Tesis Doctoral

2021

Numerical study of drop impacts on solid surfaces. Influence of the contact line dynamics

Adolfo Esteban Paz

Programa de Doctorado en Tecnologías Industriales

Director: Julio Hernández Rodríguez
Codirector: Pablo Joaquín Gómez del Pino

Abstract

This work focuses on the numerical investigation of the drop impact on a solid surface, with especial emphasis on the influence of the contact angle on the dynamics of the contact line and the impact outcome. To this end, different numerical models and computational tools are developed and implemented to accurately and efficiently simulate the complex phenomena associated to this type of two-phase flow.

Interface tracking is one of the main difficulties encountered in the simulation of immiscible two-phase flows, and therefore special attention is given to the task of selecting, assessing and implementing a volume of fluid (VOF) method capable of accurately simulating this type of three-dimensional and highly unsteady flows. On the one hand, an algebraic VOF method has been used to simulate axisymmetric drop impacts leading to deposition. On the other hand, a geometric advection method already available in OpenFOAM has been used in combination with a reconstruction method that has been implemented in OpenFOAM for this purpose. The accuracy and efficiency of the methods used have been assessed through a systematic and comprehensive comparison with other VOF methods, which has revealed their advantages. This has been carried out by means of several kinematic tests and numerical simulations in which the interface tracking methods have been solved along with the Navier-Stokes equations.

The influence of the scheme chosen for the discretization of the convective term of the momentum equation is also studied, an issue that is crucial in the instants immediately after the drop impacts the solid surface and that determines the ability of the numerical model to reproduce the complexity of the flow that can occur in impacts with fingering and splashing phenomena.

Another major issue in the numerical simulation of unsteady interfacial flows in the vicinity of a solid surface is the simulation of the dynamics of the contact line. This is due to the singularity that arises when using the continuum hypothesis in the description of the contact line motion combined with a no-slip boundary condition on the solid surface. To simply but effectively reproduce the contact line dynamics on solid surfaces, a contact line force model is proposed. The contact line force is introduced as a force per unit volume in the Navier-Stokes equations, in combination with the simultaneous imposition of the interface angle at the contact line. The suitability of the proposed model is studied by simulating several impacts over a wide range of Reynolds and Weber numbers. The results are compared with experimental data obtained by other authors and with numerical results obtained with different contact angle models.

The proposed contact line model has been used, along with the implemented geometric VOF method, in the numerical simulation of the impact of a drop on a hydrophobic surface, under fingering and splashing conditions. The results are compared with our own experimental results, obtaining a good degree of agreement despite the complex phenomena involved.

Resumen

El presente trabajo está centrado en el estudio numérico del impacto de gotas sobre superficies sólidas, con especial énfasis en la influencia del ángulo de contacto sobre la dinámica de la línea de contacto y el patrón de flujo obtenido. Con este propósito, se han desarrollado e implementado diferentes modelos numéricos y herramientas computacionales que permiten simular de manera precisa y eficiente los complejos fenómenos asociados a este tipo de flujo interfacial.

El seguimiento de la interfaz es una de las principales dificultades que surgen al simular numéricamente flujos de dos fases no miscibles, y, por tanto, se ha prestado especial atención a la tarea de seleccionar, mejorar e implementar un método de tipo "volume of fluid" (VOF) capaz de simular este tipo de flujos tridimensionales y altamente no estacionarios. Por un lado, se ha usado un método tipo VOF algebraico en la simulación de impactos axisimétricos de gotas que dan lugar a una deposición. Por otro lado, se ha empleado un método geométrico de advección de la interfaz disponible en OpenFOAM en combinación con un método de reconstrucción de la interfaz que ha sido implementado en OpenFOAM con este propósito. La precisión y eficiencia de los métodos empleados se han validado mediante una comparación sistemática y exhaustiva con otros métodos VOF, lo que ha puesto de manifiesto sus ventajas. Esto ha sido llevado a cabo empleando varios tests cinemáticos y simulaciones numéricas en las que las ecuaciones de Navier-Stokes son resueltas.

También se ha estudiado la influencia del esquema elegido en la discretización del término convectivo en la ecuación de conservación de cantidad de movimiento, lo que es crucial en instantes inmediatamente posteriores al impacto de la gota sobre la superficie sólida y que determina la capacidad del modelo numérico para reproducir la complejidad del flujo que puede tener lugar en impactos con fenómenos de salpicadura y "fingering".

Otro de los aspectos esenciales en la simulación numérica de flujos interfaciales no estacionarios cerca de una superficie sólida es la adecuada reproducción de la dinámica de la línea de contacto. Esto es debido a la singularidad que surge al emplear la hipótesis de medio continuo en la descripción del movimiento de dicha línea junto con la aplicación de la condición de no deslizamiento en la superficie sólida. Para reproducir de manera simple pero efectiva la dinámica de la línea de contacto en superficies sólidas, se ha propuesto un modelo de fuerza de la línea de contacto. La fuerza es introducida como una fuerza de volumen en las ecuaciones de Navier-Stokes al tiempo que se impone el ángulo de la interfaz en la línea de contacto. Se ha estudiado la adecuación del modelo mediante la simulación de varios impactos en un amplio rango de números de Reynolds y Weber. Los resultados son comparados con datos experimentales obtenidos por otros autores y con resultados numéricos obtenidos con diferentes modelos de ángulo de contacto.

El modelo de línea de contacto propuesto se ha empleado, junto con el método VOF geométrico implementado, en la simulación numérica del impacto de una gota sobre una superficie hidrofóbica, bajo condiciones en las que se forman salpicaduras y "fingers". Los resultados son comparados con resultados experimentales propios, obteniendo un buen grado de concordancia a pesar de los complejos fenómenos que intervienen.

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Professor Julio Hernández, for his dedication, patience, guidance and his, sometimes excessive but indispensable and contagious, rigor. I would also like to thank the countless advice I have received from my co-supervisor, Professor Pablo Gómez, which have made this work possible. The help of Professor Joaquín López with the code and numerical modelling made me learn a still-growing countless list of things. The suggestions received from Professor Claudio Zanzi have also been very important for the elaboration of this work, as well as his encouragement.

I would also like to thank Professor Markus Bussmann at the University of Toronto for the knowledge about drop impact modeling he has shared with me. Also, the suggestions and recommendations from Professor Johan Rønby at the Aalborg University have been very helpful.

The constant support of my parents, my sister and my uncles has been vital not only for the conclusion of this work, but even for thinking about starting it.

Thank you, Lidia, for being at my side even when I couldn't be at yours.

Agradecimientos

Me gustaría empezar agradeciendo a mi director de tesis, el catedrático Julio Hernández, su completa dedicación, paciencia y orientación como su, a veces excesivo pero indispensable y contagioso, rigor. Agradezco también los innumerables consejos que he recibido de mi codirector de tesis, el profesor Pablo Gómez, sin los que no hubiera podido llevar a cabo este trabajo. La imprescindible ayuda del catedrático Joaquín López con el código y el modelado numérico me han hecho aprender una innumerable, y todavía creciente, lista de cosas. No menos importante ha sido la ayuda que me ha prestado el profesor Claudio Zanzi, así como sus ánimos y sugerencias.

Por otro lado, también me gustaría agradecer el conocimiento sobre el modelado numérico del impacto de gotas que el profesor Markus Bussmann en la University of Toronto ha compartido conmigo. Así mismo, las recomendaciones y consejos del profesor Johann Rønby en la Aalborg University me han sido de gran ayuda.

El constante apoyo de mis padres, mi hermana, mis tías y tíos ha sido imprescindible no solo para la elaboración de esta tesis, si no para el simple hecho de pensar en realizarla siquiera.

Gracias, Lidia, por estar a mi lado incluso cuando no he podido estar al tuyo.

A mi familia, y en especial a Lidia

Contents

List of Tables

List of Figures

Nomenclature

Roman symbols

$\mathbf{A}$	matrix of coefficients
$\mathbf{b}$	vector of coefficients
C	constant in a plane equation
Ca	capillary number
C_k	kernel normalization constant
$\mathbf{d}$	vector joining two centers
D	diameter
E_{bound}	boundedness error
E_g	geometric error
E_{rec}	reconstruction error
E_{vol}	volume conservation error
E_{v}	initialization error
F	volume fraction
F_{tol}	volume fraction tolerance
f_H	Hoffman function
f_{imp}	implicit function
$\mathbf{f}_v$	body force per unit volume
$\mathbf{g}$	gravity vector
K	kernel
L	length scale of the intermediate region
l_m	slip length

$\dot{m}$	mass flux
n	size of the mesh
N	number of fingers
n_c	number of cells
n_f	number of faces
$\mathbf{n}$	unit vector normal to the interface
$\mathbf{n}_{\text{iso}}$	unit vector normal to the 0.5-isosurface
n_p	number of points
n_{proc}	number of processors
$\mathbf{n}_w$	unit vector normal to the wall
Oh	Ohnesorge number
p	pressure
p_d	modified pressure
Re	Reynolds number
r_k	kernel radius
$\mathbf{S}_f$	surface area vector of face f
t	time
t_{adv}	cpu-time consumed in advection
t_{rec}	cpu-time consumed in reconstruction
t_{tot}	total consumed cpu-time
$\mathbf{u}$	velocity vector
U	impact velocity magnitude
u_{cl}	contact line velocity
$\mathbf{u}_{\text{int}}$	interface velocity
V_c	cell volume
V_f	volume of liquid advected through face f
V_{init}	initialized volume
w	weighting factor

We	Weber number
$\mathbf{x}$	position vector

Greek symbols

χ	indicator function
δ_s	Dirac delta function
Δt	time interval
ε_r	radial error
ε_z	height error
κ	interface curvature
λ	contact line longitude
μ	dynamic viscosity
Ω	computational cell
ϕ_f	face volumetric flux
φ	scalar field
$\widetilde{\varphi}$	isovalue of the scalar field
$\boldsymbol{\Phi}$	vector of unknowns
ψ	limiter function
ρ	density
σ	surface tension
ϑ	azimuthal coordinate
θ_a	advancing contact angle
θ_d	dynamic contact angle
θ_e	equilibrium contact angle
θ_m	microscopic contact angle
θ_r	receding contact angle
θ_s	static contact angle

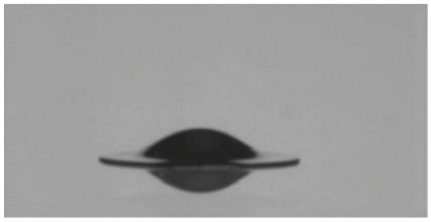
Chapter 1

Introduction

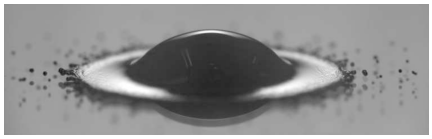
1.1 Problem description and motivation

The impact of a drop on a solid surface is a phenomenon of great interest that it is present in a wide range of applications such as soil erosion, surface painting and spray coating, structured materials and circuit micro-manufacturing, 3D printing based on micro-droplet deposition, among others [183]. The drop impact problem can be pictured as a liquid drop of initial diameter D_0 that impacts with a velocity U on a dry solid surface. After the impact, the liquid spreads over the surface following different flow patterns. Rioboo et al. [135] classified the different outcomes as shown in Fig. 1.1. The deposition (Fig. 1.1(a)) is characterized by the formation of a liquid lamella which spreads over the solid surface without any breakup or liquid ejection. The splash is considered to occur when the drop impact results on the breakup of the drop and at least a secondary droplet is formed. Two different splash patterns are distinguished: (1) prompt splash (Fig. 1.1(b)), in which the secondary droplets are ejected from the contact line immediately after the impact; and (2) corona splash (Fig. 1.1(c)), in which the lamella detaches from the surface and several secondary droplets are ejected from its rim. Another possible outcome is the receding breakup (Fig. 1.1(d)), which occurs when the lamella retracts from its maximum spreading diameter and some drops are left behind by the receding lamella. The total and partial rebounds (Figs. 1.1(e) and (f)), which are only observed if a receding stage takes place, occur when the entire drop rebounds on the surface and when only a part of the drop rebounds while the rest remains attached to the surface, respectively.

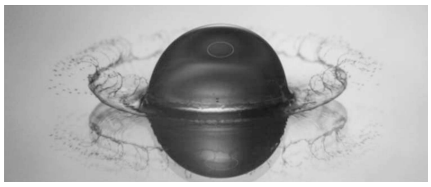
The resulting outcome depends on a great variety of factors. These include fluids properties, surface characteristics and impact parameters [166]. Some authors have studied the influence of the ambient temperature and pressure or the physical properties of the impact surface [21, 181], although, for example, air density is usually considered to be constant and therefore the thermal and compressibility effects are neglected. The liquid properties involved are the density, ρ_l, the dynamic viscosity, μ_l, and the surface tension, σ. The following two dimensionless numbers can be formed:



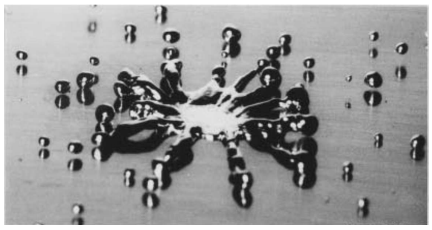
(a) Deposition [118]



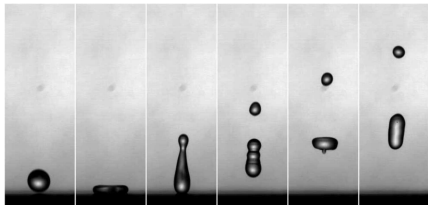
(b) Prompt splash [72]



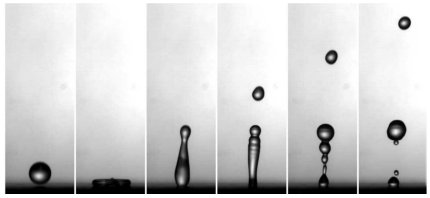
(c) Corona splash [72]



(d) Receding breakup [9]



(e) Total rebound [129]



(f) Partial rebound [129]

Figure 1.1: Possible outcomes when a drop impacts on a solid surface as classified by Rioboo et al. [135].

Reynolds number,

$$\mathrm{Re} = \frac{\rho_l \, U \, D_0}{\mu_l}, \tag{1.1}$$

and Weber number,

$$\mathrm{We} = \frac{\rho_l \, U^2 \, D_0}{\sigma}, \tag{1.2}$$

which represent the ratio of the inertia and viscous and capillary forces, respectively. Combinations of these two numbers are also often used, which include the Ohnesorge number,

$$\mathrm{Oh} = \frac{\sqrt{\mathrm{We}}}{\mathrm{Re}} = \frac{\mu_l}{\sqrt{\rho \sigma D_0}}, \tag{1.3}$$

and the capillary number,

$$\mathrm{Ca} = \frac{\mathrm{We}}{\mathrm{Re}} = \frac{\mu_l \, U}{\sigma}. \tag{1.4}$$

(Note that in this work reference will be made to a capillary number based on the contact line velocity, u_{cl}, rather than on the impact velocity.)

When the drop comes into contact with the solid surface after impact, a moving contact line is formed between the solid and the two immiscible fluids, giving rise to a triple phase zone. There are great differences in the length scales involved in the

phenomena that appear near this zone, ranging from the macroscopic to the molecular scales, which sometimes make it difficult to clearly identify their influence on the flow. The angle at which the interface between the two fluids intersects the solid surface, known as contact angle, characterizes the wettability of the solid surface, determined by the liquid-gas-solid interaction. However, the definition of this parameter is somehow controversial due to its dependence on the length scale and to the fact that it can be influenced by parameters such as the surface roughness [52].

In situations where the drop impact outcome is not a splash, the time evolution of the spread factor, D/D_0, where D is the diameter of the spreading lamella, can be divided into four successive stages [134, 139]: (1) inertial, in which the shape of most of the drop is still nearly spherical; (2) spreading, in which the liquid spreads over the solid surface and a lamella bounded by a rim can be clearly identified; (3) receding, which takes place after the maximum spreading factor has been reached and is characterized by the growth of the width of the lamella rim and the decrease of the spreading diameter; and (4) equilibrium or rebound, in which the receding velocity and the properties of the liquid and impact surface determine the last stage of the outcome.

The drop impact problem has received special attention from many researchers since the early work of Worthington [178] in 1876, not only due to its practical interest, but also to its complexity derived from the factors mentioned above: the dependence on the physical properties of the liquid and impact surface and on the impact conditions; the several length scales involved; the difficulty in clearly identifying the mechanisms that take place during drop spreading and splashing; and the problem of the moving contact line, which poses significant mathematical modeling difficulties. These issues have given rise to different theories in the literature, sometimes contradictory to each other, which try to explain the complex phenomena that occur when a drop impacts on a solid surface [72]. The mathematical modeling of the contact line dynamics poses an especial difficulty since its motion is incompatible with the no-slip boundary condition imposed at the solid surface in the Navier-Stokes equations, which arises from a continuum description of the problem. Different strategies have been developed to handle this singularity and capture the dynamics of the contact line, which will be described below.

