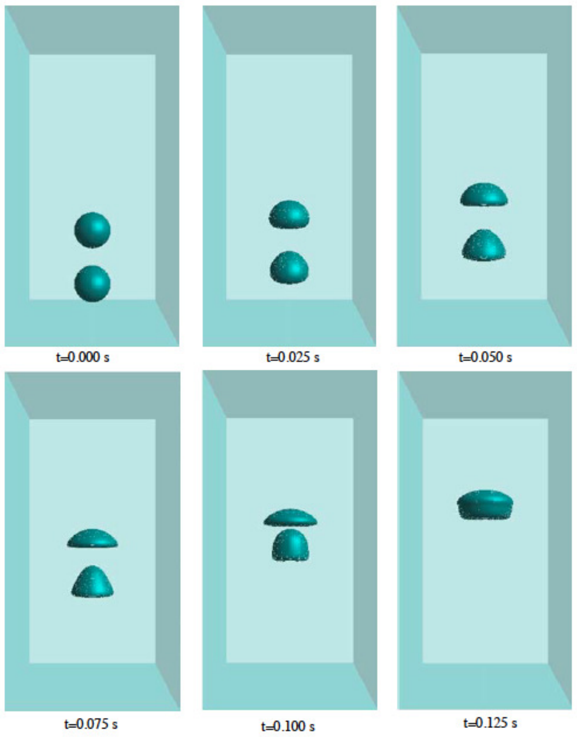


(a) Coaxial coalescence

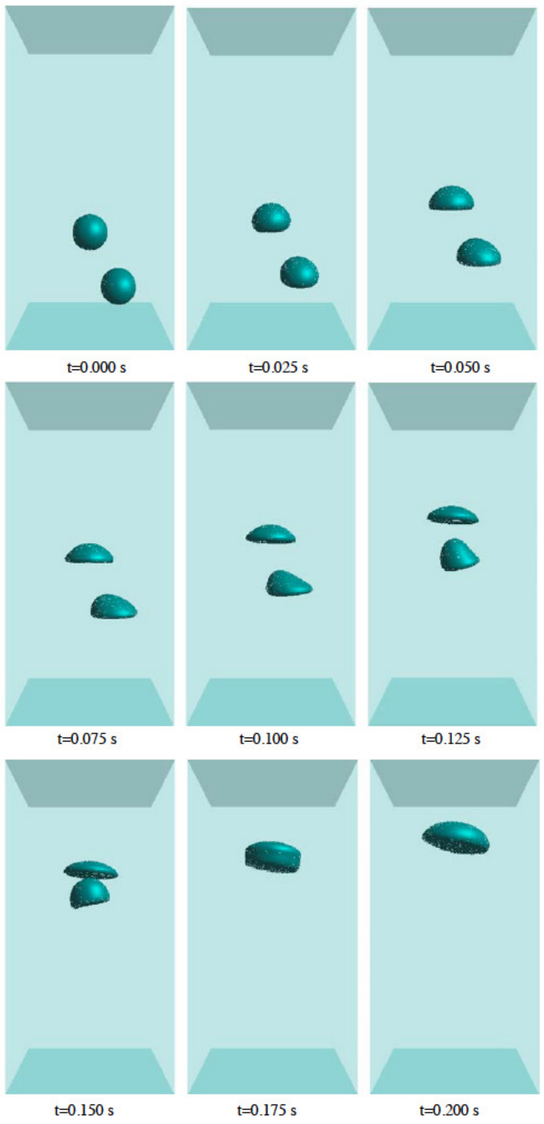


(b) Oblique coalescence

Figure 6.15.: Experimental results of coaxial and oblique bubble coalescence published by Brereton and Korotney [21]. The time interval between the individual photographs is 0.03 *s*. Reprinted from van Sint Annaland *et al.* [245], Copyright 2005, with permission from Elsevier.



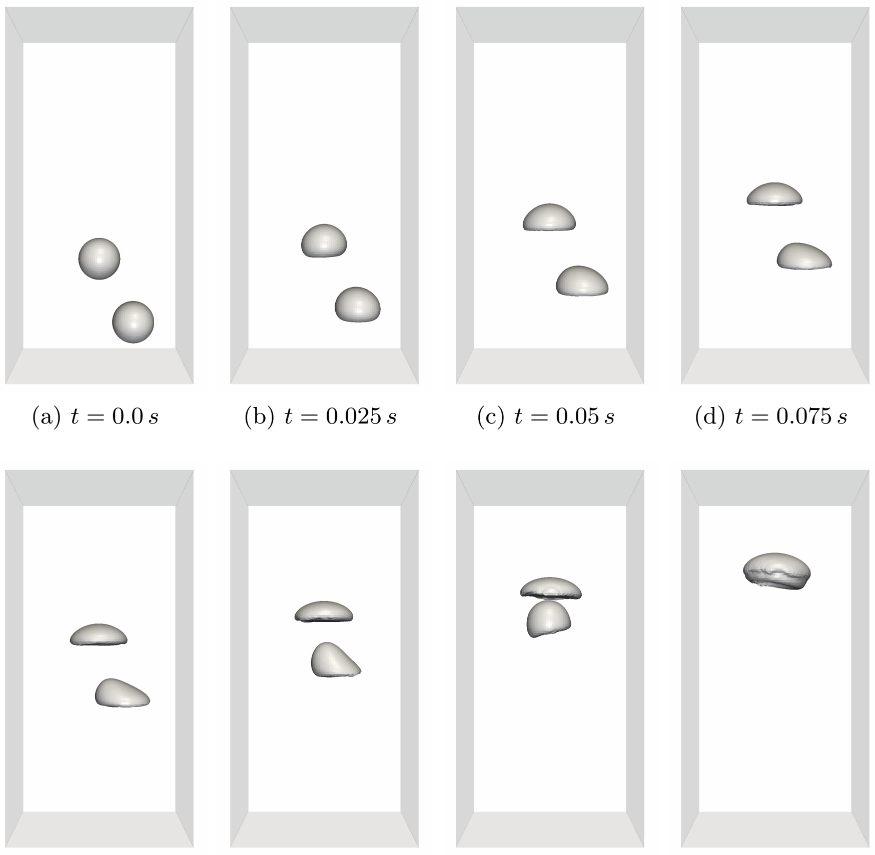
(a) Coaxial coalescence



(b) Oblique coalescence

Figure 6.16.: Numerical results of coaxial and oblique bubble coalescence published by van Sint Annaland *et al.* [245]. Reprinted from van Sint Annaland *et al.* [245], Copyright 2005, with permission from Elsevier.

As in the coaxial case, the centre of the bubbles in the oblique case are initially positioned with a distance of 1.5 diameter from each other with respect to the direction of gravity. The trailing bubble is shifted by 1.6 diameters in positive x-direction away from the coaxial position. After the bubbles have moved out of their initial position, the trailing bubble slowly moves into the slip stream of the leading bubble, as shown in Figure 6.17. The rise of the leading bubble seems not to be affected by the trailing bubble in the first $0.1\,s$. In subsequent time instants the leading bubble becomes successively more influenced by the trailing bubble, leaving its initially purely vertical trajectory. The results presented in Figure 6.17 are in good qualitative agreement with the experimental data shown in Figure 6.15b and the simulations of van Sint Annaland *et al.* [245], reprinted in Figure 6.16b, Marchandise *et al.* [148] and Singh and Shyy [217]. The bubbles in the experimental studies of Brereton and Korotney [21] appear not to be initially spherical, as suggested by Figure 6.15b.


(a) $t = 0.0\,s$ (b) $t = 0.025\,s$ (c) $t = 0.05\,s$ (d) $t = 0.075\,s$

(e) $t = 0.1\,s$ (f) $t = 0.125\,s$ (g) $t = 0.15\,s$ (h) $t = 0.175\,s$

Figure 6.17.: Oblique coalescence of two bubbles ($Eo = 16$, $Mo = 2 \times 10^{-4}$) rising due to gravity on an equidistant Cartesian mesh with a mesh resolution of 20 cells per initial bubble diameter.

6.5. Numerical Diffusion of Interface Advection

Considerable numerical diffusion of the colour function advection on tetrahedral meshes is observed in the simulation of a bubble rising due to buoyancy, presented in Section 6.2. Ubbink [238] suggested two possible reasons for numerical diffusion of the colour function using the CICSAM scheme, presented in Section 3.4.2. Firstly, the extrapolation of the upwind node, required to determine the advection of the colour function on unstructured meshes and, secondly, the implicit assumption that a face contains the interface if both adjacent cells contain an interface.

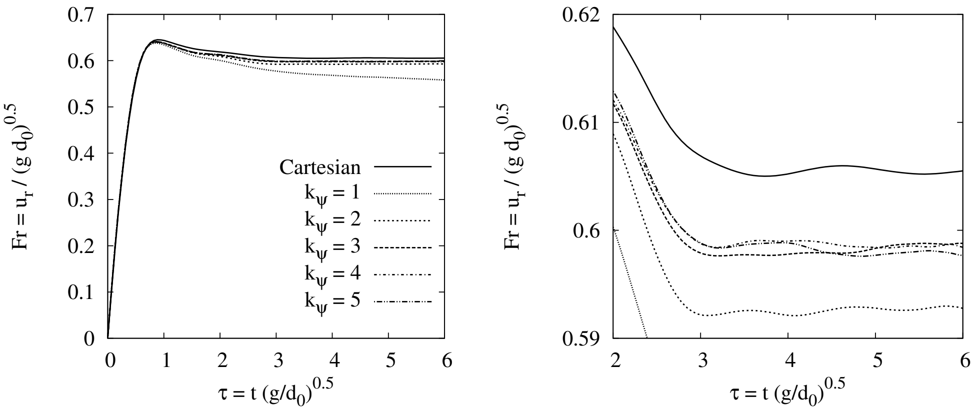


Figure 6.18.: Froude number of an initially spherical bubble rising due to buoyancy as a function of the non-dimensional time $\tau = t\sqrt{g/d_0}$, using different weighting factors k_ψ of the CICSAM interface advection scheme. The graph on the right is a closeup of the graph on the left, showing the Froude number region relevant to the terminal rise velocity.

One readily available measure to tackle numerical diffusion originating from the advection of the colour function is the application of a more compressive advection scheme. Increasing the coefficient k_ψ of the CICSAM scheme increases the dominance of the compressive Hyper-C scheme over the more diffusive UQ scheme (see also Section 3.4.2) and, therefore, makes the CICSAM scheme more compressive. Figure 6.18 shows the rise velocity of the bubble rising due to buoyancy presented in Section 6.2 on a tetrahedral mesh of approximately 1.4×10^6 cells, shown in Figure 6.3b, simulated with different CICSAM weighting coefficients k_ψ. The specific case of $k_\psi = 1$, as recommended by Ubbink and Issa [239], is the result previously presented in Section 6.2. For a sound comparison the result obtained on the equidistant Cartesian mesh, also previously presented in Section 6.2, is depicted as well. The rise velocity is considerably impacted by the CICSAM weighting coefficient in the range $1 \leq k_\psi < 3$ and shows only small differences for $k_\psi \geq 3$. The higher terminal rise velocity for $k_\psi \geq 2$ indicates a significantly reduced numerical diffusion and the stable rise velocity for $\tau > 3$ suggests that the accumulation of numerical diffusion is diminished after the bubble has assumed its terminal shape. Similar to the development of the rise velocity over time, the evolution of the bubble shape presented in Figure 6.19 does not show considerable difference for $k_\psi \geq 3$. The results indicate that, using the CICSAM scheme, a weighting coefficient of $k_\psi = 3$ reduces the numerical diffusion on tetrahedral meshes without the negative side-effects of a fully compressive scheme, *i.e.* wrinkling of the interface.

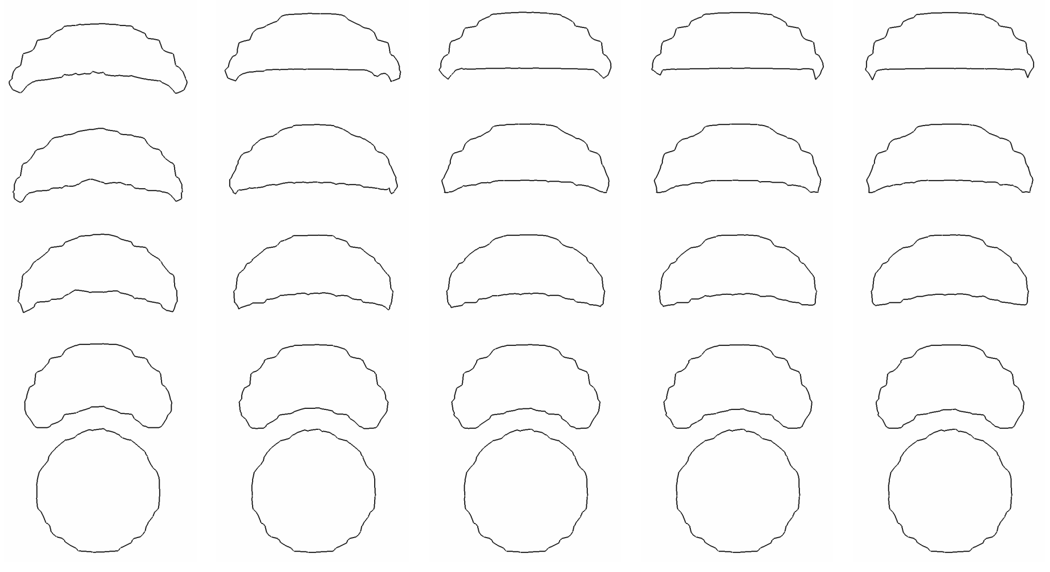

(a) $k_\psi = 1$ (b) $k_\psi = 2$ (c) $k_\psi = 3$ (d) $k_\psi = 4$ (e) $k_\psi = 5$

Figure 6.19.: Rise velocity and shape evolution of the rising bubble on a tetrahedral mesh using different weighting factors k_ψ of the CICSAM interface advection scheme. The bubble shapes are illustrated every 0.07 s ($\tau = 1.57$), starting from the initial position, in the x-y plane crossing through the centre of the domain.