The impact of a drop on a solid surface is a case of interfacial flow where two immiscible fluids coexist separated by a possibly high complex interface through which a discontinuity in the physical properties of the fluids is produced. A variable for tracking the interface is generally used, therefore one more equation that describes its evolution must be added to the mathematical model. There are many approaches for the interface evolution tracking in numerical simulation of interfacial flows, and can be classified into three main categories [142]: Lagrangian, where the governing equations are solved on a mesh that moves along with the interface; mixed Eulerian-Lagrangian

(e.g., front-tracking method [167]), in which the governing equations are solved on a fixed mesh and the interface is tracked by marker points; and Eulerian, in which the governing equations are also solved on a fixed mesh and the interface is represented through a function defined on each computational cell of the mesh, advected along with the fluid velocity. Examples of methods belonging to this last category are the phase-field, level-set and volume of fluid (VOF). The latter method has received special attention by the scientific community due, among others advantages, its good mass conservation properties compared to other methods.

The impact of a drop on a solid surface is a case of interfacial flow in which two immiscible fluids coexist separated by a possibly highly complex interface across which there is a discontinuity in the physical properties of the fluids. To solve such a flow numerically, it is common to use an indicator variable or marker particles to track the interface, therefore an equation describing its evolution is added to the mathematical model. There are many approaches for interface tracking in the numerical simulation of interfacial flows, which can be classified into three main categories [142]: Lagrangian, where the governing equations are solved on a mesh that moves along with the interface; mixed Eulerian-Lagrangian (e.g., front-tracking method [167]), in which the governing equations are solved on a fixed mesh and the interface is tracked by marker points; and Eulerian, in which the governing equations are also solved on a fixed mesh and the interface is represented through a function defined in each computational cell of the mesh, advected with the fluid velocity. Examples of methods belonging to the last category are phase-field, level-set and volume of fluid (VOF). The latter has received special attention by the scientific community due, among others advantages, to its good mass conservation properties compared to other methods.

In the VOF method, the indicator function representing the interface is approximated by the fraction of a computational mesh cell occupied by the liquid (volume fraction), which can be obtained by geometric operations (geometric VOF methods) or by using numerical approximations (algebraic VOF methods). The accuracy of the latter type of methods is usually substantially lower than that of the former, mainly due to numerical diffusion, although they are less computationally expensive [109]. In geometric VOF methods, an additional step is required, prior to the computation of the liquid volume advected across the cell boundaries, in which the interface is reconstructed from the volume fraction. The interface in a computational cell is generally represented by a plane. The interface reconstruction method is of the piecewise linear interface calculation (PLIC) type if no restrictions on the plane orientation are considered and of the simple line interface calculation (SLIC) type if the plane orientation is restricted to be aligned with a coordinate axis. The plane is positioned in the cell so that, for a given orientation, the volume enclosed by the intersection between the plane and the cell corresponds to the volume of liquid in that cell. The orientation of the plane in a cell is obtained from the volume fraction distribution, using meth-

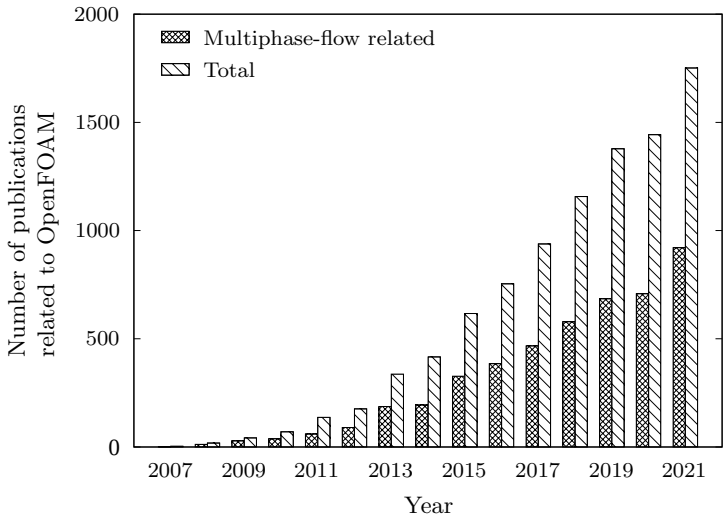


Figure 1.2: Estimation of the number of publications in which the OpenFOAM code was used, showing the fraction of which were related to multiphase flow simulation. Data obtained from [146, 157].

ods such as those based on its gradient, a reconstructed distance function (RDF) or an isosurface (a generally non-planar polygon in which all its vertices have the same volume fraction value), among others. An accurate computation of the interface position and orientation is of utmost importance because, on the one hand, it determines where the change in fluid properties occurs, where the surface tension is applied, and the intensity of the surface tension, which depends on the local shape of the interface, and, on the other hand, it is used to estimate the fluxes needed to solve the advection equation that describes the evolution of the indicator function. Interface advection in geometric VOF methods is generally based on the construction of donating flux regions used to determine the liquid volume flux across cell boundaries, which involve complex truncation operations.

The VOF method and other interface tracking methods can be implemented in a great variety of computational frameworks to be used along with algorithms for solving the Navier-Stokes equations. In recent years, the open-source CFD software OpenFOAM [115] has been increasingly and widely used by the research community as a general-purpose code to simulate fluid-mechanics problems, including those related to multiphase-flow (see Fig. 1.2). Interest has been focused both on using the code to carry out simulations and on contributing to increase its simulation capabilities by developing, implementing and releasing new numerical methods and improving those already included in the code. In this context, the present work uses OpenFOAM as the computational framework in which the proposed contact line force model is implemented and assessed [47], and the task of selecting, assessing and implementing a VOF method capable of accurately simulating three-dimensional and highly unsteady two-phase flows is carried out [46].

1.2 Mechanisms involved in the drop impact

As discussed in the previous section, one of the major issues when studying the impact of a drop on a solid surface arises from the influence of the large number of factors involved. Numerous works have studied the influence of different parameters on the flow pattern obtained after drop impact, such as liquid properties (especially viscosity), surface tension, impact velocity, size and shape of the drop, impact surface characteristics, air properties and ambient conditions, among others. However, there is no general agreement on which physical mechanisms are behind the cause of drop splashing, although several authors have investigated the splashing/deposition threshold [103, 118, 131, 166]. On the other hand, there is much more consensus on the processes that take place in impacts that result in deposition outcomes [72].

The effects of the liquid viscosity and surface tension have been widely studied. When surface tension is decreased, the splash outcome is promoted [151]. When liquid viscosity is increased, the momentum dissipation at the surface is higher and, therefore, the maximum spread factor is smaller. The retraction stage is also slower, and starts earlier for less viscous liquids [151, 176]. However, the viscosity effect involving the splash outcome is more complex. For $\mathrm{Re} > 1000$, an increase in the viscosity leads to the formation of splash as, in this regime, the expanding liquid film is stabilized mainly by surface tension while viscosity only affects the film thickness. A greater viscosity causes a thicker film, which is easier to be destabilized, promoting the splash. For $\mathrm{Re} < 1000$, viscous drag is more relevant and helps to stabilize the lamella, so the increase of the liquid viscosity tends to suppress splash.

The effects of the liquid viscosity and surface tension have been widely studied. When surface tension is decreased, the splashing outcome is promoted [151]. When viscosity is increased, the momentum dissipation at the surface is higher and, therefore, the maximum spread factor is smaller. The viscosity effect in splashing drop impacts has been under discussion [118]. For $\mathrm{Re} \gtrsim 1000$, an increase in viscosity promotes splashing since, in this regime, the expanding liquid film is stabilized mainly by surface tension, while viscosity affects the film thickness. A higher viscosity causes a thicker film, which is easier to be destabilized, promoting splashing. For $\mathrm{Re} \lesssim 1000$, viscous drag is more relevant and helps to stabilize the lamella, so an increase in liquid viscosity tends to suppress splashing.

For sufficiently low values of the impact velocity, a deposition takes place. If the impact velocity increases, the spreading lamella diameter also increases [151]. For $\mathrm{Re} \gtrsim 1000$, if the impact velocity is increased, instabilities appear that promote the apparition of the splash. For $\mathrm{Re} \lesssim 1000$, the splash outcome is preceded by the lift of a thin liquid film moving parallel to the solid surface without the formation of any secondary droplets. In this regime, if the impact velocity is sufficiently high, secondary droplets detach from the rim of the lifted thin liquid film [118].

The drop size also affects the resulting outcome. For a given We, if the drop size is diminished, the maximum spread factor reached during the impact is smaller since the viscous effects on a smaller drop are more important. If the Re number is kept constant, the maximum spreading diameter becomes larger for smaller drops because the capillary effects are less important.

The effect of the air trapped beneath the impacting drop was investigated by Kolinski et al. [76]. This trapped air forms a very thin film over which the liquid spreads up to a certain distance where the liquid-solid contact is produced. Their results indicated that the thickness of the air film and the length over which the drop skates decrease with impact velocity, up to reach a minimum value in both parameters. Mandre and Brenner [103] indicated that there is a critical impact velocity over which a thin liquid sheet is ejected before the liquid-solid contact, and when this sheet contacts the surface the splash is produced. Jian et al. [69] studied the effects of the density and viscosity of the gas in the formation of splash. The volume of the trapped bubble increases with gas density. They distinguished two types of splashing mechanisms: jet and detachment splashing, corresponding to a jet ejection from the lamella before and after, respectively, the liquid-solid contact. The first case corresponds to higher values of gas viscosity. If the gas viscosity is reduced sufficiently the splash does not appear. Liu et al. [86] carried out experiments in the low viscous regime, in which the lamella skates over a thin air film and then the liquid is ejected from the rim of the lamella before it touches the surface of impact. The authors found that the entrapped air under the lamella is responsible for triggering the splash since, when the drop impacts on a porous surface where the air can escape through the pores, splashing is completely suppressed.

The previous comments highlight the relevance of the entrapped air in the splash formation. A mechanism that takes this into account is based on the Kelvin-Helmholtz instability, which is supposed to destabilize the lamella [74, 86] and is originated by the high velocity gradient between the liquid and the thin gas film, in which the continuum hypothesis is not valid anymore, since its size is of the order of the molecular mean free path [35]. This instability generates waves on the lamella that might cause the liquid to touch the surface, resulting in contact points as observed by Kolinski et al. [76], although this is not valid when the lift force deviates the liquid upwards [103]. Riboux and Gordillo [131] proposed a decomposition of this force into two contributions, one due to the gas dynamics, which takes into account the non-continuum effects near the contact line, and the other due to the suction force acting on the upper part of the lamella.

The air surrounding the drop and its interaction with the expanding circular liquid jet after impact gives rise, for certain impact conditions in which the Reynolds number is sufficiently high, to azimuthal undulations in the lamella with a fingering pattern which were first observed by Worthington [178]. The mechanism that triggers the

growth of fingers in the periphery of the lamella has been proposed to be related to the Rayleigh-Taylor instability (two immiscible fluids of different densities accelerated towards each other), although there is no consensus on the nature by which this instability first appears through the impact process. Allen [6] stated that this instability is due to the deceleration of the lamella and, following a model for the wave amplitude at the edge of the lamella proposed by Chandrasekhar [22], made an estimation of the deceleration. However, Thoroddsen and Sakakibara [164] explained that it is due to the rapidly decelerating annular ring of the fluid that first touches the surface but before hitting it. With the aim to predict the number of fingers, Marmanis and Thoroddsen [106] proposed a correlation based on the Weber and a modified Reynolds numbers while Aziz and Chandra [9] presented a distinct formula for molten droplets.

A different effect of the air surrounding the drop was found by Xu et al. [182]: reduction of ambient pressure leads to suppression of splash. There are some discrepancies between this behavior, which has been confirmed by the experiments, and the mechanism proposed by Mandre and Brenner [103] for splashing. The role of the air is not only related to the formation of the splash but also to the formation of a dimple in the drop, in the zone around the impact point, when it is close to impact the surface. The increased pressure in the thin air film causes the drop to deform and trap an air disc [72]. This disc tends to minimize its surface energy, contracting and eventually resulting in a bubble. The rapid receding of the contact line in the disc might cause the formation of a bubble ring concentric with the central bubble. As the contraction occurs much faster than typical wetting processes, surface wettability is supposed not to affect the formation of the bubble ring [163]. On the other hand, Lee et al. [80] found that surface wettability is a critical parameter in the bubble detachment from the surface, since increasing the contact angle increases the adhesion force that tends to attach the bubble.

The air disc radius decreases exponentially with time and, in certain circumstances for Oh numbers sufficiently low, a complex process may take place in which, due to the capillary waves caused by the rapid contraction of the disc that propagate towards the bubble center, the upper surface of the bubble touches the solid surface adopting a toroidal shape. Then, the self-coalescence of the toroidal bubble results in a liquid droplet adhered to the surface inside the bubble [80, 83, 163].

Air bubble entrapment seems to be related to impacts in the high viscosity regime [41], and the physical mechanisms that cause the splash in this regime, which still remain obscure. Palacios et al. [117] found the formation of a second microbubble ring. They pointed out the relation between this bubble ring and the formation of a lifted thin liquid film, and also observed that, when the impact velocity exceeds a critical value, an abrupt change in size of this second ring is produced, which influences the splash flow pattern.

It seems obvious that the characteristics of the impact surface, including roughness, porosity and wettability, should also affect the outcome obtained. Roisman et al. [138] investigated the effect of the roughness and porosity on the splash. Since these are not uniform properties, some statistical values should be used. The authors argued that the ratio between the average surface roughness, R_a, and the initial drop diameter cannot be used as a relevant scale for characterizing the influence of roughness on the splash threshold since R_a does not give detailed information about the surface [127]. Therefore, the authors used the ratio between the average height of the protruding peaks above the roughness core profile and the mean width of a profile element, which represents a characteristic slope of the surface topography. They found that increasing this ratio, and therefore the roughness of the surface, for a given We, promotes splashing. On the other hand, porosity tends to suppress splashing, probably due to a partial penetration of the drop into the pores during the first instants of the impact. These results are in partial agreement with those obtained by Kim et al. [74]: an increase in surface roughness promotes splashing until a certain roughness size is reached, for which the effect becomes similar to that of porosity.

The wettability of the solid surface, characterized by the contact angle, also influences the onset of splashing, since, under certain conditions, the liquid detaches directly from the surface. Yokoi [184] found that splashing can be triggered by increasing the hydrophobicity of the surface, and thus increasing the contact angle. This was also reported by Quetzeri-Santiago et al. [125], although they stated that the influence of this parameter is only relevant in hydrophobic surfaces, whereas in hydrophilic surfaces splashing remains independent of the contact angle. However, Latka et al. [78] and Roisman et al. [138] pointed out that the splash is independent of the surface wettability, since the splash occurs before the liquid of the lifted film touches the surface. These findings are in agreement with the numerical simulations carried out by Jian et al. [69], in which they showed that both prompt and corona splashing are independent of the substrate. Quintero et al. [126] and Riboux and Gordillo [131] took into account in their theoretical models the contact angle through an empirical parameter related to the contact angle for hydrophobic surfaces and through the angle established between the advancing front of the lamella above a lubricating air layer and the solid surface for hydrophilic substrates. For the latter type of surface, de Goede et al. [36] found that the angle just mentioned remains nearly constant for different surface wettabilities, hence splashing would be independent of surface wettability.

Despite this controversy on the effect of the contact angle in the splash, when the outcome results in a deposition, the agreement is much clear. For the same impact conditions and fluids but varying the surface wettability Šikalo et al. [151] carried out several experiments. At the first instants, when the inertial effects dominate over the viscous and capillary forces, the contact angle does not affect the spreading dynamics. However, in the following stages, especially close to the maximum spread factor, its

effect is very important. When the surface wettability is reduced, i.e., when the contact angle is increased, the maximum spread factor is smaller and the retraction of the lamella starts earlier. This result coincides with that showed in [81, 166, 170, 173, 176] and might be caused by the fact that as the contact line velocity is reduced, due to the viscous dissipation at the surface at the lasts stages of the spreading stage, the capillary force equals the inertial force. As the former is opposed to the latter when the contact angle is increased above 90° due to the interface curvature, the contact line velocity is also reduced, even changing its direction and making the spread factor to decrease. This phenomenon can be observed when the time evolution of the contact angle is measured in a drop impact on a solid surface, like in Šikalo et al. [152, 153]. In these works, the authors showed that at the first instants of the impact, the contact angle takes values above 90°, independently of the surface wettability and, as the impact evolves, the contact angle decreases with very different results depending on the impact conditions and the surface characteristics.

Despite this controversy on the effect of contact angle on splashing, when the impact outcome is deposition, the agreement is clearer. Šikalo et al. [151] carried out several experiments for the same impact conditions and fluids, but varying the surface wettability. In the first instants, when the inertial effects dominate over the viscous and capillary forces, the contact angle does not affect the spreading dynamics. However, in the subsequent stages, especially close to the instant of maximum spreading factor, its effect is very important. When the surface wettability is reduced, i.e., when the contact angle is increased, the maximum spread factor is smaller and the retraction of the lamella starts earlier. This result coincides with that shown in [81, 166, 170, 173, 176] and might be caused by the fact that, as the contact line velocity is reduced, the capillary force becomes of the same order as the inertial force. As the former is opposed to the latter, the contact line velocity is further reduced, eventually changing its direction and making the spreading factor to decrease. This effect is larger the larger is the contact angle and can be observed when measuring the time evolution of the contact angle during the impact of a drop on a solid surface, as in [152, 153]. In these works, the authors showed that in the first instants of the impact the contact angle takes values above 90° independently of the surface wettability and, as the impact evolves, the contact angle decreases with very different behavior depending on the impact conditions and the surface characteristics.

1.3 Contact angle and contact line

The main difficulty in achieving a more complete understanding of the mechanism by which a liquid front advances over a solid surface is due to the multiple length scales involved in the dynamics of the wetting process, which range from the macroscopic to the molecular scales, and the very wide range of possible wetting conditions (see,

e.g., [12, 16, 34]). A possible approach, although computationally expensive, is based on molecular dynamics simulations [77], which addresses the liquid/solid interaction in a natural way, coupled or not to models that solve the scales within the continuum limit [122]. Another approach is based on the hydrodynamic theory, from whose point of view this type of two-phase flow introduces a shear-stress singularity in the Navier-Stokes equations at the contact line when a no-slip boundary condition is imposed, since continuum mechanics is no longer valid at molecular distances from the contact line [158] (below ∼10 nm for fluids such as water under normal pressure and temperature conditions [13, 73, 156]). From a numerical perspective, the molecular interactions between the fluids and solid would have to be modelled somehow to eliminate the stress singularity. With this aim, a number of approaches have been proposed, such as those of the precursor film, the diffuse interface, and the slip models [2].

Slip models are based on allowing the contact line to move, and their implementations depend on the method used to track the interface evolution. The most common slip model used in the literature is based on the Navier slip boundary condition, which introduces a slip length proportional to the shear stress [82, 104]. Also often used is the generalized Navier boundary condition (GNBC) [51, 124, 128], in which slipping depends on the tangential viscous stress and the unbalanced Young's stress that arises from the deviation between the angle at which the interface intersects the solid boundary (contact angle) and its static value.

The definition of contact angle has been already introduced in Section 1.1, although not in a precise way, since the aim was only to illustrate the dependence on this quantity of the drop impact on a solid surface. It is important to note that there is not a unique definition, since it mainly depends on the length scale considered, although the concept is always the same as that described in Section 1.1. The equilibrium contact angle, θ_e, on a perfectly flat and chemically homogeneous surface in thermodynamic equilibrium, given by the minimization of the Gibbs free energy, satisfies Young's equation, which provides a good approximation of the contact angle at the macroscopic scale [34],

$$\sigma \cos\theta_e = \sigma_{sg} - \sigma_{sl}, \tag{1.5}$$

where σ, σ_{sg} and σ_{sl} are the liquid/gas surface tension and surface free energy densities of the solid/gas and solid/liquid interfaces, respectively (Fig. 1.3(a)). Heterogeneities and roughness of the solid surface lead to the hysteresis of the static contact angle [43], θ_s, whose value may vary in the range $\theta_r < \theta_s < \theta_a$, where θ_r and θ_a are called receding and advancing contact angles (see Fig. 1.3(b) for $u_{\mathrm{cl}} = 0$). How the contact line motion and the flow in its proximity influence the contact angle is a complex question, which is the subject of intense research activity.

In the mathematical model based on the hydrodynamic theory, on which numerical simulations of interfacial flows involving contact lines are usually based, three length scales are considered: (1) the microscopic scale (the smallest length within the contin-

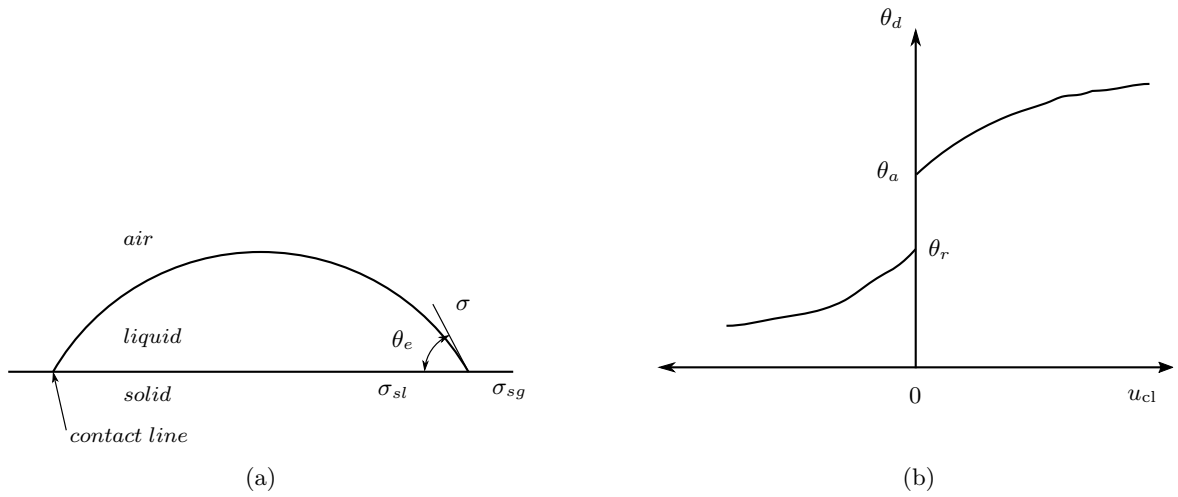


Figure 1.3: (a) Schematic representation of the contact line and θ_e. (b) θ_d as a function of $u_{\rm cl}$ and hysteresis of θ_s ($u_{\rm cl}$=0).

uum limit) where the contact angle (which could be assumed either to be equal to the static angle or to be dependent on the contact line velocity [172]) is governed by the intermolecular forces; (2) an intermediate length scale where viscous bending of the interface becomes relevant; and (3) the macroscopic scale which is of the order of the capillary length, where the contact angle corresponds to that measured experimentally. This last contact angle is usually referred to as the apparent contact angle and is often used as a boundary condition in the numerical resolution of the hydrodynamic equations [1, 42, 158]. The dynamic contact angles obtained experimentally are obviously apparent contact angles, and in most cases the length scale on which they have been measured depends on the spatial resolution of the measurement system and is not accurately known. This is an unavoidable drawback when imposing an apparent contact angle as a boundary condition as if it was the dynamic angle corresponding to the microscopic or intermediate scales.

The contact angle that is measured when the contact line moves, θ_d, differs from the static values, evidencing that the wetting process must be dissipative. It is generally observed that the measured advancing and receding dynamic contact angles are at any instant higher and lower, respectively, than the corresponding static values ($\theta_d \geq \theta_a$, $\theta_d \leq \theta_r$), and that the relationship between θ_d and the contact line velocity is monotonic (Fig. 1.3(b)). The deviation of the dynamic contact angle from its value θ_s at static conditions induces an unbalanced stress in Young's equilibrium since $\theta_d \neq \theta_s$. Taking into account this behavior, Eq. (1.5) can be rewritten as

$$\sigma_{sg} - \sigma_{sl} - \sigma\cos\theta_d = \sigma(\cos\theta_s - \cos\theta_d), \tag{1.6}$$

where the right hand side corresponds to the unbalanced Young's force or contact line force [34]. As this force tends to restore the Young's equilibrium, it causes the contact

line to advance when $\cos\theta_d < \cos\theta_s$ and to recede when $\cos\theta_d > \cos\theta_s$, and is zero when Young's equilibrium is reached ($\theta_d = \theta_s$).

With the aim to predict the contact angle using different flow parameters, several dynamic contact angle models have been proposed in the literature. These usually assume that the contact angle depends on the contact line velocity, as well as on other fluid parameters. This dependence is mathematically introduced through the capillary number $\mathrm{Ca} = \mu\, u_{\mathrm{cl}}/\sigma$, which represents the ratio of the viscous forces to the surface tension forces in the vicinity of the contact line at the interface.

The model proposed by Kistler [75] is based on a correlation of data obtained from the experiments carried out by Hoffman [62] in a study on the dynamic contact angle in capillary tubes for a wide range of capillary numbers, $4 \times 10^{-5} < \mathrm{Ca} < 36$. The dynamic contact angle is assumed to be a function of Ca and θ_e,

$$\theta_d = f_H(\mathrm{Ca} + f_H^{-1}(\theta_e)), \tag{1.7}$$

where $f_H(x)$ is the Hoffman function, defined as

$$f_H(x) = \arccos\left\{1 - 2\tanh\left[5.16\left(\frac{x}{1 + 1.13\,x^{0.99}}\right)^{0.706}\right]\right\}. \tag{1.8}$$

The model proposed by Jiang et al. [70] is also based on an empirical correlation to the data provided by Hoffman [62]. The dynamic contact angle is now obtained through

$$\frac{\cos\theta_e - \cos\theta_d}{\cos\theta_e + 1} = \tanh\left(4.96\,\mathrm{Ca}^{0.702}\right), \tag{1.9}$$

where θ_e can again be replaced by the static advancing and receding values.

In the Cox model [30, 44], the dynamic contact angle is calculated as

$$\theta_d = g^{-1}\left(g(\theta_m) + \mathrm{Ca}\,\log\frac{L}{l_m}\right), \tag{1.10}$$

where θ_m is the microscopic contact angle, L an intermediate region length scale, and l_m the slip length. The function $g(\theta)$ can be obtained through

$$g(\theta) = \int_0^\theta \frac{x - \sin x\,\cos x}{2\sin x}\,\mathrm{d}x \tag{1.11}$$

when $\mu_g/\mu_l \ll 1$. Furthermore, $g(\theta)$ and $g^{-1}(\theta)$ can be approximated using polynomial expressions, as stated in [42], as, respectively,

$$g(\theta) \approx \frac{\theta^3}{9} - 0.00183985\,\theta^{4.5} + 1.845823 \times 10^{-6}\,\theta^{12.258487}, \tag{1.12}$$

and

$$g^{-1}(\theta) \approx (9\,\theta)^{-3} + 0.0727387\,\theta - 0.0515388\,\theta^2 + 0.00341336\,\theta^3. \tag{1.13}$$

The mathematical model developed by Shikhmurzaev [148] relies on an interface formation-destruction process, considering the fluid motion as rolling. Under certain conditions, there are asymptotic limits at which this model can be simplified [149]. Blake and Shikhmurzaev [11] showed that one of such limits is the steady motion at low Ca and Re numbers when other nearby boundaries do not affect the flow near the contact line, for which the dynamic contact angle and a dimensionless contact line velocity, $V = S_c\,\mathrm{Ca}$, are related through

$$\cos\theta_e - \cos\theta_d = \frac{2V\left[\rho_{2e}^{s*} + \rho_{1e}^{s*}\,u_0(\theta_d,0)\right]}{\left[1-\rho_{1e}^{s*}\,u_0(\theta_d,0)\right]\left[(\rho_{2e}^{s*}+V^2)^{1/2}+V\right]}, \tag{1.14}$$

where

$$u_0(\theta_d,0) = \frac{\sin\theta_d - \theta_d\cos\theta_d}{\sin\theta_d\cos\theta_d - \theta_d}, \tag{1.15}$$

S_c is a scaling factor that depends on the material properties, and the parameter ρ_{2e}^{s*} is defined as

$$\rho_{2e}^{s*} = 1 + \left[1-\rho_{1e}^{s*}\,u_0(\theta_d,0)\right](\cos\theta_e - \sigma_{\mathrm{SG}}^{*}), \tag{1.16}$$

being σ_{SG}^{*} and ρ_{1e}^{s*} experimental parameters. Thus, this model provides a theoretical curve in the plane (θ_d, Ca) that depends on the three dimensionless viscosity-independent parameters indicated above.

It is important to note that Jiang's and Shikhmurzaev's models are only valid for advancing contact lines, i.e., they do not consider negative capillary numbers. To overcome such problem, the following correlation presented by Tanner [162] can be used instead,

$$\theta_d = (\theta_e + 72\,\mathrm{Ca})^{1/3}. \tag{1.17}$$

Figure 1.4 shows the dynamic contact angle curves in the plane (θ_d, Ca) obtained using the previous described models for an equilibrium contact angle $\theta_e = 60°$. For Shikhmurzaev's model, the empirical values $S_c = 4.3$, $\rho_{1e}^{s*} = 0.63$ and $\sigma_{\mathrm{SG}}^{*} = -0.08$ estimated by Blake and Shikhmurzaev [11] for a low-viscosity water-glycerol mixture have been used, and for Cox's model, the values $l_m = 10^{-9}\,\mathrm{m}$ and $L = 9\times10^{-6}\,\mathrm{m}$ have been applied. As Ca increases from 0, the models predict a rising contact angle until they reach a limit, which in all cases, except in the Shikmurzaev's model, is 180°. When the contact line recedes (Ca<0) the models by Kistler, Cox and Tanner yield very similar curves with a huge slope. It can be observed that the change in the dynamic contact angle for negative capillary numbers is very fast compared to positive capillary numbers. In the former, the limit value $\theta_d = 0°$ is reached at $\mathrm{Ca} \approx -0.015$ while in the latter, the limit is reached for much higher Ca values (considering its magnitude). This rapid change in the receding contact angle might be a result of the mathematical models instead of a physical phenomenon, and more deep studies in this subject are needed. Note that these models can be used to account for the hysteresis

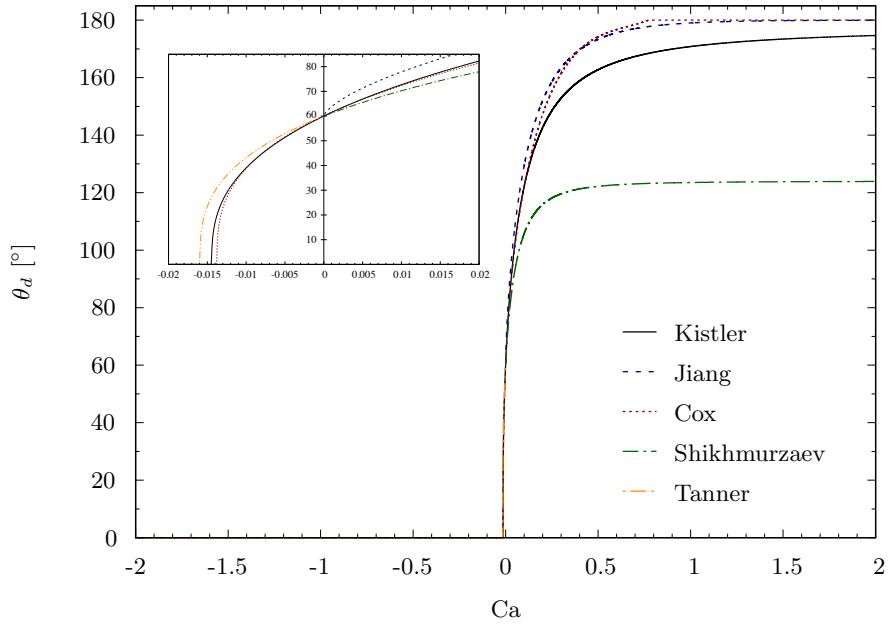


Figure 1.4: Dynamic contact angle as a function of Ca for different models and $\theta_e = 60°$. The inset shows a detailed view of the receding part of the curves (Ca < 0).

of the contact angle by substituting θ_e by θ_r or θ_a, depending on the relative direction of the contact line movement.

1.4 Volume of fluid method

As stated in Section 1.1, in this work, the interface evolution tracking is carried out using the VOF methodology, in which two types of methods are distinguished depending on the approach used to solve advection equation that describes the evolution of the indicator function. The algebraic VOF methods do not need to reconstruct the interface as it is represented using a numerical approximation (such as a polynomial or trigonometric function) and the interface advection is carried out using an algebraic procedure, generally based on finite-difference techniques, to compute the fluxes. They can be divided into two main groups: compressive schemes [112, 169], which introduce a compressive term into the advection equation in order to maintain the interface as sharp as possible; and THINC (tangent of hyperbola for interface capturing) schemes [180], in which a hyperbolic-tangent profile is assumed for the indicator function for a cell containing the interface.

Geometric VOF methods solve the advection equation performing both interface reconstruction and advection steps. Due to the complexity and the variety of geometric VOF methods available in the literature, these steps are briefly described in the following. Detailed reviews of this type of methods and recent developments can be found in [49, 105, 109, 168].