6.6. Summary

The case studies discussed in this section have further demonstrated the capabilities of the presented numerical framework and general topics on two-phase flow modelling have been investigated. The influence of surface tension coefficient and interface curvature on parasitic currents has been examined in Section 6.1, whereby a non-linear rise of parasitic currents has been observed with increasing interface curvature. The results also indicate that Laplace number limits to determine the applicability of VOF methods should be used carefully in the presence of high interface curvature.

Overall, the results for the spherical cap bubble rising due to buoyancy, for the liquid inclusions at low Reynolds number and for the bubble coalescence obtained with the presented numerical framework are in excellent agreement with the respective experiments, with empirical data and other numerical studies. The rising bubble is predicted with very good accuracy on both Cartesian and tetrahedral meshes, as shown in Figure 6.4. On the tetrahedral mesh, however, numerical diffusion of the colour function adversely affects the rise velocity. Numerical experiments presented in Section 6.5 indicate that it is possible to counteract the numerical diffusion induced by tetrahedral meshes with more compressive interface advection schemes.

The simulation of the rising bubble discussed in Section 6.2 has also demonstrated that sophisticated compressive VOF methods are capable of transporting evolving interfaces with equal accuracy as more complex VOF-based interface reconstruction methods, such as PLIC. This is of particular significance as compressive VOF methods are often overlooked in the two-phase flow community, despite the inherent applicability to unstructured meshes and the straightforward implementation as presented in Sections 3.4 and 3.7.2.

The presented numerical framework is also capable of capturing complex capillary effects, as demonstrated with the liquid inclusions rising at low Reynolds number presented in Section 6.3. Given the applied mesh resolution, the prediction of the critical capillary number is in excellent agreement with results obtained by a boundary-element method more suited to the problem.

7. Conclusion

In this thesis, the fundamentals of two-phase flow modelling in general and on unstructured meshes in particular have been examined and the available methods have been further developed. The numerical modelling of two-phase flows is complicated by a number of difficulties due to the finite resolution of the numerical representation of the molecular interface between the phases. Most research efforts dedicated to two-phase flow modelling have been focused on Cartesian meshes, whereas the development of methods for unstructured meshes has frequently been neglected. As previously mentioned in the introduction of this thesis (Section 1.1), five particular difficulties related to two-phase flow modelling can be identified:

1. the definition of the force due to surface tension acting at the interface,
2. numerical instabilities as a result of the pressure jump across the interface,
3. the accurate evaluation of the interface curvature,
4. the advection of the sharp interface, and
5. the finite discrete resolution of the interface.

In order to further investigate the problems above and to advance the modelling of two-phase flows on unstructured meshes, the research presented in this thesis has predominantly focused on the following topics:

- the sources of parasitic currents in the vicinity of the interface,
- a numerical framework for the prediction of two-phase flows on unstructured meshes,
- the maintenance of the discrete balance between pressure gradient and body forces (force-balancing) by the numerical framework,
- the accurate evaluation of the interface curvature,
- the convolution of fluid properties at the interface,
- the application of adaptive tetrahedral meshes to two-phase flow simulations, and
- the identification of suitable test cases.

In what follows, the major contributions of this thesis are summarised and the main findings of the conducted studies are concluded. Lastly, suggestions and recommendations for future research are discussed.

7.1. Parasitic Currents

Parasitic currents, developing in the interface region as a result of the numerically approximated interface representation, are a major problem of two-phase flow modelling. The reduction of parasitic currents is of utmost importance as they represent a limiting factor for the applicability of numerical methods to model two-phase flows. Apart from deteriorating the result and altering the shape of the interface, parasitic currents can become large enough to destroy the interface. Two primary sources of parasitic currents have been identified: a) an imbalance of pressure gradient and body forces and b) an inaccurate estimation of the interface curvature.

The imbalance of pressure gradient and body forces can be resolved by algorithms that maintain an accurate force-balance on the discrete level, such as the numerical framework presented in Chapter 3 or the algorithms previously presented by Francois *et al.* [71] and Mencinger and Žun [154]. Imposing the exact curvature at an interface, parasitic currents vanish if the pressure gradient and the body forces are balanced on a discrete level, following previously published studies [71, 154] and the arguments discussed in Section 3.3. This principle has been demonstrated at a stationary interface in Section 3.8.4.1, as previously shown in similar studies by Francois *et al.* [71] and Mencinger and Žun [154]. In this thesis it has also been shown that this principle holds regardless of the mesh type as well as for moving interfaces on fixed (Section 3.8.4.2) and adaptive meshes (Section 5.7.3), scenarios which have not been studied before in the literature.

As concluded from the results presented in this thesis, in particular the analysis of the relationship between parasitic currents and interface properties in Section 6.1, another cause for parasitic currents are sudden variations, *i.e.* noise, in the magnitude of the interface curvature along the interface. An inaccurate average curvature estimate, on the other hand, has been found to alter the pressure jump across the interface, as observed in different cases presented in this thesis. It is, therefore, essential to diminish numerical noise when evaluating the interface curvature, in order to reduce parasitic currents. Similar conclusions have been drawn previously in other studies as well [34, 71, 262]. In Section 6.1 it has also been shown that parasitic currents are influenced differently by surface tension and interface curvature. While a linear correlation between surface tension coefficient and parasitic currents has been identified, a predominantly second-order rise of parasitic currents with increasing interface curvature has been observed. The second-order correlation results from a changing curvature magnitude and a changing error in curvature. The changing curvature error has been attributed to increasing aliasing errors, resulting from the smaller mesh spacing when resolving an interface of higher curvature with a mesh resolution fixed to the curvature radius.

7.2. Balanced-Force Numerical Framework

The fully-coupled numerical framework presented in Chapter 3 has been developed for two-phase flow modelling on unstructured meshes. Particular attention has been payed to maintaining the accurate balance between pressure gradient and body forces.

The fully-coupled implicit implementation of the numerical framework provides two distinct advantages for the application to two-phase flows. Firstly, the fully-coupled equation system ensures a strong pressure-velocity coupling, which is particularly important for two-phase flows with considerable surface tension because of the pressure jump across the interface. The results presented in Section 3.8.4.1 demonstrate that the numerical framework avoids pressure oscillations at the interface even for surface-tension-dominated interfacial flows with a very high density ratio of 10^9. Secondly, the implicit implementation of the fully-coupled system of equations increases the numerical stability compared to segregated solution methods [51]. As pointed out by Desjardins and Moureau [58] and Gueyffier *et al.* [86], the pressure Poisson equation used in segregated methods is ill-conditioned with respect to two-phase flows as a result of the discontinuous pressure, density and surface force fields. This may lead to numerical instabilities of segregated methods in flows with large surface tension or large density ratios.

The balance of pressure gradient and body forces has been achieved by a specifically constructed advecting velocity at face centres, derived using the momentum interpolation method, initially proposed by Rhie and Chow [198]. This advecting velocity has been used to define a continuity constraint which represents the fourth equation of the fully-coupled equation system. To assure force-balancing, it is essential that the pressure gradient and the source terms of the body forces are evaluated on the same computational stencil for the momentum equations as well as for the advecting velocity. It has been demonstrated that with the proposed numerical framework, force-balancing is achieved for stationary interfaces, as presented in Section 3.8.4.1, as well as for moving interfaces, as shown in Section 3.8.4.2. In Section 5.6, the presented force-balancing methodology has been successfully extended to adaptive meshes, as verified in Section 5.7.3.

It should be noted that the presented theory on force-balancing assumes that the contribution of velocity and body forces to the pressure gradient are cumulative, as explained in Sections 3.3.3 and 3.3.4 and defined for the example of gravity in Eq. 3.52. Therefore, the implementation of the discretisation of the pressure gradient and the body forces leading to force-balancing can only be proven to be exact in a stationary situation ($Re \to 0$). This is a justified assumption for gravity, or other volume forces such as electromagnetic forces, as it superimposes a pressure gradient on the flow field [13]. The force due to surface tension, on the other hand, is a surface force in reality acting at the molecular level, modelled numerically as a volume force by means of the CSF model, which is based on the Young-Laplace equation. The Young-Laplace equation as given in Eq. 2.46 is satisfied only for an interface in equilibrium, *i.e.* if the interface is in a state of minimal energy [19]. Thus, it is not fully clear what magnitude of error is introduced by the discretisation of the surface force in a dynamic situation ($Re \gg 1$), when the fluid acceleration terms are appreciable. This has also never been discussed or even raised in the literature. It is, nevertheless, very likely that a successful general implementation of the discretisation of the pressure gradient and the body forces must at least exactly ensure force-balancing in a stationary situation, such as the force-balancing methodology derived in this thesis.

A density-weighting of pressure gradient and body forces has been proposed in Section

3.3.4 as an addition to the momentum interpolation method to increase the stability of the numerical framework. Because the pressure term of the momentum equation is effectively weighted by density, the proposed density-weighting aligns the pressure term of the momentum equation with the pressure term of the advecting velocity. For the same reason and to maintain the discrete balance with the pressure term, the density-weighting is also applied to the acting body forces. As shown by the results presented in Section 3.8.4.1, the density-weighting assures convergence even for a density ratio as high as 10^9. On the other hand, performing the same simulations without the density-weighting of the pressure term and of the body forces in the advecting velocity equation has led to divergence of the numerical solver for a density ratio of 10^9.

In conclusion, the presented fully-coupled numerical framework is very robust, enabling the simulation of two-phase flows with high surface tension (*e.g.* $Ca = 10^{-2}$ in Sections 3.8.4.2 and 5.7) as well as high density ratios (*e.g.* $\rho_i/\rho_o = 10^9$ in Section 3.8.4), and maintains an accurate balance between pressure gradient, surface force and gravity force, eliminating a major source of errors in two-phase flow modelling.

7.3. Evaluation of the Interface Curvature

The inaccurate evaluation of the interface curvature has been identified as a fundamental source of error in two-phase flow computations by the work presented in this thesis as well as by previously published studies [19, 34, 71, 193, 263]. Inaccurate curvature estimates are a major source of parasitic currents and lead to an inaccurate prediction of the pressure jump across the interface. Significant research efforts have led to sophisticated methods to evaluate the interface curvature on Cartesian meshes, *e.g.* height function techniques, whereas the curvature evaluation on unstructured meshes has been limited to direct differentiation methods using standard finite difference or finite volume methods. As explained in Section 4.1, the direct differentiation of an abruptly varying scalar field results in so-called aliasing errors, known for instance from digital signal processing. For this reason, differentiating the abruptly varying colour function field to evaluate the interface curvature results in substantial errors. Convolution of the colour function with a suitable convolution kernel has been proven in several studies [34, 262] to reduce aliasing errors upon differentiation.

In Section 4.3 of this thesis, a novel method to evaluate the interface curvature has been presented. The new method, called CELESTE, is based on a least-squares fit of an overdefined equation system, constructed with a second-order Taylor series expansion of the colour function. The differentiation by means of a least-squares fit provides two important advantages compared to finite difference or finite volume methods. Firstly, the least-squares fit of an overdefined equation system damps out aliasing errors. Thus, convolution can be applied with smaller convolution length or can be discarded entirely. Secondly, the stencil size on which the least-squares fit is applied can be tuned. The method, therefore, has great potential for applications on adaptive meshes and for the application to interfaces with large curvature variations, such as bubbles where the rim has a significantly higher curvature than the rest of the interface. The CELESTE method significantly improves the accuracy of the interface curvature estimates compared to other direct differentiation methods. The accuracy of the results obtained with CELESTE are comparable to the results gained with methods that are limited to Cartesian or structured meshes, such as height-function techniques. Moreover, the predictive quality of the CELESTE method on structured versus unstructured meshes has shown to be similar, which is a considerable improvement with respect to the state-of-the-art for the application of unstructured meshes to simulate two-phase flows.

7.4. Convolution in Two-Phase Flow Modelling

As mentioned above, the convolution of the colour function, or any other interface indicator function for this purpose, is a common practice to improve curvature estimates. Convolution is also applied to smooth the sudden change of fluid properties at the interface as well as the discontinuous surface force. Despite the frequent application in two-phase flow modelling, convolution remains a controversial and important issue.

The study reported in Section 4.4 has demonstrated a substantial influence of the convolution of fluid properties and surface force on the outcome of two-phase flow computations. Contrary to the generally accepted notion, the convolution of the surface force does neither enhance the predictive quality of two-phase flow simulations nor does it improve the stability of the numerical framework. Furthermore, calculating the surface force based on the unconvoluted colour function results in a sharp pressure jump at the interface, thus, replicating reality most closely. The study has also demonstrated, that the convolution of density and viscosity reduces parasitic currents and improves the accuracy of the results. Although particularly the convolution of density seems counterintuitive from a physical viewpoint, the smooth momentum variation around the interface has been found to be a critical factor with regards to the predictive quality of the numerical framework. Lastly, the results presented in Section 4.4 have demonstrated that the numerical stability of the solution algorithm is considerably improved if density and viscosity are treated equally with respect to convolution.

Another issue of convolution is the applied convolution length, synonymous with the applied computational stencil of the convolution. A large convolution length potentially omits essential interface data whereas a small convolution length results in noisy curvature estimates and a steep momentum transition. Previously published studies occasionally applied very large convolution lengths to circular or spherical interfaces, some as large as the interface curvature radius. As the results in Section 4.5 as well as other studies [34, 71, 262, 263] have demonstrated, parasitic currents decrease significantly if the convolution length is increased. If, however, large convolution stencils are applied to realistic applications, such as the bubble rising due to buoyancy presented in Section 4.5, the results become significantly distorted or incorrect. Therefore, the convolution stencil should be kept as small as possible to ensure a realistic interface representation, and as large as necessary to provide a smooth momentum transition and accurate interface curvature estimates.