Interface reconstruction

The early reconstruction methods of type SLIC [61, 114] were improved by Youngs [187], who avoided the orientation limitation by extending the method to allow the interface to have an arbitrary orientation based on the volume fraction gradient using finite-difference formulas or a least-square technique. This type of interface reconstruction is usually referred to as PLIC, where the interface is represented by a plane defined as

$$\mathbf{x} \cdot \mathbf{n} + C = 0, \tag{1.18}$$

where C is a constant (minimum distance from the interface to the origin of the coordinate system), $\mathbf{n}$ is the unit vector normal to the interface pointing to the liquid, and $\mathbf{x}$ is the location of a generic point on the interface. In order to improve the normal calculation, several approaches have been used, among which can be mentioned the following: Chorin [26] introduced an iterative formulation based on calculating the interface orientation from a curve defined by a 3×3 array of volume fractions and then improved by Swartz [160] for performing well in unstructured meshes, described in algorithmic form by Mosso et al. [110]; Pilliod and Puckett [121] presented the efficient least-squares volume of fluid interface reconstruction algorithm (ELVIRA), which is second-order accurate but at a high computational cost, especially for 3D; Liovic et al. [85] proposed the 3D second-order convergence CVTNA method, based on the 2D method proposed by Swartz [160]; Scardovelli and Zaleski [144] proposed a least squares fit (LSF) algorithm for the normal calculation which was extended to 3D by Aulisa et al. [8], achieving second-order convergence when applied iteratively at a small computational cost; López et al. [97] developed the CLCIR (conservative level-contour interface reconstruction) method, which involves a non-iterative method based on isosurface extraction and was extended to arbitrary meshes by López et al. [87]; Roenby et al. [136] proposed an advanced VOF method, referred to as isoAdvector, based on the extraction of isosurfaces and in which, therefore, the interface normal is estimated using the extracted isosurface; Scheufler and Roenby [145] used the RDF from Cummins et al. [31] in order to improve iteratively the interface normal estimation for unstructured meshes in a PLIC method coupled to isoAdvector.

Other interface reconstruction methods based on the extraction of isosurface were developed by López et al. [97], in which the interface orientation is obtained by a weighted-average procedure. This procedure is based on triangulated surfaces constructed from the extraction of generally non-planar isosurfaces corresponding to a 0.5 value of the fluid volume fraction distribution. In particular, the LLCIR (local level-contour interface reconstruction) method constructs the triangulated surface by joining the vertices and centroid of the extracted isosurface in the cell, the ELCIR (extended level-contour interface reconstruction) method which constructs the triangulated surface by joining the centroid of the isosurface extracted in the cell with

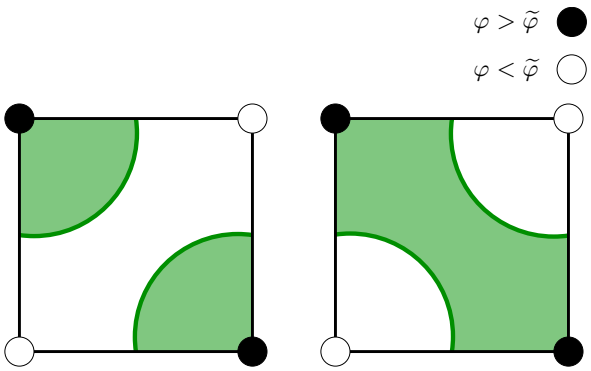


Figure 1.5: 2D example of an ambiguous situation produced in the isosurface extraction from discrete data.

those of the isosurfaces extracted in adjacent cells, the already mentioned CLCIR method translates the above mentioned centroids to the corresponding PLIC geometric centers, and the CLC-CBIR (conservative level-contour cubic-Bézier-based interface reconstruction) improves the orientation obtained from the CLCIR method by constructing a cubic-Bézier patch over each triangle of the triangulated surface.

To compute the interface normal, these and some other methods mentioned above need to solve the isosurface extraction problem, which is not a trivial task. Thus, the following describes how this problem is posed. Given a scalar field $\varphi : \mathbb{R}^3 \to \mathbb{R}$ and a value $\widetilde{\varphi} \in \mathbb{R}$, the goal is to extract the set $\{(x, y, z) : \varphi(x, y, z) = \widetilde{\varphi}\}$, which is usually called an isosurface. In this work, it will be assumed that φ is defined by discrete data available only at the cell vertices of a mesh (the cell volume fraction distribution interpolated to the mesh nodes). Therefore, the location of the extracted isosurface points must be approximated by interpolation, and the reconstructed isosurface can be represented, for example, with linear edges constructed by sequentially connecting the obtained isosurface points ordered by some tracing procedure. It should be mentioned that the use of discrete data may introduce ambiguities like that shown in the example of Fig. 1.5. These situations can produce inconsistencies in isosurface extraction and can be solved by increasing the number of sample points.

There are several methods that use polygons to reconstruct isosurfaces from discrete data; among them, the "marching cubes" algorithm, originally introduced by Lorensen and Cline [98] for 3D medical data disposed in a cubic mesh, is one of the best known. This algorithm creates a triangular surface of a given $\widetilde{\varphi}$ level. In an individual cube there are 2^8 different configurations based on the relative values of the φ-sampled data assigned to its 8 vertices with respect to $\widetilde{\varphi}$. The original algorithm proposed in [98] uses two different symmetries to reduce the number of distinctive configurations to only 14 (see Fig. 3 in [98]), which are stored in a lookup table to speed up execution of the algorithm. However, this reduction introduces a topological inconsistency in the vertex data when at least one face of the cube contains two opposite vertices, one on one side of the isosurface, and the other on the other side (similar to the ambiguous

situation shown in Fig. 1.5). This inconsistency produces isosurfaces that may have holes and may not be a 2-manifold (see the example of Fig. 1.6).

Several efforts have been made to avoid this inconsistency and to extend the use of the original marching cubes algorithm to other mesh types (see, e.g., [10, 24, 40, 107, 113, 174, 189, 190]). Recently, López et al. [87] presented a new isosurface extraction method, which can be considered as an extension of the marching cubes technique, that produces consistent results even for ambiguous situations in polyhedra of arbitrary shape. This last aspect makes the method very powerful, since it can be used with codes capable of simulating different physical phenomena using complex unstructured meshes with cells without any predetermined geometric configuration.

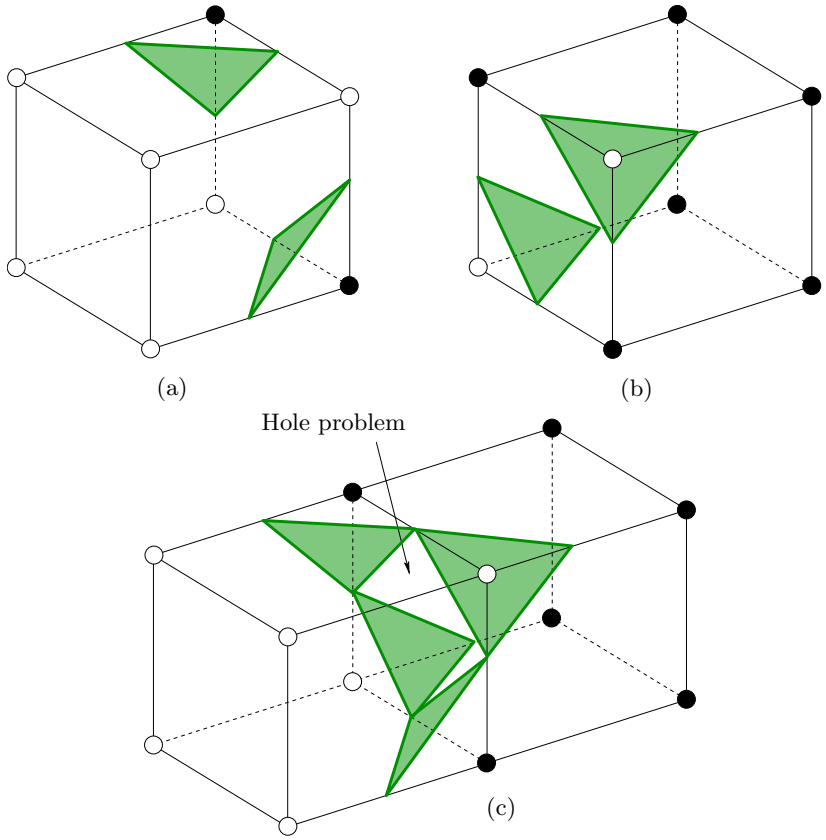


Figure 1.6: Inconsistency of the original marching cubes algorithm. A combination of cells with (a) configuration 3 of Fig. 3 in [98] and (b) the complementary symmetric case produces (c) the hole observed on the bottom picture.

Once that the interface normal $\mathbf{n}$ has been computed, the next step is to position the PLIC by computing the constant C so that the interface splits the cell Ω into two sub-polyhedral cells of volumes $V_T = FV_{c,\Omega}$ and $(1-F)V_{c,\Omega}$. This problem is usually referred to as the volume conservation enforcement (VCE) problem, which involves the resolution of

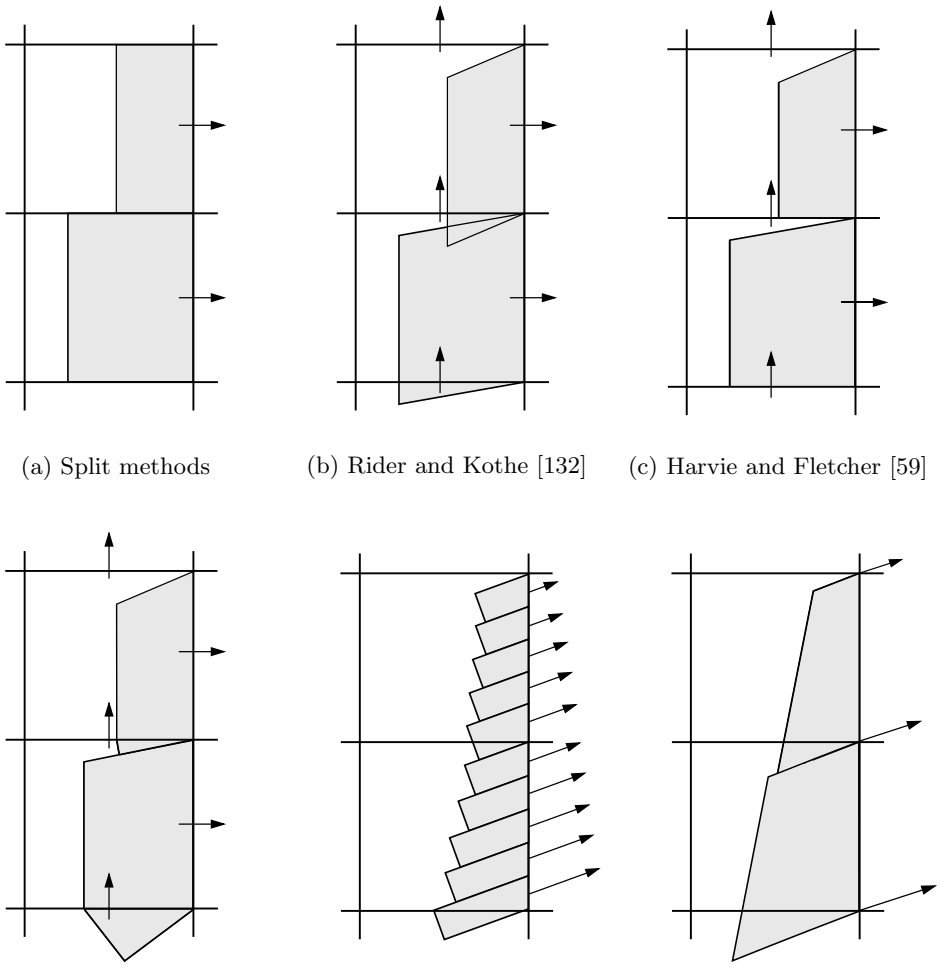
$$V(C) - V_T = 0, \tag{1.19}$$

where $V(C)$ is the volume of the sub-polyhedral cell to which $\mathbf{n}$ points. Both analytical and iterative approaches have been proposed to solve this problem, since inverting $V(C)$, i.e., obtaining $C = C(V_T)$, is not a trivial task. This problem is usually divided into two steps [93]: firstly, the solution bracketing step, where two bounds for the solution of Eq. (1.19) are found usually truncating the polyhedral cell using cutting planes parallel to $\mathbf{n}\cdot\mathbf{x} = 0$ and passing through different cell vertices; secondly, the final calculation step, where the value of C is computed iteratively or analytically. Rider and Kothe [132] solved this problem iteratively by using Brent's method [19]. Gueyffier et al. [54] and Scardovelli and Zaleski [143] extended the analytical approach proposed by Li [84] for square cells to orthogonal rectangular and hexahedral cells. Harvie and Fletcher [58] introduced an analytic method, valid for orthogonal rectangular cells, in which a series of logical steps to bracket the solution and an analytical method in the final calculation step are employed. The first analytical approach valid for general convex meshes in both two and three dimensions was proposed by López and Hernández [88], improved by López et al. [93, 96] using more efficient formulae to compute areas and volumes and extended to arbitrary meshes, with either convex or non-convex cells, by López et al. [95]. Other recent iterative methods are the proposed by Ahn and Shashkov [3], Skarysz et al. [154] and Chen and Zhang [23].

Geometric advection

Geometric multi-dimension advection methods can be classified into two categories: split and unsplit. The former are limited only to regular structured meshes, and are based on solving the problem in two (2D) or three (3D) steps every time step, one for each spatial direction [8, 144]. The lack of accuracy in the calculation of donating regions (Fig. 1.7(a)), yields to geometric errors that distort the interface, which also causes the global and local volume conservation to be not satisfied, making necessary the use of ad-hoc fluid redistribution algorithms. Thus, in this type of methods. Besides, this type of methods require an interface reconstruction step after every advection step, which substantially increases the computation time.

On the contrary, unsplit methods are harder to implement but they can be applied to unstructured meshes. In the scheme proposed by Rider and Kothe [132], the donating flux regions have a trapezoidal shape and are constructed from the velocity components normal to the cell faces (Fig. 1.7(b)). Note that these regions might be overlapped without ensuring strict volume conservation and introducing numerical diffusion. The methods proposed by Harvie and Fletcher [59] and Pilliod and Puckett [121] avoid the ovelarpping of the donating regions, although its calculation is limited to the normal components of the velocity (Figs. 1.7(c) and 1.7(d)). Other method proposed by Harvie and Fletcher [58] calculates the donating regions on a more precise way by joining several sub-regions constructed along the cell face, although it is not strictly conservative (Fig. 1.7(e)). The edge-matched flux polygons



(a) Split methods (b) Rider and Kothe [132] (c) Harvie and Fletcher [59]

(d) Pilliod and Puckett [121] (e) Harvie and Fletcher [58] (f) López et al. [91]

Figure 1.7: Donating flux regions for the main geometric advection methods in two dimensions.

advection (EMFPA) method proposed by López et al. [91, 92] constructs the donating flux regions using the information of the cell-vertex velocities and the cell-face normal velocities in order to correct the volume of the donating region (Fig. 1.7(f)). Since the extension to 3D of this method requires complex geometric operations over the donating regions, Hernández et al. [60] presented a modification of the method, the face-matched flux polyhedra advection (FMFPA), which simplifies its implementation but does not avoid the overlapping of the donating flux regions when two cells share only a vertex. However, they also extended their EMFPA method to three dimensions [96], as well as Owkes and Desjardins [116], Jofre et al. [71] or Ivey and Moin [67].

A different approach for the interface advection was developed by Roenby et al. [136] in their isoAdvector method. It is based on the estimation of the interface movement within a time interval to obtain the flux on each cell face.

VOF methods in OpenFOAM

Initially, OpenFOAM only had an algebraic VOF method implemented, usually referred to as MULES, which is described in a next chapter. However, in the recent years, several authors have contributed to the extension of the capabilities of the software in this field of investigation by developing and implementing both algebraic and geometric VOF methods, some of which have been publicly released. Albadawi et al. [5] implemented a coupled level-set volume of fluid (CLSVOF) method making use of the MULES advection scheme for the volume fraction advection. Cifani et al. [27] implemented a PLIC-based VOF method using the advection algorithm proposed by Puckett et al. [123]. They employed the volume fraction gradient for the interface orientation calculation along with the analytical method of Scardovelli and Zaleski [143] to solve the VCE problem for the PLIC positioning, thus limiting the application of the VOF method only to cubic meshes.

Roenby et al. [136] implemented the isoAdvector method in OpenFOAM. They carried out several advection tests, obtaining good results in terms of volume conservation, boundedness and efficiency. They released the code as an extension of OpenFOAM. Dai and Tong [32] modified the reconstruction step of the isoAdvector algorithm in order to implement several PLIC-based analytical interface reconstruction methods for 2D polygonal unstructured meshes, and then extended this procedure to arbitrary convex polyhedral cells [33]. They conducted interface reconstruction tests into single cells and simulated both 2D and 3D incompressible multiphase flow cases. They also released their implementation as open-source code.

Dianat et al. [39] implemented a CLSVOF in OpenFOAM. They used the methods presented by Ahn and Shashkov [4] for the PLIC positioning, the gradient of the level-set function for the interface normal calculation and the MULES algorithm for the interface advection. This methodology was validated for hexahedral and tetrahedral meshes using several advection tests and coupling it with the Navier-Stokes equations OpenFOAM's solver, for the simulation of the impact of a drop onto a solid surface. Following this work, Skarysz et al. [155] introduced an iterative interface reconstruction method valid for convex cells, which is based on tetrahedral cell decomposition for the volume calculation of the truncated polyhedron. Haghshenas et al. [57] also implemented a CLSVOF method in OpenFOAM.

1.5 Drop impact modeling

As shown in the previous sections, one of the major challenges when simulating a drop impact on a solid surface is to account for the influence of the surface characteristics and its relationship with the fluids, mainly due to the contact line problem. One of the most commonly used approaches to deal with the interaction between the fluids and the solid surface involves the use of the static or dynamic contact angles when

imposing boundary conditions at the contact line [177]. The dynamic contact angle is obtained using different models, such as that by Kistler [75] or Cox [30], described in the section above. The way to implement the boundary conditions at the contact line depends on the method used to track the interface evolution [177]. With the VOF method, the most common way is to force the orientation of the interface at the wall, along with the no-slip boundary condition. However, with this approach results are very sensitive to the contact angle model used, and most of the models do not describe well the dynamics of the contact line receding phase [147] (as shown in Section 1.3).

Other approach is based on considering the force acting on the contact line, given by Eq. (1.6), and try to reproduce it numerically to move the contact line, introducing it into the Navier-Stokes equations as a force per unit volume. Several studies have used a contact line force model to simulate droplet deposition and rebound. Šikalo et al. [153] considered a distributed force at the contact line, proportional to the surface tension and the cosine of the dynamic contact angle, which is applied within the framework of the continuum surface force (CSF) method [18]. Thus, they did not explicitly introduced a contact line force into the Navier-Stokes equations derived from Eq. (1.6), but imposed a surface tension force at the contact line that takes into account the interface orientation given by the dynamic contact angle of Kistler's model [75]. In their numerical simulations, carried out using a VOF method, the authors used a 2D axisymmetric domain and calculated the contact line velocity as the time derivative of the position of the interface at the wall. With such approach, they obtained good results for the spreading and receding stages. Deganello et al. [37] used a similar force model in combination with a 3D level set method. They used a diffuse interface that avoids local stress singularities and allows for contact line movement without the need for a slip condition.

Malgarinos et al. [102] implemented in a commercial CFD software a wetting force model based on the adhesion force model of Antonini et al. [7]. A contact line force, dependent on the static and dynamic angles and the lamella radius, was applied on various cells in the vicinity of the interface at the wall. The authors used a VOF method and a 2D axisymmetric domain to simulate several drop impacts for a wide range of Reynolds numbers and relatively low Weber numbers. They compared their numerical results with experiments made by other authors and with numerical results obtained using several dynamic contact angle models to impose the boundary condition at the contact line. Boelens and de Pablo [15] used the generalized Navier boundary condition with a VOF method in OpenFOAM. However, instead of relating the unbalanced Young's force to the contact line velocity, they introduced a distributed contact line force into the Navier-Stokes equations as a body force proportional to the surface tension, the cosine of the surface contact angle, the gradient of the volume fraction, and cell size. They implemented the Navier slip condition required by the GNBC framework despite the VOF method already introduces an "implicit" slip length at

no-slip boundaries [2]. However, as the authors argue, the introduction of the Navier slip condition in a VOF approach allows the contact line velocity to be calculated in a more precise way. They tested static and quasi-dynamic cases on a 2D domain, in which submillimetric droplets were released on smooth walls under zero gravity conditions.

Several experimental studies on the splash of a drop impacting onto a solid surface have been carried out to analyze the influence of the impact conditions on the flow pattern, as shown in Section 1.2. However, the number of numerical studies on this subject is considerably smaller, possibly due to the difficulty of reproducing the experiments with sufficient accuracy, sometimes even qualitatively. Bussmann et al. [20] used a 3D VOF method assuming a constant contact angle to impose the boundary condition at the contact line and, using an approach slightly different to that proposed by Gueyffier and Zaleski [55] and Rieber and Frohn [133] for the drop impact on a liquid film, applied a numeric perturbation on the radial velocity of the drop near the solid surface in order to trigger finger growth and subsequent detachment of secondary droplets. Yokoi [184, 185, 186] simulated a drop impact onto a solid surface for a high static contact angle (163°) using a 3D coupled level-set VOF (CLSVOF) method, imposing a constant contact angle at the contact line and obtaining a good qualitative agreement with experiments. Numerical errors triggered finger growth, so there was no need to perturb the velocity field. However, the results were very sensitive to the parameters of the continuum surface force model used. Guo et al. [56] used the Moment of Fluid method to simulate low- and high-speed drop impacts onto solid surfaces and liquid films in 3D and 2D axisymmetric computational domains, making use of several dynamic contact angle models at the solid walls. Jian et al. [69] simulated the impact of a drop onto a solid surface for fluids of different density and viscosity ratios using a 2D axisymmetric VOF method. They found that the dynamics of the contact line did not affect the impact outcome due to the rapid spreading of the liquid; therefore, they used a static contact angle equal to 90° in all simulations. They also found that the splashing outcome was only well predicted for relatively high gas to liquid viscosity ratios. Boelens and de Pablo [14] used the same approach as in [15] to simulate the splashing of high and low viscosity droplets using a 2D axisymmetric domain. Sun et al. [159] employed a 3D CLSVOF method for the simulation of a low speed drop impact onto a solid surface with a high static contact angle, although no information was provided on how the contact angle was imposed. Xavier et al. [179] used an algorithm based on the CLSVOF method for the 3D simulation, in a quarter of the physical domain, of a drop impacting onto a solid surface. The results showed a prompt splash, but the subsequent unstable corona that appeared in their experiments was not reproduced in the numerical simulations at ambient pressure. However, when the gas density was increased in the numerical simulations, the unstable corona was well predicted and the numerical results were in good agreement with the experiments.

On the other side, the amount of numerical studies involving drop impacts with deposition, receding breakup and rebound outcomes is much bigger (see, e.g., [53, 56, 72, 79, 82, 104]). This is due to the less complexity of the problem, which, in principle, makes it more feasible to simulate and less computational-resource demanding.

1.6 Objectives of the thesis

The focus of this thesis is on the numerical modeling of the drop impact on a solid surface by using efficient and accurate computational methods. The objectives of this work are:

- Development of an efficient model that improves the numerical simulation of the contact line dynamics and that is able to accurately capture the phenomena involved in the spreading and splashing of a drop on a solid surface.
- Analysis and implementation in OpenFOAM of advection and reconstruction VOF schemes to be used to track the interface evolution.
- Validation of the proposed contact line model on a wide range of impact conditions in deposition simulations, comparing the results obtained with experimental results and with numerical results obtained with other models.
- Validation of the advection and reconstruction VOF schemes by means of a consistent comparison of its efficiency and accuracy with those of other methods.
- Application of the proposed contact line model and the implemented advection and reconstruction schemes to a complex two-phase flow problem in which a drop impacts on a solid surface resulting in fingering and splashing outcomes and comparison of the numerical results with experimental results.

The remainder of the thesis is organized as follows. Chapter 2 presents the mathematical model used to describe the impact of a drop on a solid surface. Chapter 3 describes the discretization of the mass and momentum conservation equations, the numerical methods used for interface tracking, the proposed model for the contact line, the implementation of the interface tracking methods and other computational details. Chapter 4 focuses on the validation of the interface tracking methods. Chapter 5 presents the results for the drop impact problem and the comparison of the numerical results with experimental results. Finally, conclusions and future work are presented in Chapter 6.

Chapter 2

Mathematical model

The equations that describe the drop impact on a solid surface are presented along with the initial and boundary conditions required to solve the problem.

2.1 Governing equations

The impact of a drop on a solid surface is considered as an unsteady incompressible viscous flow of two immiscible fluids with constant and uniform properties separated by an interface, which can be described by the Navier-Stokes equations. Thus, the governing equations read

$$\nabla \cdot \mathbf{u} = 0, \tag{2.1}$$

$$\frac{\partial(\rho \mathbf{u})}{\partial t} + \nabla \cdot (\rho \mathbf{u}\mathbf{u}) = -\nabla p + \nabla \cdot \{\mu[\nabla \mathbf{u} + (\nabla \mathbf{u})^T]\} + \mathbf{f}_v, \tag{2.2}$$

where $\mathbf{u}$ is the velocity, ρ the density, μ the dynamic viscosity, p the pressure, and $\mathbf{f}_v$ any body force per unit volume. The properties ρ and μ are calculated as

$$\rho = \chi\rho_l + (1-\chi)\rho_g, \;\; \mu = \chi\mu_l + (1-\chi)\mu_g, \tag{2.3}$$

where the l and g subscripts denote liquid and gas, respectively, and $\chi = \chi(\mathbf{x}, t)$ is an indicator function used to identify if one of the fluids (in this case, the liquid) is present at a particular location $\mathbf{x}$ at instant t, considering the Eulerian frame of reference in which Eqs. (2.1) and (2.2) are written. χ is defined as a Heaviside function, continuous everywhere except at the interface, where it jumps from 0 to 1,

$$\chi(\mathbf{x}, t) = \begin{cases} 1 & \text{if } \mathbf{x} \text{ is in the liquid,} \\ 0 & \text{otherwise.} \end{cases} \tag{2.4}$$

If a fluid particle is followed on its movement, the function χ, assuming that there is no phase change, will not vary because the fluid particle will always belong to the same fluid. Thus, the substantial derivative of χ is zero, i.e., χ satisfies the advection

equation

$$\frac{\partial \chi}{\partial t} + \mathbf{u} \cdot \nabla \chi = 0. \tag{2.5}$$

In the impacts of drops with diameters of the order of millimeters, surface tension effects must be taken into account since the typical length associated to the free surface is usually of the same order as the capillary length. In order to do that, surface tension is included in Eq. (2.2) as a body force defined as

$$\mathbf{f}_\sigma = \sigma \kappa \mathbf{n} \delta_s, \tag{2.6}$$

where σ is the surface tension, κ is the interface curvature, $\mathbf{n}$ is the unit vector normal to the interface pointing to the liquid, and $\delta_s = \delta(\mathbf{x} - \mathbf{x}_s)$ is the Dirac delta function, where subscript s denotes interface location.

Gravity is also taken into account as a body force

$$\mathbf{f}_g = \rho \mathbf{g}, \tag{2.7}$$

where $\mathbf{g}$ is the gravity vector.

In this formulation of the problem, instead of using the governing equations separately for each fluid and then relate the solutions through jump conditions [168], only one set of governing equations for the entire flow domain is considered (Eqs. (2.1) and (2.2)), where the fluid properties change abruptly at the interface that separates the two phases (Eqs. (2.3) and (2.5)).

2.2 Initial and boundary conditions

The mathematical description of the flow requires to impose appropriate initial and boundary conditions at the first instant and at the domain boundaries, respectively.

The initial χ distribution is given by the shape of the drop, characterized by the drop diameter D_0, which is initially placed at a particular location above the solid surface and close to it. The initial velocity of the drop is the impact velocity, U, and the initial velocity of the gas is set to zero. The initial pressure at the gas phase is $p_{g,0}$ and the initial pressure within the drop, $p_{l,0}$, is given by the Laplace equation

$$p_{l,0} - p_{g,0} = \sigma \kappa = \frac{4\sigma}{D_0}. \tag{2.8}$$

At the solid surface, the fluids must have the same velocity as that of the surface, and, since there is no flux across the solid surface, the normal component of the pressure gradient at the solid surface must be zero. The rest of the boundaries are considered to be located far away from the impact zone, thus the pressure is $p_{g,0}$ and the velocity is zero.

Chapter 3

Numerical methods

In this chapter, the numerical methods used in the present thesis for the numerical simulation of the impact of a drop on solid surfaces are described. The open-source library OpenFOAM is used as computational framework. The finite volume method (FVM) is used for the discretization of the governing equations, the pressure-implicit with splitting operators (PISO) algorithm to treat the pressure-velocity coupling, and the VOF method to track the interface evolution. Also, a contact line force model used to improve the simulation of the contact line dynamics is presented and the solution procedure is described.

3.1 Finite volume discretization

The solution domain is divided into a finite number of contiguous cells (finite volumes) to which the conservation equations are applied, integrating them over each cell. Dependent variables such as velocity and pressure are stored at the centers of the cells. This storage arrangement is known as collocated mesh arrangement, although, in order to avoid the checkerboard problem, the volumetric fluxes are stored at the control volume faces centers using the Rhie-Chow momentum interpolation [130]. More detailed descriptions of the FVM can be consulted in References [48, 68, 111, 171].

In order to simplify the imposition of the pressure boundary condition in the numerical model [140], a modified pressure is defined as

$$p_d = p - \rho\, \mathbf{g} \cdot \mathbf{x}, \tag{3.1}$$

which is used instead of the pressure p. Therefore, p_d is introduced in the momentum equation by calculating the gradient of Eq. (3.1) as

$$\nabla p_d = \nabla p - \rho \mathbf{g} - \mathbf{g} \cdot \mathbf{x} \nabla \rho, \tag{3.2}$$

where the term $\mathbf{g} \cdot \mathbf{x} \nabla \rho$ is included into the Rhie-Chow interpolation.

Integration of Eqs. (2.1) and (2.2) over a cell Ω of volume V_Ω taking into account Eq. (3.2) yields, respectively,

$$\int_{V_\Omega} \nabla \cdot \mathbf{u}\, dV = 0, \tag{3.3}$$

$$\begin{aligned}\int_{V_\Omega} \frac{\partial(\rho\mathbf{u})}{\partial t}\, dV + \int_{V_\Omega} \nabla \cdot (\rho\mathbf{u}\mathbf{u})\, dV \\ = -\int_{V_\Omega} \nabla p_d\, dV + \int_{V_\Omega} \nabla \cdot (\mu\nabla\mathbf{u})\, dV - \int_{V_\Omega} \mathbf{g}\cdot\mathbf{x}\nabla\rho\, dV + \int_{V_\Omega} \sigma\kappa\mathbf{n}\delta_s\, dV.\end{aligned} \tag{3.4}$$

Taking a look at Eqs. (2.2) and (3.4), it can be observed that the diffusion term includes the second derivative of $\mathbf{u}$ in space, making the equation second order. Therefore, to achieve good accuracy, the order of the discretization must be, when possible, equal or higher than the order of the equation (to preserve the boundedness of the solution or take into account irregularities in the computational mesh structure, this accuracy requirement might be relaxed in some parts of the discretization [68]). Usually, the discretized variables are considered to vary linearly in space around the center of the cell, yielding a second-order accurate discretization in space. The midpoint rule is used to approximate the volume/surface integrals in Eqs. (3.3) and (3.4) as the product of the integrand at the cell/face center (which is an approximation to the mean value over the volume/surface) and the cell/face volume/area (note that some of the volume integrals in Eq. (3.4) are rewritten as surface integrals, as shown below). This approximation is second-order accurate and also the simplest. Other approximations could be used, such as the trapezoid rule or Simpson's rule, although at the cost of a more difficult implementation, since more values than that at the cell/face center are needed.

The fluid properties ρ and μ that appear in Eq. (3.4) are calculated with the help of Eq. (2.3). However, when the discretization of Eq. (3.4) is carried out, the scalar function χ is approximated by its discretized version, the volume fraction F, which, for a cell Ω, is defined as the fraction of the cell occupied by the liquid at time t and is calculated as

$$F = \frac{1}{V_\Omega}\int_{V_\Omega} \chi(\mathbf{x}, t)\, dV. \tag{3.5}$$

Therefore, the fluid properties at cell Ω are obtained as

$$\rho = F\rho_l + (1-F)\rho_g, \;\; \mu = F\mu_l + (1-F)\mu_g. \tag{3.6}$$

It should be noted that the volume fraction is also used in the discretization of the surface tension term in Eq. (3.4), as shown below.

3.1.1 Convection

If this term is discretized without any further approximation than that given by the FVM, the resulting system of discretized equations would be non-linear, which can only be solved using complex methods and at a high computation cost. To overcome this situation, the convective term is linearized using the divergence theorem as:

$$\int_{V_\Omega} \nabla \cdot (\rho \mathbf{u}\mathbf{u})\, dV \approx \sum_f \rho_f \mathbf{S}_f \cdot \mathbf{u}_f \mathbf{u}_f \approx \sum_f \rho_f \phi_f \mathbf{u}_f, \tag{3.7}$$

where the summation is performed over all cell faces, $\phi_f = \mathbf{S}_f \cdot \mathbf{u}_f$ is the volumetric flux through face f, $\mathbf{u}_f$ is the velocity at the center of face f, $\mathbf{S}_f$ is the area vector, and ρ_f is the density interpolated to face f.