7.5. Adaptive Tetrahedral Meshes and Interfacial Flows

The application of adaptive mesh algorithms to two-phase flows is at present predominantly focused on Cartesian quadtree/octree meshes. The application of unstructured adaptive mesh algorithms in general and tetrahedral adaptive mesh algorithms in particular to two-phase flow simulations, on the other hand, is still premature and the literature on it is scarce. The predominant reason for the focus on quadtree/octree meshes in the two-phase flow community is presumably the lack of high-fidelity methods for two-phase flow modelling on unstructured meshes. The research conducted as part of this thesis has studied the fundamentals of the application of adaptive tetrahedral meshes to interfacial flows, focusing on three important issues:

1. the quantification of the required mesh resolution,

2. the force-balancing at moving interfaces on adaptive meshes (as mentioned in Section 7.2), and

3. the implementation of mesh adaption algorithms.

The definition of a reference length scale or suitable alternative error measure to determine the required mesh resolution in the vicinity of the interface is critical to the application of adaptive tetrahedral meshes to interfacial flows. Due to the practically infinitesimally thin interface, the interface is the prevailing flow feature with respect to the mesh resolution. A reference length scale for the interface region has been derived in Section 5.5, which is based on the local interface curvature and colour function gradient, including explicit bounds for the minimum and maximum interface thickness. The presented results demonstrate the applicability of the proposed reference length scale and show the potential of adaptive tetrahedral meshes for two-phase flows.

The implementation of adaptive (tetrahedral) mesh algorithms is critical with respect to the computational performance and, ultimately, the applicability of the methodology. In Section 5.4, an implementation approach for adaptive tetrahedral mesh algorithms, called *setwise-local*, has been presented. The setwise-local approach limits the application of the mesh adaption algorithm to a local subset of mesh cells. The limitation to a local subset of mesh cells simplifies the data management, the control of required memory resources and the extension to parallel computer architectures. The extension to parallel computer architectures requires only minor amendments to the interprocessor communication.

7.6. Test Cases for Two-Phase Flow Modelling

Suitable test cases to scrutinise and test the developed methods presented in this thesis had to be identified. The choice of test cases is essential for an accurate and meaningful assessment of numerical methods. The correct definition and informative value of test cases for two-phase flows is a frequently neglected issue in the relevant literature. For instance, although a circular or spherical interface is ideal to test the evaluation of curvature or the discrete balance between pressure gradient and body forces, it is not suitable to compare parasitic currents caused by different convolution stencils, as demonstrated in Section 4.5. Another example is a bubble rising due to buoyancy, as presented in Section 6.2, which is not suitable to quantify parasitic currents caused by an inexact curvature estimation because parasitic currents cannot be reliably identified in the dynamic flow field.

The test cases presented in this thesis each provides specific information. The inviscid static drop in equilibrium has been used for the evaluation of the error in interface curvature, the error in pressure jump across the interface and the parasitic currents caused by the new curvature evaluation method CELESTE, as presented in Section 4.3.5. Because the interface is spherical, the exact curvature and the exact pressure jump across the interface are readily available as a reference. Also, because both fluids are by definition initially at rest, all observed velocity magnitudes are parasitic currents. Simulating the same static drop with viscous fluids, as used in Sections 4.3.5.5 and 4.4.1, the kinetic energy induced by the parasitic currents can be examined. The magnitude and convergence of this parasitic kinetic energy are important characteristics of numerical methods for two-phase flows. The force-balancing of the presented numerical framework has been verified by simulating a surface-tension-dominated interface and applying the geometrically exact curvature to it, as presented in Sections 3.8.4 and 5.7.3. Thus, given the velocity gradients are negligible and no additional body force is acting, the pressure error and magnitude of the parasitic currents must be of the order of solver tolerance to prove a discrete balance between surface force and pressure gradient. The precise calculation and interaction of viscous stresses, buoyancy force and surface force are essential for the accurate prediction of the shape and the rise velocity of the spherical cap bubble rising due to buoyancy, examined in Sections 4.4.2, 4.5 and 6.2. The particular challenge of the liquid inclusions rising at low Reynolds number, presented in Section 6.3, is the accurate prediction of the critical capillary number beyond which the initially ellipsoidal inclusion does not restore its equilibrium shape. This test case represents a particularly severe validation of methods for two-phase flow modelling because the shape and stability of the liquid inclusion are strongly dependent on the correct prediction of the surface tension effects and the viscous stresses in proximity of the interface.

7.7. Recommendations for Future Research

In the following sections, suggestions and recommendations for future work are discussed based on the results and findings of this thesis.

7.7.1. Interface Advection Methods

According to the findings presented in this thesis, the numerical methods for force-balancing and for the evaluation of the interface curvature developed as part of the presented research, are practically independent of the mesh type. Nonetheless, the bubble rising due to buoyancy analysed in Section 6.2 shows differences in rise velocity between Cartesian and tetrahedral meshes. The differences have been attributed to the numerical diffusion resulting from the interface advection. This assumption has been further validated by customising the spatial interface advection scheme in Section 6.5. As pointed out by Ubbink [238], the extrapolation of the upwind node as well as the implicit assumption that an interface cuts through a mesh face if both adjacent cells contain an interface are possible reasons for the increased numerical diffusion observed on tetrahedral meshes. Additionally, interface advection schemes, such as CICSAM [239] or STACS [39], generally do not include a correction for skewness of the mesh faces. Increasing the mesh resolution in the vicinity of the interface reduces the errors origination from all three sources. Nevertheless, it is suggested to investigate the possibility of introducing a correction for face skewness to the interface advection scheme, in order to reduce the numerical diffusion on unstructured meshes. A skewness correction is suggested as it possibly increases the formal order of accuracy of the interface advection, reduces numerical diffusion and because of its typically straightforward implementation.

Alternatively, it is suggested to study the potential of PLIC methods with Lagrangian tracking of the reconstructed interfaces for unstructured meshes, particularly with respect to numerical diffusion. Several PLIC methods for two-dimensional unstructured meshes have recently been published [101, 155], demonstrating the applicability of PLIC methods to unstructured meshes. Yang and James [266] have derived a set of analytical formulations for PLIC interface reconstructions on triangular and tetrahedral meshes, which could serve as a basis for a new PLIC method for three-dimensional unstructured meshes.

7.7.2. Coupling of VOF Method and Lagrangian Particle Tracking

The application of VOF methods, or other interface capturing/tracking methods such as level-set or front-tracking methods, require the interface to be adequately resolved by the mesh, *i.e.* the interface curvature radius has to be at least twice the mesh spacing. Simulating flows such as the breakup of a liquid-jet, however, includes multiple physical scales. Resolving all of the involved scales adequately by the mesh to deploy, for instance, a VOF method is not possible for engineering applications. Shinjo and Umemura [212], for example, performed *Direct Numerical Simulations* (DNS) of the primary breakup of a liquid jet on a state-of-the-art supercomputing facility, using meshes with up to 6.9×10^9 cells. Despite the very high mesh resolution, capturing all scales of the secondary breakup is not possible.

The combination of interface capturing methods and Lagrangian particle tracking has great potential to enable multiscale two-phase flow simulations, as recent studies [90, 93, 95, 231] have demonstrated. Interface capturing methods are used to resolve large scale fluid bodies, such as a liquid jet, and Lagrangian particle tracking is used to simulate dispersed fluid particles which are not resolved by the Eulerian mesh. This methodology benefits from the fact that small fluid particles are more likely to be spherical as they are dominated by the acting surface force due to the high interface curvature. It would, therefore, be interesting to combine the numerical framework presented in this thesis with a Lagrangian particle tracking method to broaden the application range of the numerical framework and to further study the application of combined methods to multiscale two-phase flow problems. Future research should be particularly concerned with the accurate transition of fluid particles from the Eulerian to the Lagrangian framework and *vice versa*, building up on the work of Herrmann [93] and Tomar *et al.* [231].

7.7.3. Conservation Issue at Interfaces

The conservation issue at the interface, discussed in Section 2.3.4, represents an important problem for two-phase flows with high density ratios. Discretising the momentum equation in its conservative form results in an interface thickness that is dependent on the density ratio, which is physically implausible and violates the CSF method. However, as demonstrated by Raessi and Pitsch [189, 190], discretising the momentum equation it is non-conservative form introduces additional errors for flows with high-density ratios and, as a result, can severely distort the interface and lead to divergence of the numerical solving algorithm. Raessi and Pitsch [190] attributed these findings to the decoupling of mass conservation and momentum conservation. It is worth mentioning that Raessi and Pitsch used a segregated flow solver for their studies. In the fully-coupled numerical framework presented in Chapter 3, however, the momentum equations and the conservation of continuity, described by means of the continuity constraint formulated in Section 3.3, are implicitly coupled. The implications of the implicit coupling of momentum equations and continuity conservation for the discussed conservation issue at fluid-fluid interfaces and the potential to eliminate this issue numerically has not been part of the research presented in this thesis and is, therefore, recommended for future research. Diminishing the errors induced by the numerical description of momentum conservation and continuity conservation at interfaces would be a significant contribution to two-phase flow modelling.

7.7.4. Improving Computational Performance with GPUs

The solution of the fully-coupled implicit equation system as part of the numerical framework presented in Chapter 3 consumes the largest share of computational resources. The equation system is not only very large, *i.e.* four partial differential equations for each mesh cell, but also not well defined due to the pressure, density and viscosity discontinuity at the interface. Several studies [84, 164, 194] presented significant acceleration of two-phase flow simulations using *Graphics Processing Units* (GPUs) to solve the pressure Poisson equation of segregated flow solvers. Raessi *et al.* [194] presented a speed-up for three-dimensional two-phase flow simulations of factor 15 for single-core computations and factor $2.75 - 6$ for multi-core simulations, compared to performing the same task on an equal number of CPUs. The application of GPUs to solve the fully-coupled equation system of the presented numerical framework potentially increases the computational performance considerably, since its characteristics are very similar to the pressure Poisson equation from a numerical viewpoint. Conveniently, several freely-available software libraries are already providing support for GPUs, such as the PETSc library [11, 12] or the CUDA library [168]. It is, therefore, recommended to study the potential of GPUs for the solution of fully-coupled equation systems for two-phase flows.

7.7.5. Adaptive Computational Stencil for CELESTE

As mentioned above, the new curvature evaluation method CELESTE is applicable on variable stencil sizes, providing the opportunity to adapt the stencil size with respect to the ratio between local interface curvature and mesh resolution. The adaptation of the stencil size to the local interface curvature could assure a suitable stencil size when adaptive meshes are used as well as for severely deformed interfaces on fixed meshes. Using a large stencil potentially underestimates the curvature of high-curvature regions whereas a small stencil disregards essential data required to provide the best possible curvature estimate. An adaptive stencil for the evaluation of the interface curvature could, therefore, assure the most accurate curvature estimate is obtained with the available data. Thus, it is recommended to assess the potential of adaptive stencil sizes for the CELESTE method with respect to fixed and adaptive meshes.

7.7.6. Height Function Technique for Unstructured Meshes

As observed in the validation of the CELESTE method in Section 4.3.5, height function (HF) techniques generally provide more accurate curvature estimates, and as a result also typically more accurate predictions of the pressure jump across the interface, than direct differentiation methods, such as finite difference methods, finite volume methods or the proposed CELESTE method. To date, however, HF methods are limited to Cartesian meshes. It is, therefore, suggested to evaluate the possibility of extending height function techniques to unstructured meshes. The orientation-independent HF method for Cartesian meshes reported by Liovic *et al.* [129] is recommended as a basis for the extension to unstructured meshes.

Bibliography

[1] Afkhami, S. and Bussmann, M. (2009). Height functions for applying contact angles to 3D VOF simulations. *International Journal for Numerical Methods in Fluids*, **61**, 827–847.

[2] Ahipo, Y. M. and Traoré, P. (2009). A robust iterative scheme for finite volume discretization of diffusive flux on highly skewed meshes. *Journal of Computational and Applied Mathematics*, **231**(1), 478–491.

[3] Albadawi, A., Donoghue, D., Robinson, A., Murray, D., and Delauré, Y. (2013). Influence of surface tension implementation in Volume of Fluid and coupled Volume of Fluid with Level Set methods for bubble growth and detachment. *International Journal of Multiphase Flow*, **53**, 11–28.

[4] Aleinov, I. and Puckett, E. G. (1995). Computing surface tension with high-order kernels. In *Proc of the 6th Int Symp on Comput Fluid Dynamics*.

[5] Alhendal, Y., Turan, A., and Hollingsworth, P. (2013). Thermocapillary simulation of single bubble dynamics in zero gravity. *Acta Astronautica*, **88**, 108–115.

[6] Anderson, A., Zheng, X., and Cristini, V. (2005). Adaptive unstructured volume remeshing I: The method. *Journal of Computational Physics*, **208**(2), 616–625.

[7] Anderson, E., Bai, Z., Bischof, C., Blackford, S., Demmel, J., Dongarra, J., Du Croz, J., Greenbaum, A., Hammarling, S., McKenney, A., and Sorensen, D. (1999). *LAPACK Users's Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, third edition.

[8] Arnold, D., Mukherjee, A., and Pouly, L. (2000). Locally Adapted Tetrahedral Meshes Using Bisection. *SIAM Journal on Scientific Computing*, **22**(2), 431–448.

[9] Aulisa, E., Manservisi, S., Scardovelli, R., and Zaleski, S. (2007). Interface reconstruction with least-squares fit and split advection in three-dimensional Cartesian geometry. *Journal of Computational Physics*, **225**(2), 2301–2319.

[10] Baker, T. (2003). Mesh deformation and modification for time dependent problems. *International Journal for Numerical Methods in Fluids*, **43**, 747–768.

[11] Balay, S., Gropp, W. D., McInnes, L. C., and Smith, B. F. (1997). Efficient Management of Parallelism in Object Oriented Numerical Software Libraries. In E. Arge, A. Bruasat, and H. Langtangen, editors, *Modern Software Tools in Scientific Computing*, pages 163–202. Birkhaeuser Press.

[12] Balay, S., Brown, J., Buschelman, K., Eijkhout, V., Gropp, W., Kaushik, D., Knepley, M., McInnes, L., Smith, B., and Zhang, H. (2011). PETSc Users Manual.

[13] Batchelor, G. K. (1967). *An Introduction to Fluid Dynamics*. Cambridge University Press, New York.

[14] Benilov, E., Cummins, C., and Lee, W. (2012). Why do bubbles in Guinness sink? *arXiv (http://arxiv.org/abs/1205.5233)*, pages 1–5.

[15] Bern, M., Chew, P., Eppstein, D., and Ruppert, J. (1995). Dihedral Bounds for Mesh Generation in High Dimensions. In *Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '95*, pages 186–196.

[16] Bhaga, D. and Weber, M. (1981). Bubbles in viscous liquids: shapes, wakes and velocities. *Journal of Fluid Mechanics*, **105**, 61–85.

[17] Blanco-Alvarez, A. (1995). *Quelques aspects de l'écoulement d'un fluide visqueux autour d'une bulle déformable: une analyse par simulation directe*. Phd thesis, Institut National Polytechnique de Toulouse.

[18] Bornia, G., Cervone, a., Manservisi, S., Scardovelli, R., and Zaleski, S. (2011). On the properties and limitations of the height function method in two-dimensional Cartesian geometry. *Journal of Computational Physics*, **230**(4), 851–862.

[19] Brackbill, J. U., Kothe, D., and Zemach, C. (1992). Continuum Method for Modeling Surface Tension. *Journal of Computational Physics*, **100**, 335–354.

[20] Brady, P., Herrmann, M., and Lopez, J. (2012). Code verification for finite volume multiphase scalar equations using the method of manufactured solutions. *Journal of Computational Physics*, **231**(7), 2924–2944.

[21] Brereton, G. and Korotney, D. (1991). Coaxial and oblique coalescence of two rising bubbles. In I. Sahin and G. Tryggvason, editors, *Dynamics of bubbles and vortices near a free surface*. ASME, New York.

[22] Chen, Z. and Przekwas, A. J. (2010). A coupled pressure-based computational method for incompressible/compressible flows. *Journal of Computational Physics*, **229**(24), 9150–9165.

[23] Chew, L. (1989). Constraint Delaunay Triangulations. *Algorithmica*, **4**, 97–108.

[24] Choi, J.-H., Byun, K.-R., and Hwang, H.-J. (2003a). Quality-improved local refinement of tetrahedral mesh based on element-wise refinement switching. *Journal of Computational Physics*, **192**(1), 312–324.

[25] Choi, S. (1999). Note on the use of momentum interpolation method for unsteady flows. *Numerical Heat Transfer Part A*, **36**, 545–550.

[26] Choi, S., Kim, S., Lee, C.-H., and Choi, H.-K. (2003b). Use of the momentum interpolation method for flows with a large body force. *Numerical Heat Transfer Part B: Fundamentals*, **43**(3), 267–287.

[27] Clift, R., Grace, J., and Weber, M. (1978). *Bubbles, Drops and Particles*. Academic Press, New York.

[28] Compere, G., Marchandise, E., and Remacle, J.-F. (2008). Transient adaptivity applied to two-phase incompressible flows. *Journal of Computational Physics*, **227**, 1923–1942.

[29] Coyajee, E. and Boersma, B. J. (2009). Numerical simulation of drop impact on a liquid-liquid interface with a multiple marker front-capturing method. *Journal of Computational Physics*, **228**(12), 4444–4467.

[30] Cristini, V., Bawzdziewicz, J., and Loewenberg, M. (2001). An Adaptive Mesh Algorithm for Evolving Surfaces: Simulations of Drop Breakup and Coalescence. *Journal of Computational Physics*, **168**(2), 445–463.

[31] Cubero, A. and Fueyo, N. (2007). A Compact Momentum Interpolation Procedure for Unsteady Flows and Relaxation. *Numerical Heat Transfer, Part B: Fundamentals*, **52**(6), 507–529.

[32] Cubero, A. and Fueyo, N. (2008). Preconditioning Based on a Partially Implicit Implementation of Momentum Interpolation for Coupled Solvers. *Numerical Heat Transfer, Part B: Fundamentals*, **53**(6), 510–535.

[33] Cullum, J. (1971). Numerical differentiation and regularization. *SIAM Journal on Numerical Analysis*, **8**(2), 254–265.

[34] Cummins, S., Francois, M., and Kothe, D. (2005). Estimating curvature from volume fractions. *Computers & Structures*, **83**(6-7), 425–434.

[35] Dai, M. and Schmidt, D. (2005). Adaptive tetrahedral meshing in free-surface flow. *Journal of Computational Physics*, **208**(1), 228–252.

[36] Dalal, A., Eswaran, V., and Biswas, G. (2008). A Finite-Volume Method for Navier-Stokes Equations on Unstructured Meshes. *Numerical Heat Transfer, Part B: Fundamentals*, **54**(3), 238–259.

[37] Darwish, M. (2003). Development and Testing of a Robust Free-Surface Finite Volume Method. Technical report, Faculty of Engineering and Architecture, Amercian University of Beirut, Beirut, Lebanon.

[38] Darwish, M. and Moukalled, F. (2003). TVD schemes for unstructured grids. *International Journal of Heat and Mass Transfer*, **46**(4), 599–611.

[39] Darwish, M. and Moukalled, F. (2006). Convective Schemes for Capturing Interfaces of Free-Surface Flows on Unstructured Grids. *Numerical Heat Transfer Part B Fundamentals*, **49**(1), 19–42.

[40] Darwish, M., Sraj, I., and Moukalled, F. (2009a). A coupled finite volume solver for the solution of incompressible flows on unstructured grids. *Journal of Computational Physics*, **228**(1), 180–201.

[41] Darwish, M., Rached, J., and Moukalled, F. (2009b). Unstructured Adaptive Mesh Refinement and Coarsening for Fluid Flow at All Speeds Using a Coupled Solver. In T. Simos, G. Psihoyios, and C. Tsitouras, editors, *International Conference on Numerical Analysis and Applied Mathematics*, volume 577, pages 577–680, 1822 September 2009, Rethymno, Greece. American Institute of Physics.

[42] Darwish, M., Aziz, A., and Moukalled, F. (2010). A Coupled Finite Volume Solver for the Simulation of Disperse Multiphase Flows. In *7th International Conference on Multiphase Flows 2010*, Tampa, FL USA.

[43] de Berg, M., van Kreveld, M., Overmars, M., and Schwarzkopf, O. (2000). *Computational Geometry: Algorithms and Applications*. Springer, Berlin Heidelberg New York, second edi edition.

[44] De Cougny, H. and Shephard, M. (1999). Parallel refinement and coarsening of tetrahedral meshes. *International Journal for Numerical Methods in Fluids*, **46**, 1101–1125.

[45] de Oliveira, S. (2012). A Review on Delaunay Refinement Techniques. In *12th International Conference on Computational Science Its Applications (ICCSA 2012)*, pages 172–187, Salvador de Bahia, Brazil, June 18-21 2012.

[46] Deen, N., Van Sint Annaland, M., and Kuipers, J. (2004). Multi-scale modeling of dispersed gasliquid two-phase flow. *Chemical Engineering Science*, **59**(8-9), 1853–1861.

[47] Delage-Santacreu, S., Vincent, S., and Caltagirone, J.-P. (2009). Tracking Fronts in One and Two-phase Incompressible Flows Using an Adaptive Mesh Refinement Approach. *Journal of Scientific Computing*, **41**(2), 221–237.

[48] Dembo, R., Eisenstat, S., and Steihaug, T. (1982). Inexact newton methods. *SIAM Journal on Numerical Analysis*, **19**(2), 400–408.

[49] Demirdžić, I. (1982). *A Finite Volume Method for Computation of Fluid Flow in Complex Geometries*. Ph.D. thesis, Imperial College London.

[50] Demirdžić, I. and Muzaferija, S. (1995). Numerical method for coupled fluid flow, heat transfer and stress analysis using unstructured moving meshes with cells of arbitrary topology. *Computer Methods in Applied Mechanics and Engineering*, **125**(1-4), 235–255.

[51] Deng, G., Piquet, J., Vasseur, X., and Visonneau, M. (2001). A new fully coupled method for computing turbulent flows. *Computers and Fluids*, **30**(4), 445–472.

[52] Denner, F. (2009). *CFD Simulation of Emboli in a Cannulated Aorta*. Diploma thesis, Laboratory of Thermodynamics in Emerging Technologies, ETH Zurich.

[53] Denner, F. and van Wachem, B. (2012). Two-Phase Flow Modelling on Arbitrary Meshes: Superior VOF Curvature Estimation and the Issue of Convolution. In R. Kunz and A. Prosperetti, editors, *International Conference on Numerical Methods in Multiphase Flows*, 12-14 June 2012, State College, PA, USA.

[54] Denner, F. and van Wachem, B. (2013a). Force-balancing at moving surface-tension-dominated interfaces on collocated unstructured meshes. In *8th International Conference on Multiphase Flow (ICMF 2013)*, 26-31 May 2013, Jeju, Korea.

[55] Denner, F. and van Wachem, B. (2013b). Fully-coupled balanced-force VOF framework for arbitrary meshes with least-squares curvature evaluation from volume fractions. *Numerical Heat Transfer, Part B: Fundamentals, accepted for publication. DOI: 10.1080/10407790.2013.849996*.

[56] Denner, F. and van Wachem, B. (2013c). On the convolution of fluid properties and surface force for interface capturing methods. *International Journal of Multiphase Flow*, **54**, 61–64.

[57] Denner, F., van der Heul, D., Oud, G., Martins Villar, M., da Silveira Neto, A., and van Wachem, B. (2013). Comparative study of mass-conserving interface capturing frameworks for two-phase flows with surface tension. *Submitted to International Journal of Multiphase Flow on 27/06/2013*.

[58] Desjardins, O. and Moureau, V. (2010). Methods for multiphase flows with high density ratio. *Center for Turbulence Research Proceedings of the Summer Program 2010*, pages 313–322.

[59] Dey, T. and Slatton, A. (2011). Localized Delaunay Refinement for Volumes. *Computer Graphics Forum*, **30**(5), 1417–1426.

[60] Dey, T., Janoos, F., and Levine, J. (2012). Meshing interfaces of multi-label data with Delaunay refinement. *Engineering with Computers*, **28**, 71–82.

[61] Dupont, T. and Liu, Y. (2003). An explanation to the phenomonon of coupled first order shift of conservative quantities during the interaction of captured discontinuities. Technical report, School of Mathematics, Georgia Institute of Technologie, Atlanta, GA, USA.

[62] Dysthe, K., Krogstad, H., and Müller, P. (2008). Oceanic Rogue Waves. *Annual Review of Fluid Mechanics*, **40**(1), 287–310.

[63] Eiseman, P. (1987). Adpative Grid Generation. *Computer Methods in Applied Mechanics and Engineering*, **64**, 321–376.

[64] Farhangi, M. M., Passandideh-Fard, M., and Moin, H. (2010). Numerical study of bubble rise and interaction in a viscous liquid. *International Journal of Computational Fluid Dynamics*, **24**(1-2), 13–28.

[65] Farre, C., Perez-Segarra, C., Soria, M., and Oliva, A. (2006). Analysis of different numerical schemes for the resolution of convection-diffusion equations using finite-volume methods on three-dimensional unstructured grids. Part II: Numerical Analysis. *Numerical Heat Transfer Part B Fundamentals*, **49**, 351–375.

[66] Ferdowsi, P. A. and Bussmann, M. (2008). Second-order accurate normals from height functions. *J Comp Phys*, **227**(22), 9293–9302.

[67] Ferziger, J. (2003). Interfacial transfer in Tryggvason's method. *International Journal for Numerical Methods in Fluids*, **41**, 551–560.

[68] Ferziger, J. and Perić, M. (2002). *Computational Methods for Fluid Dynamics*. Springer Verlag, Berlin Heidelberg New York, 3. edition.

[69] Field, D. (1988). Laplacian smoothing and Delaunay triangulations. *Communications in Applied Numerical Methods*, **4**(6), 709–712.

[70] Field, D. (2000). Qualitative measures for initial meshes. *International Journal for Numerical Methods in Engineering*, **47**, 887–906.

[71] Francois, M., Cummins, S., Dendy, E., Kothe, D., Sicilian, J., and Williams, M. (2006). A balanced-force algorithm for continuous and sharp interfacial surface tension models within a volume tracking framework. *Journal of Computational Physics*, **213**(1), 141–173.

[72] Francois, M. M. and Swartz, B. K. (2010). Interface curvature via volume fractions, heights, and mean values on nonuniform rectangular grids. *Journal of Computational Physics*, **229**(3), 527–540.

[73] Freitag, L. and Ollivier-Gooch, C. (1996). A Comparison of Tetrahedral Mesh Improvement Techniques. In *Fifth International Meshing Roundtable*, pages 87–100.

[74] Freitag, L. A. and Ollivier-Gooch, C. (1997). Tetrahedral mesh improvement using swapping and smoothing. *International Journal for Numerical Methods in Engineering*, **40**(21), 3979–4002.

[75] Gaskell, P. and Lau, A. (1988). Curvature-compensated convective transport: SMART, A new boundedness- preserving transport algorithm. *International Journal for Numerical Methods in Fluids*, **8**(6), 617–641.

[76] Gerlach, D., Tomar, G., Biswas, G., and Durst, F. (2006). Comparison of volume-of-fluid methods for surface tension-dominant two-phase flows. *International Journal of Heat and Mass Transfer*, **49**(3-4), 740–754.

[77] Ghia, U., Ghia, K. N., and Shin, C. T. (1982). High-Re Solutions for Incompressible Flow Using the Navier-Stokes Equations and a Multigrid Method. *Journal of Computational Physics*, **48**, 387–411.

[78] Gibou, F., Chen, L., Nguyen, D., and Banerjee, S. (2007). A level set based sharp interface method for the multiphase incompressible Navier-Stokes equations with phase change. *Journal of Computational Physics*, **222**(2), 536–555.

[79] Gingold, R. and Monaghan, J. (1977). Smoothed particle hydrodynamics: theory and application to non-sperical stars. *Monthly Notices of the Royal Astronomical Society*, **181**, 375–389.

[80] Golias, N. and Dutton, R. (1997). Delaunay triangulation and 3D adaptive mesh generation. *Finite Elements in Analysis and Design*, **25**(3-4), 331–341.