Many discretization schemes have been proposed to obtain $\mathbf{u}_f$. If a linear variation is assumed between the center of the cell Ω, $\mathbf{x}_\Omega$, and the center of the neighbor cell N, $\mathbf{x}_N$, the value at the shared face f can be calculated as

$$\mathbf{u}_f = \mathbf{u}_\Omega + \frac{|\mathbf{x}_f - \mathbf{x}_\Omega|}{|\mathbf{x}_N - \mathbf{x}_\Omega|}(\mathbf{u}_N - \mathbf{u}_\Omega), \tag{3.8}$$

where $\mathbf{x}_f$ is the position of the face center, $\mathbf{u}_\Omega$ is the velocity at the cell Ω center, and $\mathbf{u}_N$ is the velocity at the cell N center. This scheme is called central differencing, which is second-order accurate but does not guarantee the boundedness of the solution since it does not preserve the directional nature of the convective term. An alternative is to determine $\mathbf{u}_f$ according to the direction of the flow, which is known as the upwind differencing (UD) scheme. If the flow point towards N or there is no flux across the face, $\mathbf{u}_f = \mathbf{u}_\Omega$, and $\mathbf{u}_f = \mathbf{u}_N$ otherwise. However, this scheme may introduce numerical diffusion, is only first-order accurate, but is very stable. An extension of this scheme is the second-order upwind or linear upwind differencing (LUD) scheme, which assumes a linear variation of $\mathbf{u}$ from the cell center to the considered face using $(\nabla \mathbf{u})_\Omega$ or $(\nabla \mathbf{u})_N$ to correct the corresponding upwind value of $\mathbf{u}_f$ depending on the direction of the flow, increasing the accuracy of the first-order upwind scheme.

Other relevant discretization schemes are, for example, the quadratic upstream interpolation for convective kinematics (QUICK), which is based on a quadratic polynomial interpolation, the FROMM scheme, which is similar to the second-order upwind but also using a downwind node, and higher resolution schemes, which are based on different techniques, such as the total variation diminishing (TVD), but all of them relying on the combination of the previously cited convection schemes. TVD schemes combine the upwind and central differencing schemes using a limiter function, ψ, which depends on the ratio of the upwind-side gradient to the downwind-side gradient. Many limiter functions have been developed, and some of the most used are: the van Leer function $\psi = (r + |r|)/(1 + r)$, the linear function $\psi = \max[\min(2r/k, 1), 0]$, where k is a constant parameter, and the Sweby function $\psi = \max[0, \min(kr, 1), \min(r, k)]$.

Then, the face value in a TVD scheme is obtained as

$$\mathbf{u}_f = [1 - \psi]\, \mathbf{u}_{f,UD} + \psi\, \mathbf{u}_{f,CD}, \tag{3.9}$$

where $\mathbf{u}_{f,UD}$ and $\mathbf{u}_{f,CD}$ are the face values obtained using the upwind and central differencing schemes, respectively. The assessment of the discretization schemes is usually evaluated using the conservativeness, boundedness and transportiveness properties (see References [111, 171]).

3.1.2 Diffusion

The diffusion term is discretized as

$$\int_{V_\Omega} \nabla \cdot (\mu \nabla \mathbf{u})\, dV = \sum_f \int_{S_f} (\mu \nabla \mathbf{u}) \cdot d\mathbf{S} \approx \sum_f \mu_f \mathbf{S}_f \cdot (\nabla \mathbf{u})_f, \tag{3.10}$$

where μ_f is de dynamic viscosity interpolated to the face center. The product $\mathbf{S}_f \cdot (\nabla \mathbf{u})_f$ can be obtained as

$$\mathbf{S}_f \cdot (\nabla \mathbf{u})_f = |\mathbf{S}_f| \frac{\mathbf{u}_N - \mathbf{u}_\Omega}{|\mathbf{d}_{\Omega N}|}, \tag{3.11}$$

if the computational mesh is orthogonal, where $\mathbf{d}_{\Omega N}$ is the vector joining the cells centers. If it is non-orthogonal, i.e., $\mathbf{d}_{\Omega N}$ and $\mathbf{S}_f$ are not parallel, the contribution of the non-orthogonality must be taken into account since the gradient has a component normal to $\mathbf{d}_{\Omega N}$. To accomplish that, vector $\mathbf{S}_f$ is decomposed as the sum of a vector parallel to $\mathbf{d}_{\Omega N}$, $\mathbf{k}_f$, and another vector, $\mathbf{l}_f = \mathbf{d}_{\Omega N} - \mathbf{k}_f$, being possible different decomposition methods. Thus, Eq. (3.11) can be rewritten as

$$\mathbf{S}_f \cdot (\nabla \mathbf{u})_f = \mathbf{k}_f \cdot (\nabla \mathbf{u})_f + \mathbf{l}_f \cdot (\nabla \mathbf{u})_f = |\mathbf{k}_f| \frac{\mathbf{u}_N - \mathbf{u}_\Omega}{|\mathbf{d}_{\Omega N}|} + \mathbf{l}_f \cdot (\nabla \mathbf{u})_f, \tag{3.12}$$

where the first term of the right side is the orthogonal contribution, and the second term the non-orthogonal correction. $(\nabla \mathbf{u})_f$ in the non-orthogonal correction is obtained by linear interpolation of the velocity gradients at the centers of the cells sharing the face. These gradients are calculated using the Green-Gauss theorem. Therefore, for a cell Ω, $(\nabla \mathbf{u})_\Omega$ is obtained following

$$(\nabla \mathbf{u})_\Omega \approx \frac{1}{V_\Omega} \sum_f \int_{S_f} \mathbf{u}_f\, d\mathbf{S} \approx \frac{1}{V_\Omega} \sum_f \mathbf{S}_f\, \mathbf{u}_f. \tag{3.13}$$

Depending on the decomposition method employed, vectors $\mathbf{k}_f$ and $\mathbf{l}_f$ will vary, although the direction of the former will not. Usually, three of them are considered [68]: (1) minimum correction, in which $\mathbf{l}_f$ is set perpendicular to $\mathbf{k}_f$; (2) orthogonal correction, in which the contribution from $\mathbf{u}_\Omega$ and $\mathbf{u}_N$ is kept the same as on the orthogonal mesh, $|\mathbf{k}_f| = |\mathbf{S}_f|$; and (3) over-relaxed, in which the importance of the orthogonal contribution increases as the non-orthogonality increases.

3.1.3 Transient term

Multitude of schemes have been proposed in order to obtain the full discretized momentum equation, i.e., to perform the time integration over a time interval $[t_1, t_2]$, and all of them differ on the known values inside the time interval. It is important to note that all non-transient terms are evaluated at the same time t using the midpoint rule [111].

The first-order implicit Euler scheme sets $t_1 = t - \Delta t$ and $t_2 = t$, where the time interval Δt is defined as a small increment of time. In this scheme, the information is needed at time t, thus, in order to obtain the value of $\mathbf{u}_\Omega$ at instant t, a system of algebraic equations needs to be solved. On the contrary, the first-order explicit Euler scheme sets $t_1 = t$ and $t_2 = t + \Delta t$, making possible to evaluate the term $\mathbf{u}_\Omega$ at time $t + \Delta t$ explicitly. Second-order schemes use a linear interpolation profile. If a central difference profile is used, it yields the Crank-Nicolson scheme, which, for a constant time step, discretizes the transient term as

$$\left.\frac{\partial(\rho\mathbf{u})_\Omega}{\partial t}\right|_t = \frac{(\rho\mathbf{u})_\Omega^{t+\Delta t} - (\rho\mathbf{u})_\Omega^{t-\Delta t}}{2\Delta t}. \tag{3.14}$$

The implicit second-order upwind Euler scheme uses the second-order upwind interpolation profile to discretize the transient term as

$$\left.\frac{\partial(\rho\mathbf{u})_\Omega}{\partial t}\right|_t = \frac{3(\rho\mathbf{u})_\Omega^{t} - 4(\rho\mathbf{u})_\Omega^{t-\Delta t} + (\rho\mathbf{u})_\Omega^{t-2\Delta t}}{2\Delta t}. \tag{3.15}$$

It is worth mentioning that in the previous temporal discretization schemes the assumption of constant time step has been taken into account. If that is not the case, second-order temporal schemes must be modified since they use a stencil involving two different time step values. Expressions for the Crank-Nicholson and second-order Euler upwind schemes can be found in [111]. In addition, since the first instant does not have an upwind neighbor and, in order to avoid large initial errors that will affect the solution at the following steps, a ghost initial instant is set at $t_{\text{init}} - \Delta t/2$. Consequently, the initial time interval is $[t_{\text{init}} + \Delta t/2, t_{\text{init}} + 3\Delta t/2]$.

In order to preserve boundedness of the spatial discretization schemes and to ensure stability of the temporal schemes, the time step is limited through the Courant-Friedrichs-Lewy number (CFL), as

$$\text{CFL} = \frac{\Delta t|\mathbf{u}|}{\Delta x}, \tag{3.16}$$

where Δx is a characteristic length related to the spatial discretization and $|\mathbf{u}|$ is the magnitude of a representative velocity. This number arises from convergence and stability studies of the different schemes used in the temporal discretization of the partial differential equations associated to unsteady fluid flows [29]. In such studies,

it can be shown that for convection problems the condition CFL ≤ 1 must be satisfied by the explicit schemes used in this work [111]. This condition can also be interpreted as that a fluid particle cannot move more than one mesh length in a single time step [48].

In the numerical simulation of multiphase flows using the FVM is commonly accepted that, in order to keep stability in the calculations, the CFL number must be lesser or equal to a predefined CFL number, $\mathrm{CFL}_{\max}$, which itself is, as much, equal to 1. However, there are two different definitions of the CFL number used in the literature which can yield very different values for the same flow conditions. In OpenFOAM, the maximum CFL number is calculated as

$$\mathrm{CFL}_{\max} = \Delta t \max_i \left(\frac{1}{2} \frac{\sum_f |\phi_f|}{V_i} \right), \tag{3.17}$$

where V_i is the volume of cell i. Due to the use of the average volumetric flux (note that in a solenoidal velocity field the face fluxes summation should be zero), the factor $1/2$ is applied. If a maximum $\mathrm{CFL}_{\max}$ is predefined and an adaptive time step is used, the latter is calculated accordingly to Eq. (3.17) and limiting its increase to a 20% of the previous value to avoid unstable oscillations.

On the other hand, instead of calculating a CFL for every cell in the computational domain and then selecting the maximum value among them, another approach commonly used in the literature is to define a CFL number for the whole computational domain whose maximum value is calculated as

$$\mathrm{CFL}_{\max} = \Delta t \max \left[\frac{\max_f |u|}{\min_i (\Delta x)}, \frac{\max_f |v|}{\min_i (\Delta y)}, \frac{\max_f |w|}{\min_i (\Delta z)} \right], \tag{3.18}$$

where u, v, and w are the three components of the velocity vector at face f such that $\mathbf{u}_f = (u, v, w)$, and $\Delta x = x_{\max} - x_{\min}$, $\Delta y = y_{\max} - y_{\min}$, and $\Delta z = z_{\max} - z_{\min}$ are the maximum cell sizes for the x, y, and z directions, respectively, in a Cartesian coordinate system (note that the mesh is not required to be aligned with the reference system). Since such definition looks for the maximum value of the face velocity along each direction, as well as the minimum cell size along each direction, the face for which $|u|$ is maximum might not correspond to the cell for which Δx is minimum and so on.

Both methods might give similar time steps under certain conditions, e.g., for only one non-zero velocity component on a regular hexahedral mesh, but, in general, Eq. (3.17) is more restrictive due to the flux average, yielding smaller time steps than Eq. (3.18). Thus, it becomes crucial to use the same definition when several methods are compared because in general the accuracy of the result depends on the size of the time step used. However, in the literature, the CFL definition used in the numerical simulations is not always provided, making difficult to compare the results obtained by different authors. For example, in the comparison of different VOF algo-

rithms using prescribed velocity tests, Owkes and Desjardins [116] do not provide any information about the CFL number calculation nor mention which CFL definition is used in the simulations. Therefore, when they compare their results with that obtained by Hernández et al. [60], conclusions might be taken carefully. Another example of this can be seen in [71], where the authors provide the CFL number for each test but do not detail its calculation. In this work, the former definition is used for all the simulations considered.

3.1.4 Surface tension

The discretization procedure of the surface tension term is carried out using the CSF. In this approach, the surface tension integrand is approximated as

$$\sigma\kappa\mathbf{n}\delta_s \approx \sigma\kappa\nabla F, \tag{3.19}$$

where the volume fraction gradient is calculated as

$$\nabla F \approx \frac{1}{V_\Omega}\sum_f F_f \mathbf{S}_f, \tag{3.20}$$

and F_f is obtained through a linear interpolation.

The curvature is calculated through

$$\kappa = -\nabla \cdot \mathbf{n}, \tag{3.21}$$

where $\mathbf{n}$ is the unit normal vector to the interface and is calculated as

$$\mathbf{n} = \frac{\nabla F}{|\nabla F|}. \tag{3.22}$$

Substituting Eq. (3.22) into (3.21) and applying the divergence theorem, Eq.(3.21) can be approximated as

$$\kappa \approx -\frac{1}{V_\Omega}\sum_f \mathbf{S}_f \cdot \left[\frac{(\nabla F)_f}{|\nabla F|_f}\right], \tag{3.23}$$

where $(\nabla F)_f$ is the gradient at the face center interpolated from cell centers.

3.1.5 Pressure equation

As the pressure only appears in the momentum equation and under the form of a gradient, the pressure field cannot be obtained explicitly, i.e., Eqs. (2.1) and (2.2) form an implicitly coupled pressure-velocity system. In order to overcome this issue, the discretized equations are solved in a segregated manner using the PISO algorithm proposed by Issa et al. [66]. Detailed descriptions of this and other algorithms to treat the pressure-velocity coupling can be found, among others, in [48, 111, 171]. In

the PISO method, as in other algorithms of the same family, the pressure gradient is treated explicitly and a Poisson equation is derived from the discretized continuity and momentum equations, so that the pressure field obtained satisfies this resulting equation. For a given computational cell Ω, the semi-discretized momentum equation can be written grouping the terms corresponding to the values of the cell Ω, the rest of terms corresponding to the neighbor cells and the source terms, $H(\mathbf{u})$, and the pressure gradient, as

$$a_\Omega \mathbf{u}_\Omega = H(\mathbf{u}) - \nabla p_d, \tag{3.24}$$

where

$$H(\mathbf{u}) = -\sum_N a_N \mathbf{u}_N + S_{\mathbf{u}_\Omega}, \tag{3.25}$$

and $S_{\mathbf{u}_\Omega}$ includes the surface tension and gravity terms from Eq. (3.4).

Then, this equation is used to predict $\mathbf{u}_f$ at each cell face, which is needed in the discretization of the mass conservation equation given by

$$\int_{V_\Omega} \nabla \cdot \mathbf{u}\, dV = \sum_f \int_{S_f} \mathbf{u} \cdot d\mathbf{S} \approx \sum_f \mathbf{S}_f \cdot \mathbf{u}_f = 0. \tag{3.26}$$

Special care must be taken in this step when using a collocated mesh arrangement to avoid the decoupling of velocity and pressure. For this purpose, the interpolation proposed by Rhie and Chow [130] is used, where the gradient of the pressure at face f is calculated explicitly from the pressure values at the shared cells. Thus, the face velocity is calculated as

$$\mathbf{u}_f = \left(\frac{H(\mathbf{u})}{a_\Omega}\right)_f - \left(\frac{1}{a_\Omega}\right)_f (\nabla p_d)_f, \tag{3.27}$$

where the face values, except for the pressure gradient, are calculated from linear interpolations.

Substituting Eq. (3.27) into the discretized mass conservation equation (Eq.(3.26)) yields the pressure equation

$$\sum_f \left[\left(\frac{1}{a_\Omega}\right)_f \mathbf{S}_f \cdot (\nabla p_d)_f\right] = \sum_f \left[\mathbf{S}_f \cdot \left(\frac{H(\mathbf{u})}{a_\Omega}\right)_f\right]. \tag{3.28}$$

The steps performed by the PISO algorithm are:

1. Obtain the initial pressure, p_d^*, and velocity fields, $\mathbf{u}^*$, from the previous time step or from the initial conditions.

2. Obtain, if the momentum predictor is carried out, a new velocity field, $\mathbf{u}^{*,p}$, by solving the momentum equation (Eq. (3.24)) using $\mathbf{u}^*$ and p_d^* along with the source terms, and set $\mathbf{u}^* = \mathbf{u}^{*,p}$. Note that this velocity field does not satisfy the continuity equation.

3. Construct the operator $H(\mathbf{u}^*)$ and obtain the new pressure field p_d^{**} by solving the pressure equation (Eq. (3.28)).

4. Update the volumetric fluxes using the velocities at the faces corrected in Eq. (3.27).

5. Obtain a new velocity field, $\mathbf{u}^{**}$, consistent with the pressure field, by using $H(\mathbf{u}^*)$ and p_d^{**} in Eq. (3.24).