[81] Gopala, V. R. and van Wachem, B. G. M. (2008). Volume of fluid methods for immiscible-fluid and free-surface flows. *Chemical Engineering Journal*, **141**(1-3), 204–221.

[82] Gosselin, S. and Ollivier-Gooch, C. (2011). Constructing Constrained Delaunay Tetrahedralizations of Volumes Bounded By Piecewise Smooth Surfaces. *International Journal of Computational Geometry & Applications*, **21**(5), 571–594.

[83] Greaves, D. (2004). A quadtree adaptive method for simulating fluid flows with moving interfaces. *Journal of Computational Physics*, **194**(1), 35–56.

[84] Griebel, M. and Zaspel, P. (2010). A multi-GPU accelerated solver for the three-dimensional two-phase incompressible Navier-Stokes equations. *Computer Science - Research and Development*, **25**(1-2), 65–73.

[85] Gross, S. and Reusken, A. (2005). Parallel multilevel tetrahedral grid refinement. *SIAM Journal on Scientific Computing*, **26**(4), 1261–1288.

[86] Gueyffier, D., Li, J., Nadim, A., Scardovelli, R., and Zaleski, S. (1999). Volume-of-fluid interface tracking with smoothed surface stress methods for three-dimensional flows. *Journal of Computational Physics*, **152**, 423–456.

[87] Habashi, W., Dompierre, J., Bourgault, Y., Ait-ali yahia, D., Fortin, M., and Vallet, M.-G. (2000). Anisotropic mesh adaptation: towards user-independent, mesh-independent and solver-independent CFD. Part I: general principles. *International Journal for Numerical Methods in Fluids*, **32**, 725–744.

[88] Harmathy, T. Z. (1960). Velocity of large drops and bubbles in media of infinite or restricted extent. *AIChE Journal*, **6**(2), 281–288.

[89] Haselbacher, A. and Vasilyev, O. (2003). Commutative discrete filtering on unstructured grids based on least-squares techniques. *Journal of Computational Physics*, **187**(1), 197–211.

[90] Hecht, N., Bouali, Z., Menard, T., Reveillon, J., and Demoulin, F. (2013). Towards fully coupled modelling of liquid interface and dispersed sprays. In *8th International Conference on Multiphase Flow (ICMF 2013)*, pages 1–6, 26-31 May 2013, Jeju, Korea.

[91] Hernandez, J., Lopez, P., Gomez, P., Zanzi, C., and Faura, F. (2008). A new volume of fluid method in three dimensions Part I : Multidimensional advection method with face-matched flux polyhedra. *International Journal for Numerical Methods in Fluids*, **58**, 897–921.

[92] Herrmann, M. (2008). A balanced force refined level set grid method for two-phase flows on unstructured flow solver grids. *Journal of Computational Physics*, **227**(4), 2674–2706.

[93] Herrmann, M. (2010). A parallel Eulerian interface tracking/Lagrangian point particle multi-scale coupling procedure. *Journal of Computational Physics*, **229**(3), 745–759.

[94] Herrmann, M. (2011). The influence of density ratio on the primary atomization of a turbulent liquid jet in crossflow. *Proceedings of the Combustion Institute*, **33**(2), 2079–2088.

[95] Herrmann, M. (2012). Parallel Eulerian Interface Tracking / Lagrangian Point Particle Multi-Scale Coupling for Atomization Simulations. In R. Kunz and A. Prosperetti, editors, *International Conference on Numerical Methods in Multiphase Flows*, 12-14 June 2012, State College, PA, USA.

[96] Heyns, J., Malan, A., Harms, T., and Oxtoby, O. (2013). Development of a compressive surface capturing formulation for modelling free-surface flow by using the volume-of-fluid approach. *International Journal for Numerical Methods in Fluids*, **71**, 788–804.

[97] Hirt, C. and Nichols, B. (1981). Volume of fluid (VOF) method for the dynamics of free boundaries. *Journal of Computational Physics*, **39**(1), 201–225.

[98] Hong, J.-M., Shinar, T., Kang, M., and Fedkiw, R. (2007). On Boundary Condition Capturing for Multiphase Interfaces. *Journal of Scientific Computing*, **31**(1-2), 99–125.

[99] Hua, J., Stene, J., and Lin, P. (2008). Numerical simulation of 3D bubbles rising in viscous liquids using a front tracking method. *Journal of Computational Physics*, **227**(6), 3358–3382.

[100] Huang, M., Chen, B., and Wu, L. (2010). A SLICVOF Method Based on Unstructured Grid. *Microgravity Science and Technology*, **22**(3), 305–314.

[101] Huang, M., Wu, L., and Chen, B. (2012). A Piecewise Linear Interface-Capturing Volume-of-Fluid Method Based on Unstructured Grids. *Numerical Heat Transfer Part B: Fundamentals*, **61**, 412–437.

[102] Hysing, S. (2006). A new implicit surface tension implementation for interfacial flows. *International Journal for Numerical Methods in Fluids*, **51**, 659–672.

[103] Issa, R. (1985). Solution of the implicitly discretised fluid flow equations by operator-splitting. *Journal of Computational physics*, **62**, 40–65.

[104] Ito, K., Kunugi, T., Ohshima, H., and Kawamura, T. (2009). Formulations and Validations of a High-Precision Volume-of-Fluid Algorithm on Nonorthogonal Meshes for Numerical Simulations of Gas Entrainment Phenomena. *Journal of Nuclear Science and Technology*, **46**(4), 366–373.

[105] Ito, K., Kunugi, T., and Ohshima, H. (2011). A high-precision unstructured adaptive mesh technique for gas-liquid two-phase flows. *International Journal for Numerical Methods in Fluids*, **67**, 1571–1589.

[106] Jasak, H. (1996). *Error Analysis and Estimation for the Finite Volume Method with Applications to Fluid Flow*. Ph.D. thesis, Imperial College London.

[107] Jasak, H. and Gosman, A. (2000). Automatic Resolution Control for the Finite-Volume Method, Part 2: Adaptive Mesh Refinement and Coarsening. *Numerical Heat Transfer, Part B: Fundamentals*, **38**(3), 257–271.

[108] Jasak, H. and Weller, H. (1995). Interface Tracking Capabilities of the Inter-Gamma Differencing Scheme. Technical report, Department of Mechnical Engineering, Imperial College London.

[109] Jones, D., Zou, Q., and Reeve, D. (2013). Computational modelling of coastal flooding caused by combined surge overflow and wave overtopping on embankments. *Journal of Flood Risk Management*, **6**, 70–84.

[110] Jouvet, G., Huss, M., Funk, M., and Blatter, H. (2011). Modelling the retreat of Grosser Aletschgletscher, Switzerland, in a changing climate. *Journal of Glaciology*, **57**(206), 1033–1045.

[111] Juretić, F. (2004). *Error Analysis in Finite Volume CFD*. Ph.D. thesis, Imperial College London.

[112] Juretić, F. and Gosman, A. (2010). Error Analysis of the Finite-Volume Method with Respect to Mesh Type. *Numerical Heat Transfer, Part B: Fundamentals*, **57**(6), 414–439.

[113] Karimian, S. and Straatman, A. (2006). Discretization and parallel performance of an unstructured finite volume NavierStokes solver. *International Journal for Numerical Methods in Fluids*, **52**(6), 591–615.

[114] Kharif, C. and Pelinovsky, E. (2003). Physical mechanisms of the rogue wave phenomenon. *European Journal of Mechanics B/Fluids*, **22**(6), 603–634.

[115] Kim, K., Yang, D., and Jeong, J. (2006). Adaptive refinement techniques based on tetrahedral and hexahedral grids for finite element analysis of mold filling in casting processes. *Computer Methods in Applied Mechanics and Engineering*, **195**(48-49), 6799–6821.

[116] Klingner, B. and Shewchuk, J. (2007). Aggressive Tetrahedral Mesh Improvement. In *Proceedings of the 16th International Meshing Roundtable*, pages 3–23.

[117] Koh, C. J. and Leal, L. G. (1989). The stability of drop shapes for translation at zero Reynolds number through a quiescent fluid. *Physics of Fluids A*, **1**, 1309–1313.

[118] Koh, C. J. and Leal, L. G. (1990). An experimental investigation on the stability of viscous drops translating through a quiescent fluid. *Physics of Fluids A*, **2**, 2103–2109.

[119] Kothe, D. (1998). Perspective on Eulerian Finite Volume Methods for Incompressible Interfacial Flows. In H. Kuhlmann and H. Rath, editors, *Free Surface Flows*, volume M, pages 267–331. Springer, Wien, New York.

[120] Kothe, D., Rider, W., Mosso, S., and Brock, J. (1996). Volume tracking of interfaces having surface tension in two and three dimensions. *AIAA 96-0859*, page 25.

[121] Kraus, M. and Ertl, T. (2000). Simplification of Nonconvex Tetrahedral Meshes. In *Electronic Proceedings of NSF/DoE Lake Tahoe Workshop for Scientific Visualization*.

[122] Labelle, F. (2007). *Tetrahedral Mesh Generation with Good Dihedral Angles Using Point Lattices*. Ph.D. thesis, University of California at Berkeley.

[123] Lafaurie, B., Nardone, C., Scardovelli, R., Zaleski, S., and Zanetti, G. (1994). Modelling merging and fragmentation in multiphase flows with SURFER. *Journal of Computational Physics*, **113**, 134–147.

[124] Lebaigue, O., Duquennoy, C., and Vincent, S. (2005). Test-case No 1: Rise of a spherical cap bubble in a stagnant liquid (PN). In H. Lemonnier, D. Jamet, and O. Lebaigue, editors, *Validation of Advanced Computational Methods for Multiphase Flow*, page 260. Begell House Inc.

[125] Lemmonier, H. and Hervieu, E. (2005). Test case No 2: Free rise of a liquid inclusion in a stagnant fluid. In H. Lemonnier, D. Jamet, and O. Lebaigue, editors, *Validation of Advanced Computational Methods for Multiphase Flow*, page 260. Begell House Inc.

[126] Leonard, B. P. (1991). The ULTIMATE conservative difference scheme applied to unsteady one-dimensional advection. *Computer Methods in Applied Mechanics and Engineering*, **88**(1), 17–74.

[127] Li, X. (2003). *Mesh modification procedures for general 3d non-manifold domains*. Ph.D. thesis, Rensselaer Polytechnic Institute.

[128] Li, X., Shephard, M., and Beall, M. (2005). 3D anisotropic mesh adaptation by mesh modification. *Computer Methods in Applied Mechanics and Engineering*, **194**(48-49), 4915–4950.

[129] Liovic, P., Francois, M., Rudman, M., and Manasseh, R. (2010). Efficient simulation of surface tension-dominated flows through enhanced interface geometry interrogation. *Journal of Computational Physics*, **229**(19), 7520–7544.

[130] Lisita, R. (2012). Personal Communication. Federal University of Uberlandia, Uberlandia, Brazil.

[131] Liu, A. and Joe, B. (1994). Relationship between tetrahedron shape measures. *BIT Numerical Mathematics*, **34**, 268–287.

[132] Löhner, R., Yang, C., and Oñate, E. (2006). On the simulation of flows with violent free surface motion. *Computer Methods in Applied Mechanics and Engineering*, **195**(41-43), 5597–5620.

[133] López, J. and Hernández, J. (2008). Analytical and geometrical tools for 3D volume of fluid methods in general grids. *Journal of Computational Physics*, **227**(12), 5939–5948.

[134] López, J. and Hernández, J. (2010). On reducing interface curvature computation errors in the height function technique. *Journal of Computational Physics*, **229**(13), 4855–4868.

[135] López, J., Zanzi, C., Gómez, P., Zamora, R., Faura, F., and Hernández, J. (2009). An improved height function technique for computing interface curvature from volume fractions. *Computer Methods in Applied Mechanics and Engineering*, **198**, 2555–2564.

[136] López, J., Gómez, P., and Hernández, J. (2010). A volume of fluid approach for crystal growth simulation. *Journal of Computational Physics*, **229**(19), 6663–6672.

[137] Lörstad, D. and Fuchs, L. (2004). High-order surface tension VOF-model for 3D bubble flows with high density ratio. *Journal of Computational Physics*, **200**(1), 153–176.

[138] Lorstad, D., Francois, M., Shyy, W., and Fuchs, L. (2004). Assessment of volume of fluid and immersed boundary methods for droplet computations. *International Journal for Numerical Methods in Fluids*, **46**(2), 109–125.

[139] Losasso, F., Fedkiw, R., and Osher, S. (2006). Spatially adaptive techniques for level set methods and incompressible flow. *Computers & Fluids*, **35**(10), 995–1010.

[140] Lubin, P., Vincent, S., Abadie, S., and Caltagirone, J.-P. (2006). Three-dimensional Large Eddy Simulation of air entrainment under plunging breaking waves. *Coastal Engineering*, **53**, 631–655.

[141] Lv, X., Zou, Q., Zhao, Y., and Reeve, D. (2010). A novel coupled level set and volume of fluid method for sharp interface capturing on 3D tetrahedral grids. *Journal of Computational Physics*, **229**(7), 2573–2604.

[142] Lv, X., Zou, Q., and Reeve, D. (2011). Numerical simulation of overflow at vertical weirs using a hybrid level set/VOF method. *Advances in Water Resources*, **34**(10), 1320–1334.

[143] Malik, M., Fan, E. S.-C., and Bussmann, M. (2007). Adaptative VOF with curvature-based refinement. *Int J Num Methods in Fluids*, **55**(April), 693–712.

[144] Mallouppas, G. (2010). Personal Communication. Imperial College London, London, United Kingdom.

[145] Manga, M. and Stone, H. (1995). Low Reynolds number motion of bubbles, drops and rigid spheres through fluidfluid interfaces. *Journal of Fluid Mechanics*, **287**, 279–298.

[146] Marchand, A., Weijs, J. H., Snoeijer, J. H., and Andreotti, B. (2011). Why is surface tension a force parallel to the interface? *American Journal of Physics*, **79**(10), 999–1008.

[147] Marchandise, E. and Remacle, J.-F. (2006). A stabilized finite element method using a discontinuous level set approach for solving two phase incompressible flows. *Journal of Computational Physics*, **219**(2), 780–800.