6. Repeat steps 3 to 5 up to a predefined number of iterations (usually 3).

3.2 Volume of fluid methods used in this thesis

In this section, the VOF methods used in this work are briefly described. The algebraic VOF method MULES and the geometric VOF method isoAdvector available in OpenFOAM are used. Also, several reconstruction and advection schemes available as routines in the `gVOF` open-source package [89] have been implemented in OpenFOAM. Detailed descriptions of all these methods considered in this work can be found in [38, 60, 65, 87, 97, 136, 145].

In the VOF method, Eq. (2.5) is integrated over a given cell Ω and time interval Δt using its conservative form, taking into account the incompressible flow assumption and the definition introduced in Eq. (3.5), resulting as

$$F^{t+\Delta t} = F^t - \frac{1}{V_\Omega}\int_t^{t+\Delta t}\int_{V_\Omega} \nabla\cdot(\mathbf{u}\chi)\, dV\, dt, \tag{3.29}$$

where the second term in the right hand side represents the net volume of fluid advected out of cell Ω. However, if the discrete velocity divergence in the cell Ω is not null, the divergence term in the conservative form of Eq. (2.5) should be retained. Depending on the VOF method used, Eq. (3.29) is solved either algebraically or geometrically.

3.2.1 Algebraic method

In this work, the algebraic VOF method MULES available in OpenFOAM is used. This method is based on the flux corrected transport technique developed by Boris and Book [17] and then improved by Zalesak [188]. It introduces a compression term in the conservative form of the advection equation (Eq. (2.5)), which, taking into account the incompressible flow simplification, results in

$$\frac{\partial\chi}{\partial t} + \nabla\cdot(\mathbf{u}\chi) + \nabla\cdot(\mathbf{u}_r\chi(1-\chi)) = 0, \tag{3.30}$$

where $\mathbf{u}_r$ is the compression velocity used to avoid the interface smearing. In this last equation, the non-transient terms are integrated and discretized in the FVM framework as described in Section 3.1 at time t, whereas the transient term is discretized in a

finite-difference manner using a first-orderexplicit Euler scheme, resulting in

$$\frac{F^{t+\Delta t} - F^t}{\Delta t} + \frac{1}{V_\Omega} \sum_f \left[F_f^t \, \mathbf{u}_f^t \cdot \mathbf{S}_f + F_f^t (1 - F_f^t) \mathbf{u}_{r,f}^t \cdot \mathbf{S}_f \right] = 0, \tag{3.31}$$

where $\mathbf{u}_f^t$ is the velocity at the cell face center obtained from the Rhie-Chow interpolation at time t and $\mathbf{u}_{r,f}^t$ is the compression velocity at face f, whose computation is discussed later. The first term inside the sum operator represents the flux of the liquid through face f from time t to $t + \Delta t$ and the second term acts as a fictitious flux. The summation of these two terms is denoted hereafter as Φ_f^t. Thus, Eq. (3.31) can be written as

$$F^{t+\Delta t} = F^t - \frac{\Delta t}{V_\Omega} \sum_f \Phi_f^t, \tag{3.32}$$

where Φ_f is evaluated through

$$\Phi_f = \Phi_{f,UD} + \psi_M \Phi_{f,HR}, \tag{3.33}$$

where ψ_M is the MULES limiter whose value is 1 at the interface and 0 otherwise, $\Phi_{f,UD}$ is the flux calculated with the upwind scheme, and $\Phi_{f,HR}$ is the flux calculated with a high-resolution scheme. These fluxes are obtained as

$$\Phi_{f,UD} = F_{f,UD} \, (\mathbf{u}_f \cdot \mathbf{S}_f), \tag{3.34}$$

$$\Phi_{f,HR} = F_{f,HR} \, (\mathbf{u}_f \cdot \mathbf{S}_f) - \Phi_{f,UD} + F_{r,f}(1 - F_{r,f})(\mathbf{u}_{r,f} \cdot \mathbf{S}_f), \tag{3.35}$$

where $F_{f,HR}$ is obtained using a high-resolution scheme and $F_{r,f}$ is obtained using a special scheme implemented in OpenFOAM. The term involving the compression velocity, $(\mathbf{u}_{r,f} \cdot \mathbf{S}_f)$, is computed as

$$(\mathbf{u}_{r,f} \cdot \mathbf{S}_f) = C_F \left| \mathbf{u}_f \cdot \mathbf{S}_f \right| \frac{(\nabla F)_f}{\left| (\nabla F)_f \right|}, \tag{3.36}$$

where C_F is a user-specified parameter. If $C_F = 1$, the compression is conservative, if $C_F = 0$, no interface compression is applied, and if $C_F > 1$, the interface is sharpened, although the calculation becomes unstable.

Recovering the term $F_{r,f}$ of Eq. (3.35), the ad-hoc scheme for its calculation is called interface compression scheme. If the face f at which this value is calculated is defined as the face shared between cell Ω and a downwind neighbor cell N, in the interface compression scheme, $F_{r,f}$ is defined using an expression similar to Eq. (3.8) as

$$F_{r,f} = F_\Omega + \frac{|\mathbf{x}_f - \mathbf{x}_\Omega|}{|\mathbf{x}_N - \mathbf{x}_\Omega|} (F_N - F_\Omega) \psi_r, \tag{3.37}$$

with a limiter function ψ_r

$$\psi_r = \min\{\max[1 - \max(\alpha_\Omega, \alpha_N), 0], 1\}, \tag{3.38}$$

being

$$\alpha_\Omega = \{1 - [4F_\Omega(1 - F_\Omega)]\}^2, \;\; \alpha_N = \{1 - [4F_N(1 - F_N)]\}^2. \tag{3.39}$$

Since the compressive term in Eq. (3.30) is non-linear, the system of equations obtained after the discretization procedure cannot be solved using the methods outlined in Section 3.5.1. Thus, the MULES algorithm first solves the conservative advection equation without the compressive term to obtain an initial guess for the volume fraction. Then, the compression term is evaluated explicitly and used to update the obtained guess. This procedure is repeated iteratively up to a predefined number of iterations.

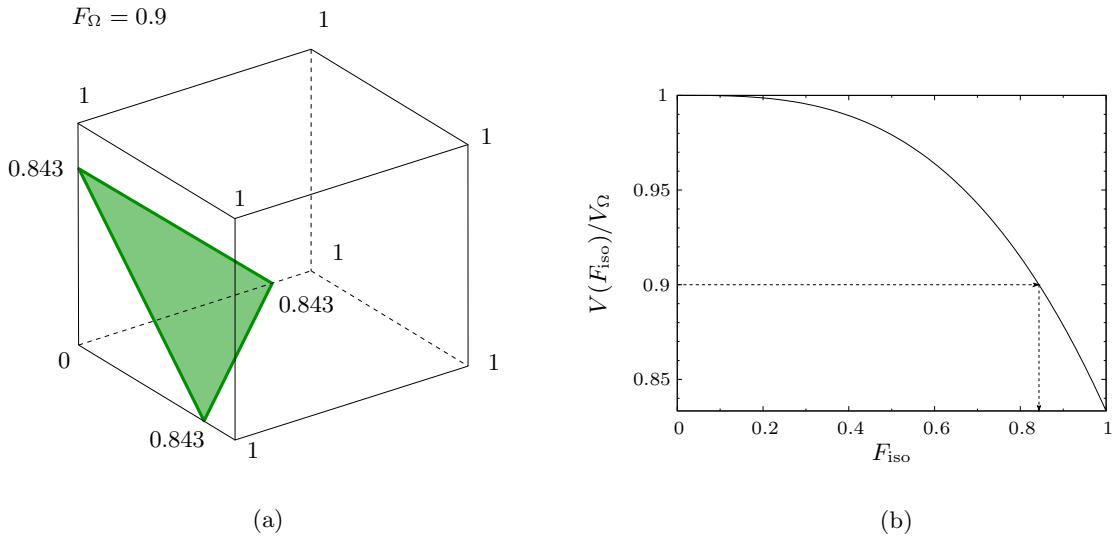
3.2.2 Geometric methods

isoAdvector

This method, as previously mentioned in Section 1.4, was originally designed to reconstruct the interface through an isosurface that cuts the polyhedral cell into a region with a given volume such that volume conservation is maintained. The problem to solve is somehow similar to the VCE problem described in Section 1.4: an isovalue for the volume fraction, F_{iso}, must be found such that the liquid volume enclosed by the isosurface $V(F_{\text{iso}})$, relative to the cell volume, V_Ω, equals the cell volume fraction F_Ω. For each cell with $F_{\text{tol}} < F_\Omega < 1 - F_{\text{tol}}$, where F_{tol} is the prescribed volume fraction tolerance, the problem to be solved can be written as

$$F_\Omega - \frac{V(F_{\text{iso}})}{V_\Omega} = 0. \tag{3.40}$$

An analytic expression for $V(F_{\text{iso}})/V_\Omega$ can be obtained since it varies monotonically like a piecewise cubic polynomial from 0 to 1 as F_{iso} varies from $\max_k(F_k)$ to $\min_k(F_k)$, being F_k the volume fraction at node k previously interpolated from the mesh cell centers of the cells surrounding cell vertex k. In the interval $F_{k,1} \leq F_{\text{iso}} \leq F_{k,2}$, where $F_{k,1}$ and $F_{k,2}$ are the closest F_k values to F_{iso}, the four coefficients of this expression are obtained by geometrically evaluating $V(F_{\text{iso}})/V_\Omega$ for four different F_{iso} values and then solving the resulting Vandermonde matrix system of equations using a LU decomposition. With the analytic expression at hand, F_{iso} is computed such that Eq. (3.40) is satisfied up to a prescribed tolerance by using the Newton's root finding method. An example of this procedure can be observed in Fig. 3.1. The volume fraction in the cell is $F_\Omega = 0.9$ with the volume fraction values at the nodes shown in Fig. 3.1(a). The coefficients for obtaining the cubic expression plotted in Fig. 3.1(b) are obtained by



(a) (b)

Figure 3.1: Example of the procedure to obtain F_{iso} in isoAdvector. (a) Cell with $F_\Omega = 0.9$ the volume fractions values at the cell nodes and the isosurface obtained after solving Eq. (3.40). (b) Plot of the analytic cubic expression for F_{iso}.

geometrically evaluating $V(F_{\text{iso}})/V_\Omega$ for $0 \leq F_{\text{iso}} \leq 1$ at four different points. Then, the cubic equation is solved, yielding $F_{\text{iso}} = 0.843$.

Once the isovalue is calculated, the center of the interface, $\mathbf{x}_{\text{int}}$, and the unit vector normal to the interface pointing to the liquid, $\mathbf{n}$, can then be obtained. The center of the isovertices is calculated through the average

$$\mathbf{x}_{c,\text{iv}} = \frac{1}{N_{\text{iv}}} \sum_k \mathbf{x}_k, \tag{3.41}$$

where N_{iv} is the number of isovertices. The vector normal to the interface is computed following

$$\mathbf{n}^* = \sum_t \mathbf{n}_{t,k}, \tag{3.42}$$

where

$$\mathbf{n}_{t,k} = \frac{1}{2}(\mathbf{x}_{k+1} - \mathbf{x}_k) \times (\mathbf{x}_{c,\text{iv}} - \mathbf{x}_k), \tag{3.43}$$

assuming that the isovertices are ordered such that $\mathbf{n}^*$ points into the liquid (note that the isovertices ordering is not a trivial task, but further details can be consulted in the open-source code available in [115]). The unit vector normal to the interface is calculated as

$$\mathbf{n} = \frac{\mathbf{n}^*}{|\mathbf{n}^*|}, \tag{3.44}$$

and the interface center as

$$\mathbf{x}_{\text{int}} = \sum_k \frac{|\mathbf{n}_{t,k}|}{|\mathbf{n}^*|} \frac{\mathbf{x}_k + \mathbf{x}_{k+1} + \mathbf{x}_{c,\text{iv}}}{3}. \tag{3.45}$$

To improve the accuracy of the method, especially for unstructured meshes, Scheufler and Roenby [145] recently implemented several interface reconstruction methods in the isoAdvector framework making use of a reconstructed distance function. The most accurate method in any mesh type is called as plicRDF. Instead of an isosurface, a PLIC is reconstructed on each interfacial cell (a cell that contains the interface), similarly to other geometric reconstruction methods. In this method, the VCE problem is solved using the procedure described to solve Eq. (3.40), where instead of the isovalue F_{iso}, the position of the PLIC (the interface center $\mathbf{x}_{\text{int}}$) is computed using the normal of the previous time step linearly interpolated to the corresponding cells through a weighted average as initial estimation, which improves the convergence of the method. Once the PLIC is positioned, it is used to construct the reconstructed distance function by computing the minimum distance from the centers of the cells surrounding the PLIC to the plane containing it. In the next step, the gradient of this function is computed using a least-squares method to obtain the unit vector normal to the interface $\mathbf{n}$. This whole procedure is repeated iteratively while the convergence criterion is not satisfied up to a maximum number of iterations, which is set to 5 but, in general, does not exceed of 3.

Once the interface has been reconstructed on each cell containing it at time t either using an isosurface or a PLIC, it is advected from time t to time $t + \Delta t$ by solving Eq. (3.29) through the following assumptions: the velocity in that time interval is considered to be constant and the velocity at any cell face can be obtained through the volumetric flux at the face. Thus, the volume of liquid advected through the downwind face f, V_f, is calculated as

$$V_f \approx \frac{\phi_f^t}{|\mathbf{S}_f|} \int_t^{t+\Delta t} A_f \, dt, \tag{3.46}$$

where ϕ_f^t is the volumetric flux through downwind face f at time t and A_f is the area described by the movement of the intersection between the interface and the cell face within the time interval.

In order to estimate the interface motion during a time step and then calculate the time integral of the submerged area, the velocity at the interface center, $\mathbf{u}_{\text{int}}$, is obtained by linearly interpolating the velocity field to the interface center and the interface normal velocity is then calculated as $u_{\text{int}} = \mathbf{u}_{\text{int}} \cdot (-\mathbf{n})$. The evolution of the intersection between the interface and face f (see Fig. 3.2) is estimated as a piecewise linear function by computing the time at which the interface reaches each vertex as

$$t_k \approx t^t + (\mathbf{x}_k - \mathbf{x}_{\text{int}}) \cdot \frac{(-\mathbf{n})}{u_{\text{int}}}, \tag{3.47}$$

where $\mathbf{x}_k$ is the position vector of downwind face vertex k. The position of the line segment at a given time within the sub-interval $[t_k, t_{k+1}]$ can be obtained using a

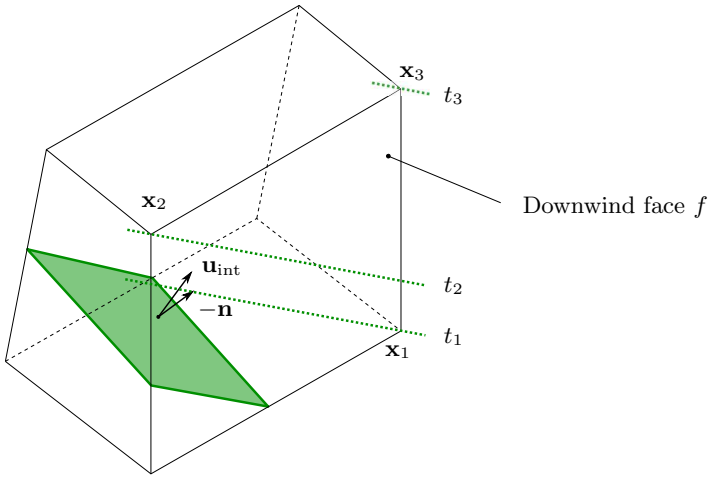


Figure 3.2: Evolution of the intersection between the interface and a downwind face f in a polyhedral cell. Green dotted lines represent the intersection line for each time it passes a face vertex. Figure adapted from [136].

linear interpolation and a quadratic polynomial expression for the submerged face area A_f is constructed within that sub-interval. This expression is integrated over the subinterval, obtaining a cubic polynomial function of time. Adding up all the contributions from these sub-intervals within a time step, the volume of liquid advected through a downwind face f in Eq. (3.46) can be obtained. For all the other faces, V_f is calculated as the volume fraction in their upwind cell multiplied by the time step and the volumetric flux at the face. The new volume fraction in Eq. (3.29), $F^{t+\Delta t}$, is finally computed.

This method does not ensure strict boundedness of the volume fraction values. Therefore, a bounding procedure is used to redistribute the unboundedness fluid values using the face fluxes as weighting factors in the calculation of the amount of redistributed fluid that crosses to the downwind neighbor cells.