[148] Marchandise, E., Geuzaine, P., Chevaugeon, N., and Remacle, J.-F. (2007). A stabilized finite element method using a discontinuous level set approach for the computation of bubble dynamics. *Journal of Computational Physics*, **225**(1), 949–974.

[149] Marić, T., Marschall, H., and Bothe, D. (2013). On the Adaptive Mesh Refinement for a 3D Geometrical Volume of Fluid Transport Algorithm on Unstructured Meshes using OpenFOAM. In *8th International Conference on Multiphase Flow (ICMF 2013)*, number 2011, pages 1–10, 26-31 May 2013, Jeju, Korea.

[150] Maric, T., Marschall, H., and Bothe, D. (2013). voFoam A geometrical Volume of Fluid algorithm on arbitrary unstructured meshes with local dynamic adaptive mesh refinement using OpenFOAM. *arXiv (http://arxiv.org/abs/1305.3417)*, pages 1–30.

[151] Martins Villar, M. (2013). Personal Communication. Federal University of Uberlandia, Uberlandia, Brazil.

[152] Mathur, S. and Murthy, J. (1997). A pressure-based method for unstructured meshes. *Numerical Heat Transfer Part B Fundamentals*, **31**, 195–215.

[153] Mavriplis, D. (2003). Revisiting the Least-squares Procedure for Gradient Reconstruction on Unstructured Meshes. Technical report, NASA/CR-2003-212683 and NIA Report No. 2003-06, National Institute of Aerospace, Hampton, VA, USA.

[154] Mencinger, J. and Žun, I. (2007). On the finite volume discretization of discontinuous body force field on collocated grid: Application to VOF method. *Journal of Computational Physics*, **221**(2), 524–538.

[155] Mencinger, J. and Žun, I. (2011). A PLICVOF method suited for adaptive moving grids. *Journal of Computational Physics*, **230**(3), 644–663.

[156] Menzies, K. (2009). *Grid Adaptation for Gas Turbine Combustor Calculations*. Ph.D. thesis, Imperial College London.

[157] Monaghan, J. (1992). Smoothed Particle Hydrodynamics. *Annual Review of Astronomy and Astrophysics*, **30**(1), 543–574.

[158] Monaghan, J. and Lattanzio, J. (1985). A refined particle method for astrophysical problems. *Astronomy and Astrophysics*, **149**, 135–143.

[159] Montazeri, H., Bussmann, M., and Mostaghimi, J. (2012). Accurate implementation of forcing terms for two-phase flows into SIMPLE algorithm. *International Journal of Multiphase Flow*, **45**, 40–52.

[160] Morris, J. P., Fox, P. J., and Zhu, Y. (1997). Modeling Low Reynolds Number Incompressible Flows Using SPH. *Journal of Computational Physics*, **136**(1), 214–226.

[161] Moukalled, F. and Darwish, M. (2012). Transient Schemes for Capturing Interfaces of Free-Surface Flows. *Numerical Heat Transfer, Part B: Fundamentals*, **61**(3), 171–203.

[162] Muzaferija, S. (1994). *Adaptive finite volume method for flow predictions using unstructured meshes and multigrid approach*. Ph.D. thesis, Imperial College London.

[163] Muzaferija, S., Perić, M., Sames, P., and Schellin, P. (1998). A Two-Fluid Navier-Stokes Solver to Simulate Water Entry. In *Proceedings of the Twenty-Second Symposium on Naval Hydrodynamics*, pages 638–649, Washington DC, USA.

[164] Nagatake, T. and Kunugi, T. (2010). Application of GPU to computational multiphase fluid dynamics. *IOP Conference Series: Materials Science and Engineering*, **10**, 012024.

[165] Nakahashi, L. and Deiwert, G. (1984). A Practical Adaptive-Grid Method for Complex Fluid-Flow Problems. Technical report, NASA Technical Memorandum 85989, NASA Ames Research Center, Moffett Field, CA 94035, USA.

[166] Navascues, G. (1979). Liquid surfaces: theory of surface tension. *Reports on Progress in Physics*, **42**, 1131 – 1186.

[167] Nordmark, H. (1991). Rezoning for higher order vortex methods. *Journal of Computational Physics*, **97**(2), 366–397.

[168] NVIDIA Corporation (2012). CUDA C Programming Guide v5.0.

[169] Osher, S. and Sethian, J. A. (1988). Fronts Propagating with Curvature-Dependent Speed: Algorithms based on the Hamilton-Jacobi Formulation. *Journal of Computational Physics*, **79**, 12–49.

[170] Otsu, N. (1979). A Threshold Selection Method from Gray-Level Histograms. *IEEE Transactions on Systems, Man and Cybernetics*, **9**, 62–66.

[171] Park, I., Kim, K., and Kim, J. (2009). A volume-of-fluid method for incompressible free surface flows. *International Journal for Numerical Methods in Fluids*, **61**, 1331–1362.

[172] Park, I., Kim, K., Kim, J., and Van, S. (2012). Numerical investigation of the effects of turbulence intensity on dam-break flows. *Ocean Engineering*, **42**, 176–187.

[173] Pascau, A. (2011). Cell face velocity alternatives in a structured colocated grid for the unsteady Navier-Stokes equations. *International Journal for Numerical Methods in Fluids*, **65**, 812–833.

[174] Patankar, S. (1980). *Numerical Heat Transfer and Fluid Flow*. Hemisphere Publishing Company.

[175] Patankar, S. and Baliga, B. (1978). A new finite-difference scheme for parabolic differential equations. *Numerical Heat Transfer Part B: Fundamentals*, **1**(1), 27–37.

[176] Patankar, S. and Spalding, D. (1972). A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows. *International Journal of Heat and Mass Transfer*, **15**, 1787–1806.

[177] Perez-Segarra, C., Farre, C., Cadafalch, J., and Oliva, A. (2006). Analysis of different numerical schemes for the resolution of convection-diffusion equations using finite-volume methods on three-dimensional unstructured grids. Part I: Discretisation schemes. *Numerical Heat Transfer Part B Fundamentals*, **49**, 333–350.

[178] Perot, B. and Nallapati, R. (2003). A moving unstructured staggered mesh method for the simulation of incompressible free-surface flows. *Journal of Computational Physics*, **184**(1), 192–214.

[179] Peskin, C. S. (1977). Numerical analysis of blood flow in the heart. *Journal of Computational Physics*, **25**(3), 220–252.

[180] Pilliod, J. E. and Puckett, E. G. (2004). Second-Order Accurate Volume-of-Fluid algorithms for tracking material interfaces. *J Comp Phys*, **199**(2), 465–502.

[181] Plaza, A., Márquez, A., Moreno-González, A., and Suárez, J. (2009). Local refinement based on the 7-triangle longest-edge partition. *Mathematics and Computers in Simulation*, **79**(8), 2444–2457.

[182] Poo, J. and Ashgriz, N. (1989). A computational method for determining curvatures. *Journal of Computational Physics*, **84**, 483–491.

[183] Popinet, S. (2009). An accurate adaptive solver for surface-tension-driven interfacial flows. *Journal of Computational Physics*, **228**(16), 5838–5866.

[184] Popinet, S. and Zaleski, S. (1999). A front-tracking algorithm for accurate representation of surface tension. *International Journal for Numerical Methods in Fluids*, **30**, 775–793.

[185] Prosperetti, A. and Tryggvason, G. (2007). Introduction: A computational approach to multiphase flows. In A. Prosperetti and G. Tryggvason, editors, *Computational Methods for Multiphase Flow*, pages 1–19. Cambridge University Press, Cambridge, UK.

[186] Pueyo, A. and Zingg, D. (1997). An efficient Newton-GMRES solver for aerodynamic computations. In *13th AIAA Computational Fluid Dynamics Conference*.

[187] Quan, S. and Schmidt, D. (2007). A moving mesh interface tracking method for 3D incompressible two-phase flows. *Journal of Computational Physics*, **221**(2), 761–780.

[188] Raessi, M. (2012). Personal Communication. University of Massachusetts-Dartmouth, Dartmouth, MA, USA.

[189] Raessi, M. and Pitsch, H. (2009). Modeling interfacial flows characterized by large density ratios with the level set method. *Center for Turbulence Research Annual Research Briefs*, pages 159–169.

[190] Raessi, M. and Pitsch, H. (2012). Consistent mass and momentum transport for simulating incompressible interfacial flows with large density ratios using the level set method. *Computers & Fluids*, **63**, 70–81.

[191] Raessi, M., Mostaghimi, J., and Bussmann, M. (2007). Advecting normal vectors: A new method for calculating interface normals and curvatures when modeling two-phase flows. *Journal of Computational Physics*, **226**(1), 774–797.

[192] Raessi, M., Bussmann, M., and Mostaghimi, J. (2009). A semi-implicit finite volume implementation of the CSF method for treating surface tension in interfacial flows. *International Journal for Numerical Methods in Fluids*, **59**, 1093–1110.

[193] Raessi, M., Mostaghimi, J., and Bussmann, M. (2010). A volume-of-fluid interfacial flow solver with advected normals. *Computers & Fluids*, **39**(8), 1401–1410.

[194] Raessi, M., Pathak, A., Mostaghimi, J., and Bussmann, M. (2012). On the accuracy and performance of the advected normals approach in simulations of interfacial flows. In R. Kunz and A. Prosperetti, editors, *International Conference on Numerical Methods in Multiphase Flows*, number June, 12-14 June 2012, State College, PA, USA.

[195] Rahman, M., Miettinen, A., and Siikonen, T. (1996). Modified SIMPLE formulation on a collocated grid with an assessment of the simplified QUICK scheme. *Numerical Heat Transfer Part B: Fundamentals*, **30**(3), 291–314.

[196] Remaki, L. and Habashi, W. (2006). 3-d mesh adaption on multiple weak discontinuities and boundary layers. *SIAM Journal on Scientific Computing*, **28**(4), 1379–1397.

[197] Renardy, Y. and Renardy, M. (2002). PROST: A Parabolic Reconstruction of Surface Tension for the Volume-of-Fluid Method. *Journal of Computational Physics*, **183**(2), 400–421.

[198] Rhie, C. M. and Chow, W. L. (1983). Numerical study of the turbulent flow past an airfoil with trailing edge separation. *AIAA Journal*, **21**(11), 1525–1532.

[199] Rider, W. J. and Kothe, D. B. (1998). Reconstructing Volume Tracking. *Journal of Computational Physics*, **141**(2), 112–152.

[200] Rudman, M. (1998). A volume-tracking method for incompressible multifluid flows with large density variations. *International Journal for Numerical Methods in Fluids*, **28**(1998), 357–378.

[201] Ryskin, G. and Leal, L. (1984). Numerical solution of free-boundary problems in fluid mechanics. Part 2. Buoyancy-driven motion of a gas bubble through a quiescent liquid. *Journal of Fluid Mechanics*, **148**, 19–35.

[202] Sadhal, S., Ayyaswamy, P., and Chung, J. (1997). *Transport Phenomena with Drops and Bubbles*. Springer Verlag, New York.

[203] Saghi, H., Ketabdari, M., and Zamirian, M. (2013). A novel algorithm based on parameterization method for calculation of curvature of the free surface flows. *Applied Mathematical Modelling*, **37**, 570–585.

[204] Sahni, O., Jansen, K., Shephard, M., Taylor, C., and Beall, M. (2008). Adaptive boundary layer meshing for viscous flow simulations. *Engineering with Computers*, **24**(3), 267–285.

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

[206] Scriven, L. (1960). Dynamics of a fluid interface: equation of motion for Newtonian surface fluids. *Chemical Engineering Science1*, **12**(2), 98–108.

[207] Shewchuk, J. (1997). *Delaunay Refinement Mesh Generation*. Ph.D. thesis, Carnegie Mellon University, Pittsburgh, PA, USA.

[208] Shewchuk, J. (1998). Tetrahedral Mesh Generation by Delaunay Refinement. In *SCG 98: Proceedings of the Fourteenth Annual Symposium on Computational Geometry*, volume 4, pages 86–95, New York, NY, USA.

[209] Shewchuk, J. (2002a). Two Discrete Optimization Algorithms for the Topological Improvement of Tetrahedral Meshes. Technical report, University of California at Berkeley, Berkeley, CA, USA.

[210] Shewchuk, J. (2002b). What Is a Good Linear Finite Element? Interpolation, Conditioning, Anisotropy, and Quality Measures. Technical report, University of California at Berkeley, Berkeley, CA, USA.

[211] Shewchuk, J. (2004). Stabbing Delaunay Tetrahedralizations. *Discrete & Computational Geometry*, **32**(3), 339–343.

[212] Shinjo, J. and Umemura, A. (2010). Simulation of liquid jet primary breakup: Dynamics of ligament and droplet formation. *International Journal of Multiphase Flow*, **36**(7), 513–532.

[213] Si, H. (2008a). Adaptive tetrahedral mesh generation by constrained Delaunay refinement. *International Journal for Numerical Methods in Engineering*, **75**, 856–880.

[214] Si, H. (2008b). *Three Dimensional Boundary Conforming Delaunay Mesh Generation*. Ph.D. thesis, Technische Universitaet Berlin, Berlin, Germany.

[215] Si, H. (2009). TetGen: A Quality Tetrahedral Mesh Generator and a 3D Delaunay Triangulator.

[216] Si, H. (2010). Constrained Delaunay tetrahedral mesh generation and refinement. *Finite Elements in Analysis and Design*, **46**(1-2), 33–46.

[217] Singh, R. and Shyy, W. (2007). Three-dimensional adaptive Cartesian grid method with conservative interface restructuring and reconstruction. *Journal of Computational Physics*, **224**(1), 150–167.

[218] Sleijpen, G. and van der Vorst, H. (1995). An overview of approaches for the stable computation of hybrid BiCG methods. *Applied Numerical Mathematics*, **19**(3), 235–254.

[219] Son, G. (2003). Efficient implementation of a coupled level-set and volume-of-fluid method for three-dimensional incompressible two-phase flows. *Numerical Heat Transfer Part B Fundamentals*, **43**(43), 549–565.

[220] Stickel, J. (2010). Data smoothing and numerical differentiation by a regularization method. *Computers & Chemical Engineering*, **34**(4), 467–475.

[221] Stone, H. (1994). Dynamics of drop deformation and breakup in viscous fluids. *Annual Review of Fluid Mechanics*, **26**, 65–102.

[222] Sun, D. L. and Tao, W. Q. (2010). A coupled volume-of-fluid and level set (VOSET) method for computing incompressible two-phase flows. *International Journal of Heat and Mass Transfer*, **53**(4), 645–655.

[223] Sussman, M. (2000). A Coupled Level Set and Volume-of-Fluid Method for Computing 3D and Axisymmetric Incompressible Two-Phase Flows. *Journal of Computational Physics*, **162**(2), 301–337.

[224] Sussman, M. and Fatemi, E. (1999). An efficient, interface-preserving level set redistancing algorithm and its application to interfacial incompressible fluid flow. *SIAM Journal on Scientific Computing*, **20**(4), 1165–1191.

[225] Sussman, M., Smereka, P., and Osher, S. (1994). A Level Set Approach for Computing Solutions to Incompressible Two-Phase Flow. *Journal of Computational Physics*, **114**(1), 146–159.

[226] Sussman, M., Almgren, A., Bell, J., Colella, P., Howell, L., and Welcome, M. L. (1999). An Adaptive Level Set Approach for Incompressible Two-Phase Flows. *Journal of Computational Physics*, **148**, 81–124.

[227] Sussman, M., Smith, K., Hussaini, M., Ohta, M., and Zhiwei, R. (2007). A sharp interface method for incompressible two-phase flows. *Journal of Computational Physics*, **221**(2), 469–505.

[228] Tam, A. (1998). *An Anisotropic Adaptive Method for the Solution of 3-D Inviscid and Viscous Compressible Flows*. Ph.D. thesis, Concordia University, Montreal, Quebec, Canada.

[229] Tam, A., Ait-ali yahia, D., Robichaud, M., Moore, M., Kozel, V., and Habashi, W. (2000). Anisotropic mesh adaptation for 3D ows on structured and unstructured grids. *Computational Methods in Applied Mechanical Engineering*, **189**, 1205–1230.

[230] Theodorakakos, A. and Bergeles, G. (2004). Simulation of sharp gasliquid interface using VOF method and adaptive grid local refinement around the interface. *International Journal for Numerical Methods in Fluids*, **45**, 421–439.

[231] Tomar, G., Fuster, D., Zaleski, S., and Popinet, S. (2010). Multiscale simulations of primary atomization. *Computers & Fluids*, **39**(10), 1864–1874.

[232] Tornberg, A.-K. and Engquist, B. (2003). Regularization Techniques for Numerical Approximation of PDEs with Singularities. *Journal of Scientific Computing*, **19**(1-3), 527–552.

[233] Tornberg, A.-K. and Engquist, B. (2004). Numerical approximations of singular source terms in differential equations. *Journal of Computational Physics*, **200**(2), 462–488.

[234] Traoré, P., Ahipo, Y. M., and Louste, C. (2009). A robust and efficient finite volume scheme for the discretization of diffusive flux on extremely skewed meshes in complex geometries. *Journal of Computational Physics*, **228**(14), 5148–5159.

[235] Tryggvason, G., Bunner, B., Esmaeeli, A., Juric, D., Al-Rawahi, N., Tauber, W., Han, J., Nas, S., and Jan, Y. (2001). A front-tracking method for the computations of multiphase flow. *Journal of Computational Physics*, **169**(2), 708–759.

[236] Tsui, Y.-Y. and Pan, Y.-F. (2006). A Pressure-Correction Method for Incompressible Flows Using Unstructured Meshes. *Numerical Heat Transfer, Part B: Fundamentals*, **49**(1), 43–65.

[237] Tuković, Z. and Jasak, H. (2012). A moving mesh finite volume interface tracking method for surface tension dominated interfacial fluid flow. *Computers & Fluids*, **55**, 70–84.

[238] Ubbink, O. (1997). *Numerical prediction of two fluid systems with sharp interfaces*. Ph.D. thesis, Imperial College London.

[239] Ubbink, O. and Issa, R. (1999). A Method for Capturing Sharp Fluid Interfaces on Arbitrary Meshes. *Journal of Computational Physics*, **153**, 26–50.

[240] Unverdi, S. and Tryggvason, G. (1992). A Front-Tracking Method for Viscous, Incompressible, Multi-fluid Flows. *Journal of Computational Physics*, **100**, 25–37.

[241] van der Heul, D. (2013). Personal Communication. Delft University of Technology, Delft, The Netherlands.

[242] van der Pijl, S., Segal, A., Vuik, C., and Wesseling, P. (2008). Computing three-dimensional two-phase flows with a mass-conserving level set method. *Computing and Visualization in Science*, **11**(4-6), 221–235.

[243] van der Pijl, S. P., Segal, A., Vuik, C., and Wesseling, P. (2005). A mass-conserving level-set method for modelling of multi-phase flows. *International Journal for Numerical Methods in Fluids*, **47**, 339–361.

[244] van der Vorst, H. (1992). Bi-CGSTAB: A fast and smoothly converging variant of BI-CG for the solution of nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, **13**(2), 631–644.

[245] van Sint Annaland, M., Deen, N., and Kuipers, J. (2005). Numerical simulation of gas bubbles behaviour using a three-dimensional volume of fluid method. *Chemical Engineering Science*, **60**(11), 2999–3011.

[246] van Wachem, B. and Gopala, V. (2006). A coupled solver approach for multiphase flow calculations on collocated grids. In *European Conference on Computational Fluid Dynamics, ECCOMAS CFD, TU Delft*, pages 1–16.

[247] van Wachem, B. and Schouten, J. (2002). Experimental validation of 3-D lagrangian VOF model: Bubble shape and rise velocity. *AIChE journal*, **48**(12), 2744–2753.

[248] van Wachem, B., Benavides, A., and Gopala, V. (2007). A coupled solver approach for multiphase flow problems. In *6th International Conference on Multiphase Flows 2007*, number 183, Leipzig, Germany.

[249] Versteeg, H. K. and Malalasekera, W. (2007). *An Introduction to Computational Fluid Dynamics: The Finite Volume Method*. Pearson Education, 2 edition.

[250] Vincent, S. and Caltagirone, J.-P. (2005). Test-case No 10: Parasitic currents induced by surface tension (PC). In H. Lemonnier, D. Jamet, and O. Lebaigue, editors, *Validation of Advanced Computational Methods for Multiphase Flow*, number 10, page 260. Begell House Inc.

[251] Von Berg, E., Edelbauer, W., Alajbegovic, A., Tatschl, R., Volmajer, M., Kegl, B., and Ganippa, L. (2005). Coupled simulations of nozzle flow, primary fuel jet breakup, and spray formation. *Journal of engineering for gas turbines and power*, **127**(October), 897.

[252] Wang, D., Li, R., and Yan, N. (2010). An Edge-Based Anisotropic Mesh Refinement Algorithm and its Application to Interface Problems. *Communications in Computational Physics*, **8**(3), 511–540.

[253] Wang, G. (2002). Finite Element Simulations of Free Surface Flows With Surface Tension in Complex Geometries. *Journal of Fluids Engineering*, **124**(3), 584.

[254] Wang, H., Zhou, H., Zhang, Y., Li, D., and Xu, K. (2011). Three-dimensional simulation of underfill process in flip-chip encapsulation. *Computers & Fluids*, **44**(1), 187–201.

[255] Wang, J. P., Borhtwick, A. G. L., and Taylor, R. E. (2004). Finite-volume-type VOF method on dynamically adaptive quadtree grids. *International Journal for Numerical Methods in Fluids*, **45**, 485–508.

[256] Wang, Z. and Tong, A. Y. (2010). A sharp surface tension modeling method for two-phase incompressible interfacial flows. *International Journal for Numerical Methods in Fluids*, **64**, 709–732.

[257] Wang, Z., Yang, J., and Stern, F. (2012). A new volume-of-fluid method with a constructed distance function on general structured grids. *Journal of Computational Physics*, **231**(9), 3703–3722.

[258] Wei, T., Hon, Y., and Wang, Y. (2005). Reconstruction of numerical derivatives from scattered noisy data. *Inverse Problems*, **21**(2), 657–672.

[259] Wenneker, I., Segal, A., and Wesseling, P. (2003). Conservation properties of a new unstructured staggered scheme. *Computers and Fluids*, **32**(1), 139–147.

[260] Wesseling, P. (2001). *Principles of Computational Fluid Dynamics*. Springer.

[261] Wiedemair, W., Tuković, Z., Jasak, H., Poulikakos, D., and Kurtcuoglu, V. (2012). On ultrasound-induced microbubble oscillation in a capillary blood vessel and its implications for the blood-brain barrier. *Physics in Medicine and Biology*, **57**(4), 1019–45.

[262] Williams, M. (2000). *Numerical Methods for Tracking Interfaces with Surface Tension in 3-D Mold-Filling Processes*. Phd thesis, University of California Davis and Los Alamos National Laboratory (Technical Report LA-13776-T).

[263] Williams, M., Kothe, D., and Puckett, E. (1999). Accuracy and convergence of continuum surface-tension models. In W. Shyy and R. Narayanan, editors, *Fluid Dynamics at Interfaces*, pages 294–305. Cambridge University Press, Cambridge.

[264] Wörner, M. (2012). Numerical modeling of multiphase flows in microfluidics and micro process engineering: a review of methods and applications. *Microfluidics and Nanofluidics*, **12**(6), 841–886.

[265] Xu, H. and Liu, J. (2009). Stable numerical differentiation for the second order derivatives. *Advances in Computational Mathematics*, **33**(4), 431–447.

[266] Yang, X. and James, A. J. (2006). Analytic relations for reconstructing piecewise linear interfaces in triangular and tetrahedral grids. *Journal of Computational Physics*, **214**(1), 41–54.

[267] Yang, X., Zhang, X., Li, Z., and He, G.-W. (2009). A smoothing technique for discrete delta functions with application to immersed boundary method in moving boundary simulations. *Journal of Computational Physics*, **228**(20), 7821–7836.

[268] Yang, Y.-J., Yong, J.-H., and Sun, J.-G. (2005). An algorithm for tetrahedral mesh generation based on conforming constrained Delaunay tetrahedralization. *Computers & Graphics*, **29**(4), 606–615.

[269] Youngs, D. L. (1982). Time-dependent multi-material flow with large fluid distortion. In K. Morton and M. Baines, editors, *Numerical Methods for Fluid Dynamics*, page 273. Academic Press, New York.

[270] Yu, B., Kawaguchi, T., Tao, W.-Q., and Ozoe, H. (2002). Checkerboard pressure predictions due to the underrelaxation factor and time step size for a nonstaggered grid with momentum interpolation method. *Numerical Heat Transfer Part B*, **41**, 85–94.

[271] Zahedi, S. and Tornberg, A.-K. (2010). Delta function approximations in level set methods by distance function extension. *Journal of Computational Physics*, **229**(6), 2199–2219.

[272] Zahedi, S., Kronbichler, M., and Kreiss, G. (2012). Spurious currents in finite element based level set methods for two-phase flow. *International Journal for Numerical Methods in Fluids*, **69**, 1433–1456.

[273] Zhang, L.-B. (2009). A Parallel Algorithm for Adaptive Local Refinement of Tetrahedral Meshes Using Bisection. *Numerical Mathematics-Theory Methods and Applications*, **2**, 65–89.

[274] Zhao, X.-Z., Hu, C.-H., and Sun, Z.-C. (2010). Numerical Simulation of Extreme Wave Generation Using VOF Method. *Journal of Hydrodynamics*, **22**(4), 466–477.

[275] Zheng, X., Lowengrub, J., Anderson, A., and Cristini, V. (2005). Adaptive unstructured volume remeshing II: Application to two- and three-dimensional level-set simulations of multiphase flow. *Journal of Computational Physics*, **208**(2), 626–650.

[276] Zuzio, D. and Estivalezes, J. (2011). An efficient block parallel AMR method for two phase interfacial flow simulations. *Computers & Fluids*, **44**(1), 339–357.

[277] Zwart, P. (1999). *The Integrated Space-Time Finite Volume Method*. Ph.D. thesis, University of Waterloo.

A. Implementation of the Numerical Framework

A key aspect of numerical simulations using unstructured meshes is the implementation and parallelisation of the data structure. Mesh connectivity information is particularly important with respect to unstructured meshes, since an unstructured mesh does not possess an inherent order. The data structure and the connectivity information directly affect the numerical discretisation and the performance of the software. In what follows, the basic implementation and parallelisation of the numerical framework presented this thesis is explained. The explanation focuses on specific details, such as connectivity information, indexing of mesh entities, numerical discretisation and ghost information.

A.1. Data Structure

The data structure consists of three types of mesh entities: *node*[9], *face* and *element*[10]. Additionally the mesh entity type *edge* is required when the tetrahedral mesh adaption algorithm presented in Section 5.4 is used. Mesh nodes are defined by their coordinates $\{x, y, z\}$ whereas the other mesh entities are defined by their constituting nodes. Each mesh entity is identified by a unique index, ranging from 0 to $N_G - 1$, where N_G is the number of this mesh entity (e.g. elements).

Connectivity information of the mesh is required to determine the position of mesh entities within the mesh, or in the context of numerical discretisation, the neighbour relations of mesh entities. The mesh connectivity information includes:

- nodes that constitute each edge,
- elements adjacent to each edge,
- nodes that constitute each face,
- elements adjacent to each face,
- nodes that constitute each element,
- bounding faces of each element, and
- neighbour elements of each element.

[9] Also called *vertex*.
[10] Also called *cell*.

This set of connectivity information is not memory-optimised but is the result of focusing on computational performance rather than memory efficiency. For instance, the neighbour elements of each element could also be determined from the elements adjacent to each face whenever necessary. This would, however, require notable additional computational effort.

Each set of connectivity information is represented by two integer-arrays. Using the faces bounding an element as an example, one array, called for instance *elementFaces*, is a continuous list of the indices of the bounding faces for each element. Thus, the length of *elementFaces* is the sum of bounding faces for each individual element of all elements in the domain. The second array, called *elementFacesPtr*, contains the position of the first bounding face in array *elementFaces* for each element. In order to perform loops over this array, it is of length *number of elements + 1.* A schematical example of the two arrays and their interrelation is given below.

```c
// Initialisation of the counting variables
int i, j;

// Loop over all mesh elements
// Number of elements is abbreviated "noElem"
for (i=0; i<noElem; i++)
{
   // Loop over the bounding faces of each element
   for (j=elementFacesPtr[i]; j<elementFacesPtr[i+1]; j++)
   {
      elementFaces[j] = 0;
   }
}
```

Listing A.1: Sample C-code to demonstrate accessing connectivity information stored in a sparse-array. The example shows the initialisation of the array holding the bounding faces of all elements.

The numerical discretisation of partial differential equations on unstructured meshes, as for instance described in Section 2.2 and Chapter 3, requires additional mesh properties. Considering a mesh without skewness, the centre and area of each face as well as the centre and volume of each mesh element are required for discretisation. Since mesh skewness is common on unstructured meshes, the interpolation point f', as illustrated in Figure 2.2b on page 48, of every mesh face is stored separately as well. The interpolation point could, of course, also be calculated on demand. However, for the benefit of computational efficiency the interpolation points are computed only once and stored for later use.

The unit normal vector of each face $\boldsymbol{n}_f$ is required for the discretisation of the governing equations. Using the finite volume method and applying the midpoint rule as given in Eq. 2.16, the face normal vector pointing in outward direction of the element under consideration is required. This means, the face normal vector has to point in a specific direction. As a directional orientation, for instance along the relevant coordinate axis, is not applicable on unstructured meshes, the normal vectors are set by default to be directed towards the adjacent mesh element of higher index, as shown in the example depicted in Figure A.1. An integer value of either 1 or -1 is stored for each bounding face at each element, dependent on the relative orientation of the respective face normal vector. This integer value is used as a multiplier for the face normal vector in the numerical discretisation, assuring the correct orientation of the face normal vector.

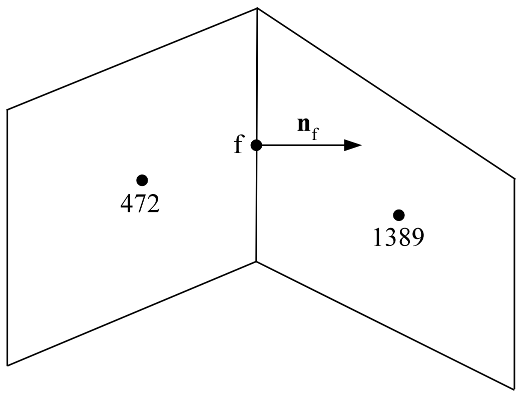


Figure A.1.: Example of the face normal vector orientation. The normal vector is pointing towards the adjacent element with the higher index.

A.2. Parallelisation

To maximise the performance and make efficient use of high-performance computer systems, a well-suited parallelisation of the numerical framework is important, in particular with respect to the numerical discretisation and the solving procedure. The mesh is divided into N_{proc} equally (or rather as equally as possible) sized partitions, where N_{proc} is the number of processors used for a given simulation.

So-called *ghost elements* are required, in order to keep the discretisation *local* to a given processor and minimise interprocessor communication. Ghost elements are copies of elements which are local to a neighbouring processor and adjacent to the respective processor boundary, schematically illustrated in Figure A.2. The data at such ghost elements is required for the discretisation of one or more elements local to a given processor and adjacent to the respective processor boundary as well. The data at ghost elements is communicated once per time-step or iteration, respectively. Therefore, interprocessor communication is reduced to a minimum. For a classical finite volume discretisation as presented in Section 2.2, only one layer of ghost elements is required. Every additional layer of ghost elements increases the interprocessor communication and adversely affects the computational performance.

As mentioned previously, every mesh entity is identified by a unique index. For the sake of simplicity and computational efficiency, all mesh entities are consecutively numbered with a local index, ranging from 0 to N_L, where N_L is the local number of a given mesh entity including all its ghost entities with respect to the given processor. The local indexing system enables quick and simple local data manipulation.

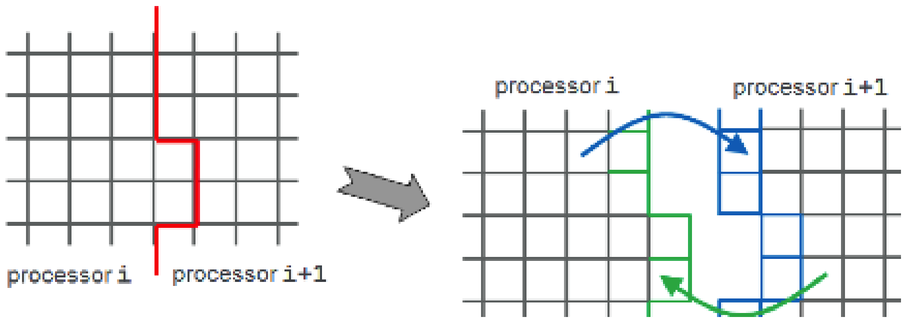
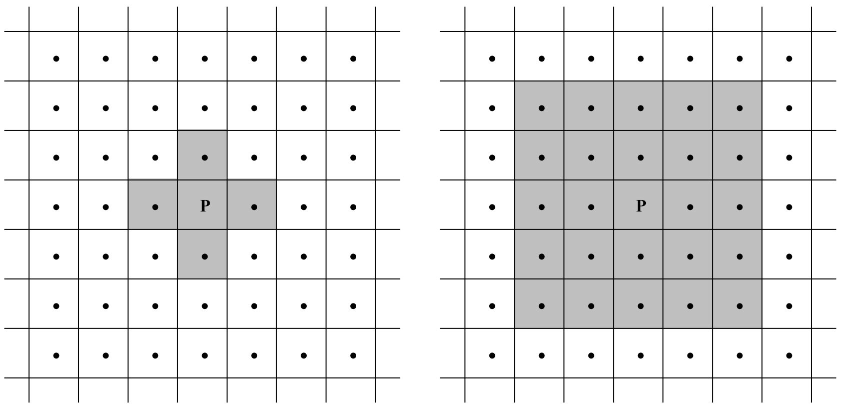


Figure A.2.: Schematical example of the ghost pattern at processor boundaries, required for numerical discretisation. The processor boundary between the two processors is highlighted in *red* and the ghost elements are illustrated in *green* and *blue*, respectively.

A.3. Modifications for CELESTE

The CELESTE method to evaluate the interface curvature on both structured and unstructured meshes requires only little modification to the existing data structure. Additional neighbour cells are required for the discretisation using CELESTE compared to the standard finite volume discretisation of the momentum equations, the continuity constraint and the interface advection, because of the symmetric and typically larger stencil size for CELESTE, as shown in Figure A.3. Therefore, the additional neighbour cells are appended to the neighbour-elements connectivity information, to allow quick and easy access of the required cells.



(a) Finite volume stencil (b) CELESTE stencil

Figure A.3.: Comparison of the typical finite volume stencil and an example of a CELESTE stencil.

Performing multi-processor simulations, the ghost pattern has to be extended to cover the additional neighbour connectivity information required for the discretisation using the CELESTE method. However, in order to reduce the additional interprocessor communication effort to a minimum, the extended ghost pattern is only applied to the coordinates of the cell-centres and the colour function. Other data is not required to evaluate the interface curvature using the CELESTE method.

B. Damping Spurious Pressure Oscillations on Collocated Meshes

The damping of spurious pressure oscillation for numerical methods with collocated variable arrangement using the momentum interpolation method, first proposed by Rhie and Chow [198], is demonstrated by means of a one-dimensional example.

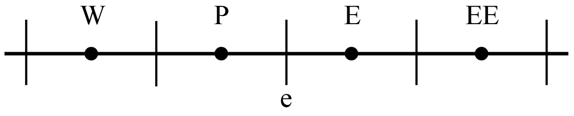


Figure B.1.: Example of an equidistant one-dimensional mesh.

Interpolating the velocity by means of Eq. 3.38 to face e of the one-dimensional mesh depicted in Figure B.1 is given as

$$u_e = \frac{u_P + u_E}{2} - \hat{d}_e \left[\left.\frac{\partial p}{\partial x}\right|_e - \left.\overline{\frac{\partial p}{\partial x}}\right|_e \right] + c\,\hat{d}_e \left[u_e^{t-\Delta t} - \frac{u_P^{t-\Delta t} + u_E^{t-\Delta t}}{2} \right] . \tag{B.1}$$

Discretising the pressure term of Eq. B.1 using standard finite difference methods, with Δx representing the mesh spacing, results in

$$\begin{aligned}
\hat{d}_e \left[\left.\frac{\partial p}{\partial x}\right|_e - \left.\overline{\frac{\partial p}{\partial x}}\right|_e \right] &= \hat{d}_e \left[\left.\frac{\partial p}{\partial x}\right|_e - \frac{1}{2} \left(\left.\frac{\partial p}{\partial x}\right|_P + \left.\frac{\partial p}{\partial x}\right|_E \right) \right] && \text{(B.2)}\\
&= \hat{d}_e \left[\frac{p_E - p_P}{\Delta x} - \frac{1}{2} \left(\frac{p_E - p_W}{2\Delta x} + \frac{p_{EE} - p_P}{2\Delta x} \right) \right] && \text{(B.3)}\\
&= \frac{\hat{d}_e}{4\Delta x} \left(p_W - 3p_P + 3p_E - p_{EE} \right) && \text{(B.4)}
\end{aligned}$$

Following the derivation of Versteeg and Malalasekera [249, page 340], the third derivative of pressure at face e is given as

$$\begin{aligned}
\left.\frac{\partial^3 p}{\partial x^3}\right|_e = \left.\frac{\partial}{\partial x}\frac{\partial^2 p}{\partial x^2}\right|_e &= \frac{1}{\Delta x} \left(\left.\frac{\partial^2 p}{\partial x^2}\right|_E - \left.\frac{\partial^2 p}{\partial x^2}\right|_P \right) && \text{(B.5)}\\
&= \frac{1}{\Delta x} \left(\frac{p_{EE} - 2p_E + p_P}{\Delta x^2} - \frac{p_E - 2p_P + p_W}{\Delta x^2} \right) && \text{(B.6)}\\
&= -\frac{1}{\Delta x^3} \left(p_W - 3p_P + 3p_E - p_{EE} \right) . && \text{(B.7)}
\end{aligned}$$

Thus, comparing Eqs. B.4 and B.7 it becomes evident that

$$\hat{d}_e \left[\left.\frac{\partial p}{\partial x}\right|_e - \frac{1}{2} \left(\left.\frac{\partial p}{\partial x}\right|_P + \left.\frac{\partial p}{\partial x}\right|_E \right) \right] = -\frac{\hat{d}_e}{4} \left.\frac{\partial^3 p}{\partial x^3}\right|_e \Delta x^2 \tag{B.8}$$

and, therefore,

$$\left.\frac{\partial p}{\partial x}\right|_e - \frac{1}{2} \left(\left.\frac{\partial p}{\partial x}\right|_P + \left.\frac{\partial p}{\partial x}\right|_E \right) \propto \left.\frac{\partial^3 p}{\partial x^3}\right|_e . \tag{B.9}$$

It is this relationship that dampens out the spurious pressure oscillations on collocated meshes. Because the momentum interpolation method is second-order accurate at best, the third-order pressure gradient term does not affect the formal accuracy of the method [249].

C. Permissions to Republish Third Party Material

See following pages.

ELSEVIER LICENSE
TERMS AND CONDITIONS

Nov 19, 2013

This is a License Agreement between Fabian Denner ("You") and Elsevier ("Elsevier") provided by Copyright Clearance Center ("CCC"). The license consists of your order details, the terms and conditions provided by Elsevier, and the payment terms and conditions.

All payments must be made in full to CCC. For payment instructions, please see information listed at the bottom of this form.

Supplier	Elsevier Limited The Boulevard,Langford Lane Kidlington,Oxford,OX5 1GB,UK
Registered Company Number	1982084
Customer name	Fabian Denner
Customer address	Department of Mechanical Engineering London, SW7 2AZ
License number	3272560817639
License date	Nov 19, 2013
Licensed content publisher	Elsevier
Licensed content publication	Chemical Engineering Science
Licensed content title	Numerical simulation of gas bubbles behaviour using a three-dimensional volume of fluid method
Licensed content author	M. van Sint Annaland,N.G. Deen,J.A.M. Kuipers
Licensed content date	June 2005
Licensed content volume number	60
Licensed content issue number	11
Number of pages	13
Start Page	2999
End Page	3011
Type of Use	reuse in a thesis/dissertation
Portion	figures/tables/illustrations
Number of figures/tables/illustrations	2
Format	both print and electronic
Are you the author of this Elsevier article?	No
Will you be translating?	No
Title of your thesis/dissertation	Balanced-Force Two-Phase Flow Modelling on Unstructured and Adaptive Meshes
Expected completion date	Nov 2013

Estimated size (number of pages)	229
Elsevier VAT number	GB 494 6272 12
Permissions price	0.00 GBP
VAT/Local Sales Tax	0.00 GBP / 0.00 GBP
Total	0.00 GBP
Terms and Conditions	

permission.

21. **Other Conditions**:

v1.6

If you would like to pay for this license now, please remit this license along with your payment made payable to "COPYRIGHT CLEARANCE CENTER" otherwise you will be invoiced within 48 hours of the license date. Payment should be in the form of a check or money order referencing your account number and this invoice number RLNK501162901.
Once you receive your invoice for this order, you may pay your invoice by credit card. Please follow instructions provided at that time.

Make Payment To:
Copyright Clearance Center
Dept 001
P.O. Box 843006
Boston, MA 02284-3006

For suggestions or comments regarding this order, contact RightsLink Customer Support: customercare@copyright.com or +1-877-622-5543 (toll free in the US) or +1-978-646-2777.

Gratis licenses (referencing $0 in the Total field) are free. Please retain this printable license for your reference. No payment is required.