`gVOF` package

In the `gVOF` open-source package, which uses as external libraries the `VOFTools` v5 [95, 100] and `isoap` [87, 90], several efficient and accurate routines for volume of fluid initialization, interface reconstruction, fluid advection, interface visualization and reconstruction errors computation on arbitrary meshes are included. Among the interface reconstruction and fluid advection routines, the `llcir`, `elcir` and `clcir` routines, which implement improved versions of, respectively, LLCIR, ELCIR and CLCIR isosurface-based interface reconstruction methods proposed by López et al. [97], the `lsgir` routine, which implements the least-squares gradient interface reconstruction (LSGIR) method (see the work by Rider and Kothe [132] and the references therein), and the `faceflux` and `vofadv` routines, which implement multidimensional unsplit advection schemes based on different procedures to construct the flux polyhedra, are

Algorithm 1 Extended CLCIR method [87]

Obtain φ at the mesh vertices from Eq. (3.48)
for every mesh cell **do**
    Maximum ($\varphi_{\max}$) and minimum ($\varphi_{\min}$) values of φ at the cell vertices
    if $\varphi_{\max} > 0.5$ and $\varphi_{\min} < 0.5$ **then**
        Extract the isosurface corresponding to $\widetilde{\varphi} = 0.5$ by using the procedure in [87]
        if there is only one extracted isopolygon **then**
            Mark the mesh cell as valid isosurface cell
            Construct the 'local triangulated surface'
            Compute $\mathbf{n}^l$ from Eq. (3.49)
        end if
    end if
end for
for every interfacial cell **do**
    if it is a valid isosurface cell **then**
        Construct the 'extended triangulated surface' $\mathcal{T}$
        Compute $\mathbf{n}^e$ from the equation equivalent to Eq. (3.49)
        if $\arccos(\mathbf{n}^e \cdot \mathbf{n}^l) < 1.2$ rad **then**
            $\mathbf{n} = \mathbf{n}^e$
        else
            $\mathbf{n} = \mathbf{n}^l$
        end if
    else
        Compute $\mathbf{n}$ from the LSGIR method
    end if
    Compute C to locate the PLIC
end for
for every interfacial cell marked as valid isosurface cell **do**
    Update the vertices of $\mathcal{T}$ with the corresponding PLIC centers
end for
for every interfacial cell marked as valid isosurface cell **do**
    Compute $\mathbf{n}^c$ from the equation equivalent to Eq. (3.49) using the updated $\mathcal{T}$
    if $\arccos(\mathbf{n}^c \cdot \mathbf{n}) < 1.2$ rad **then**
        Update $\mathbf{n}$ with $\mathbf{n}^c$
        Compute C to relocate the PLIC
    end if
end for

used in the OpenFOAM code to obtain some of the results presented in the next chapters. In this thesis, only two of the flux polyhedra construction procedures included in [89] are considered: an extension to 3D arbitrary meshes of the EMFPA method proposed by López et al. [91], and a new version of the FMFPA method proposed by Hernández et al. [60]. In the following, a description of the CLCIR method extended to arbitrary meshes by López et al. [87], which has been coupled into the OpenFOAM code with the isoAdvector advection step, is described. Also, a brief overview of the advection methods EMFPA and FMFPA is provided.

As stated in Section 1.4, the interface is represented by a plane given by Eq. (1.18), where the unit-length vector $\mathbf{n}$, normal to the interface and pointing to the fluid, is determined from the PLIC reconstruction method presented in the Algorithm 1 (see Reference [87]) described below. The PLIC position is defined by the constant C, which is computed solving the VCE problem so that the interface splits cell Ω, of

volume V_Ω, into two sub-cells of volumes FV_Ω and $(1-F)V_\Omega$. The CIBRAVE (coupled interpolation-bracketed analytical volume enforcement) method of López et al. [93] is used to compute C, except when using meshes with cubic cells, for which the efficient analytical method of Scardovelli and Zaleski [143] is used. The implementation of these two volume conservation enforcement methods is included in the `VOFTools` package [94, 99, 100].

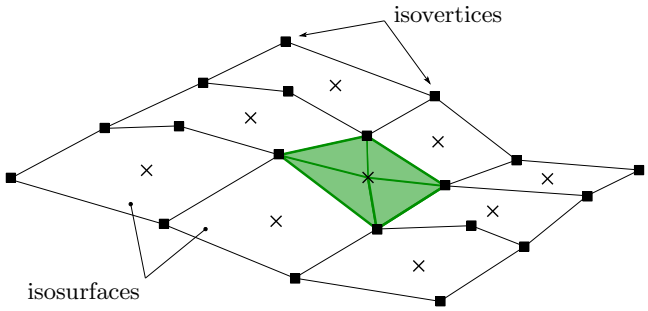
The scalar field φ at each instant t and cell vertex k of the computational domain is obtained from

$$\varphi_k = \frac{\sum\limits_l F_l w_l}{\sum\limits_l w_l}, \tag{3.48}$$

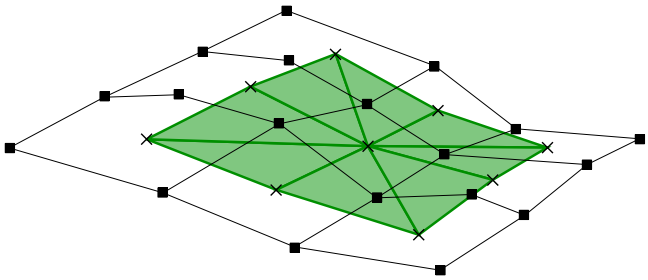
where the summations extend to all cells l containing the vertex k and $w_l = 1/|\mathbf{x}_k - \mathbf{x}_l|$, where $\mathbf{x}_k$ and $\mathbf{x}_l$ are the position vectors of the k vertex and geometric center of cell l, respectively (line 1 in Algorithm 1). For mesh cells whose maximum and minimum interpolated φ values satisfy the condition (line 4) $\varphi_{\min} < 0.5 < \varphi_{\max}$, the isosurface corresponding to $\widetilde{\varphi} = 0.5$ is extracted by using the procedure presented in [87] (line 5). When the extracted isosurface consists of a single isopolygon (line 6), the grid cell is marked as a valid isosurface cell (line 7). A 'local triangulated surface' is then constructed around the geometric center of the extracted isosurface in such a way that each triangle is formed by this geometric center and two consecutive isovertices (line 8). The unit vector normal to the PLIC interface is obtained as

$$\mathbf{n}^l = \frac{\sum\limits_t w_t \mathbf{n}_t / \sum\limits_t w_t}{\left| \sum\limits_t w_t \mathbf{n}_t / \sum\limits_t w_t \right|}, \tag{3.49}$$

where the summation extends over the N_t facets of the triangulated surface, $\mathbf{n}_t$ is the unit-length vector normal to the triangular facet t and w_t is a weighting factor (line 9). For cubic meshes, w_t is defined as the ratio of the sine of the angle between the two inner edges of each triangular facet t and the product of their lengths [108] (an inner edge joins a isovertex with the geometric center of the extracted isosurface), and for the rest of meshes, w_t is defined as the modified angle between triangle edges [165]. To increase the accuracy of the PLIC reconstruction, the initial triangulated surface is substituted by an 'extended triangulated surface' (see the sketch in Fig. 3.3) obtained by connecting its geometric center to those of the isosurfaces extracted at adjacent cells (line 15) and the corresponding vector $\mathbf{n}^e$ is obtained from the equation equivalent to Eq. (3.49) (line 16). If $\arccos(\mathbf{n}^e \cdot \mathbf{n}^l) < 1.2$ rad, $\mathbf{n} = \mathbf{n}^e$ and otherwise $\mathbf{n} = \mathbf{n}^l$ (line 17-21). For interfacial cells that are not valid isosurface cells, situations that frequently occur in regions of low grid resolution, $\mathbf{n}$ is computed using the LSGIR method. Once $\mathbf{n}$ is obtained, the constant C is computed to locate the PLIC (line 25). Finally, the vertices of the extended triangulated surfaces are moved to the corresponding PLIC



(a) Local triangulated surface



(b) Extended triangulated surface

Figure 3.3: Local and extended triangulated surfaces obtained from the extracted isosurfaces.

centers (lines 27-29). For every interfacial cell which is considered as a valid isosurface cell, $\mathbf{n}^c$ is computed from the equation equivalent to Eq. (3.49) using the updated extended triangulated surface (line 31). If $\arccos(\mathbf{n}^c \cdot \mathbf{n}) < 1.2$ rad, $\mathbf{n}$ is updated with $\mathbf{n}^c$ and the constant C is again computed to relocate the PLIC (lines 32 to 35).

Regarding the advection step, the EMFPA and FMFPA methods are based on the construction of a flux polyhedron on each cell face f to determine the volume of liquid that crosses that face during the time interval from time t to $t + \Delta t$, V_f. Thus, the volume fraction at time $t + \Delta t$ is calculated from Eq. (3.29) as

$$F^{t+\Delta t} = F^t - \frac{\sum_f V_f}{V_\Omega}. \tag{3.50}$$

The flux polyhedron constructed at a face f is determined from the volume that crosses that face from t to $t+\Delta t$ given by the volumetric flux ϕ_f and the time step Δt, and, in the EMFPA method, the velocity vectors at the face cell vertices, or, in the FMFPA method, the velocity vectors interpolated at the cell face edges centers. In the latter approach, the over/underlapping between flux polyhedra may still occur when two faces only share a vertex. Once that the flux polyhedra are constructed, they are truncated with the reconstructed interface to obtain the volume of liquid advected V_f through each cell face.

3.2.3 Implementation of the gVOF package in OpenFOAM

As mentioned in the previous section, in this work, some of the reconstruction and advection methods available in the gVOF package have been implemented into OpenFOAM, which use as external libraries the VOFTools v5 and isoap. Since these libraries along with the advection and reconstruction methods are written in FORTRAN language, they are compiled together as a single shared library in order to use it with OpenFOAM, which is written in C++. In addition, the OpenMP application programming interface is used to parallelize this library and improve the computational efficiency in shared-memory architectures. The communication between the gVOF shared library and OpenFOAM solvers is carried out using wrapper functions and several loops inside the original OpenFOAM code to update variables such as the volume fraction, among others.

The interFoam solver in OpenFOAM, suitable for unsteady two-phase flow calculations, has been modified in order to substitute the original MULES scheme by the advection and reconstruction routines available in the gVOF package. Firstly, the information of the OpenFOAM mesh is transferred to the gVOF library using several arrays (details of the mesh format are given in Section 3.5.2). Then, the execution of the VOF method is carried out in several steps: the mesh is tagged using the volume fraction distribution of the previous time step in order to identify the cells and faces where the reconstruction and advection operations will be performed; then, the interface is reconstructed using the PLIC method selected; the velocity is interpolated to the faces and nodes of the mesh; the interface is advected through the calculation of the volume of liquid that crosses each face using one of the advection methods; and, finally, the new volume fraction distribution is calculated.

In addition, with the new volume fraction distribution, the mass flux on each face, $\dot{m}_f$, must be updated as

$$\dot{m}_f = \frac{V_f}{\Delta t} \left(\rho_{l,f} - \rho_{g,f}\right) + \phi_f \, \rho_{g,f}, \tag{3.51}$$

where $\rho_{l,f}$ and $\rho_{g,f}$ are the liquid and gas densities linearly interpolated to the face center, respectively. This expression arises from the fact that the mass flux for a given face f is equal to the volumetric flux multiplied by the density ρ. If the definition given by Eq. (3.6) is introduced, the mass flux can be written as

$$\dot{m}_f = \rho \, \phi_f = \left[\rho_{l,f} \, F_f + \rho_{g,f} \left(1 - F_f\right)\right] \, \phi_f = F_f \, \phi_f \left(\rho_{l,f} - \rho_{g,f}\right) + \phi_f \, \rho_{g,f}, \tag{3.52}$$

where the identification of the liquid volumetric flux is straightforward taking a look at Eqs. (3.51) and (3.52), $F_f \, \phi_f = V_f / \Delta t$.

Alternative implementation in C++

In the implementation of the gVOF pakage into the OpenFOAM code described above, the subroutines available in the FORTRAN libraries are used within the C++ code through wrapper functions. Thus, the arrays containing the mesh information, as well as the normals and constant of the PLICs, among other quantities, must be initialized in the C++ code. The allocation in memory of these arrays is carried out at the moment of compilation, providing a macro with the estimated number of faces, points and neighbors per cell for a given mesh. This implies that the FORTRAN libraries must be recompiled every time the macro values are changed, although, from the user perspective, this can be easily done if, before the execution of the solver, a program which obtains the maximum number of faces, points and neighbors per cell and writes them as macros, is executed and then the FORTRAN libraries are compiled. This procedure would save memory compared to the use of a roughly estimation of these quantities.

However, this approach still poses a processor's memory size issue since the mesh information is being duplicated, and, therefore, it requires more computational resources than if the gVOF arrays were not used. This can be illustrated with a simple example: for a 3D polyhedral mesh of size $n = 160$ (see Section 4.1 for mesh details), assuming that data types int and double occupy 4 and 8 bytes in the memory, respectively, the amount of allocated memory required to store only the mesh data to work with gVOF is 63.3 GB. Besides, as this information is being duplicated, the real memory required is closer to the double of this quantity, 126.6 GB, since OpenFOAM already allocates this information. Therefore, is clear that using gVOF inside OpenFOAM demands more computational resources, which are not always available.

To overcome this issue, in the present work, an alternative solution is proposed. It consists on the translation from FORTRAN to C++ of those subroutines that operate over the entire mesh to avoid the duplication of the mesh information, keeping the other subroutines that perform operations over faces or polyhedra in FORTRAN language, which is the case of the VOFTools and isoap libraries, as well as the routines that construct the flux polyhedra at the mesh faces. Then, a shared library is built to be used into OpenFOAM, as an extension of the VOF libraries already available. A main class gvof is created, in which the operations over the entire mesh such as the mesh tagging, the interface reconstruction or the interface advection are considered as member functions. Also, a class named geoTools is used for the geometric operations in arbitrary polyhedra using the corresponding FORTRAN libraries. It is in this class where the communication between different languages takes place and only a few arrays containing the polyhedra information, i.e., vertex arrangement, face normals, etc., are copied using for loops. These arrays must be also allocated at the time of compilation but its size is chosen to be sufficiently big to store any arbitrary polyhedron, e.g., a

value of 200 for the number of faces and of 240 for the number of nodes, thus not requiring any further recompilation of the code.

An object of class `geoTools` is instantiated in the `gvof` class to perform the geometric operations over a mesh cell: isosurface extraction, cell truncation by a plane, solution of the VCE problem and volume fraction initialization. Another class named `fluxPolyhedron` is created to construct a flux polyhedron for a given face using either the EMFPA or the FMFPA algorithms and perform the required geometric operations using the `geoTools` class, i.e., polyhedron truncation and volume calculation.

The main advantages of the proposed solution are that it is user-friendly since do not require more code compilation than that provided in the libraries, it avoids the mesh information duplicity, it permits to update the `isoap` and `VOFTools` libraries easily (note that, at the moment, these libraries are maintained in `FORTRAN` language), and it allows the extension of the code to work in parallel mode using the MPI interface, although this requires some modifications of the original code. This last aspect is very important when computing simulations that solve the Navier-Stokes equations, since, the time spent per time-step on solving these equations typically corresponds to an 80-90 % of the total time step calculation time, i.e., the solution of the advection equation for the volume fraction is much less time consuming. Therefore, as OpenFOAM is mainly MPI parallelized, it becomes crucial to use this feature when using medium to high mesh resolutions when a static over mesh is used, so that the calculation time is reduced several orders of magnitude, making the simulations more feasible. As the original `FORTRAN` code is parallelized with the OpenMP application, changes in the translated code need to be done in order to make the code able to work with MPI. At the moment of writing this thesis, only the reconstruction methods have been parallelized with MPI. The main issue is that, in the reconstruction methods, for a given interfacial cell, the extension of the isosurface as well as the least squares gradient technique for the normal computation need information from the isosurface neighbors and cell-node neighbors, respectively, and these neighbors may not lie in the same memory portion assigned to the processor as the considered cell. Then, the communication between neighboring processors must be established so that the information available in those cell neighbors can be transferred to the current processor. In order to do that, several operations need to be carry out over the faces that lie in the processors and whose owner cells contain the interface. However, it should be noted that if both the extension of the isosurface and the cell-node neighbor stencil are not carried out at the processors boundary cells, the previous implementation of the reconstruction methods could, in principle, work in MPI but with a possible reduction in accuracy, especially due to avoiding the extension of the isosurface to calculate the interface normal at processor boundary cells.

The accuracy of this alternative implementation has been found to be the same as in the original code, only with minor discrepancies beyond the 16th significant figure,

probably due to round-off errors, which can be obviously neglected. Therefore, the differences between implementations arise in the efficiency of the code, as shown in a section below.

3.3 Contact line force model

A contact line force model (CLFM) used to reproduce more accurately the dynamics of the contact line on solid walls is presented. The force defined by the CLFM acts on the whole contact line, tending to make the contact angle equal to that given by a DCA as a function of the contact line speed. Computationally, a force parallel to the wall and normal to the contact line is applied at cells in the vicinity of the contact line cells, i.e., cells that satisfy the following conditions: at least one of their faces is contained in the wall plane; their nodes satisfy $\max_k(F_k) > 0.5$ and $\min_k(F_k) < 0.5$, where k is the node index; and the extracted 0.5-isosurface ($\widetilde{\varphi} = 0.5$ in the isosurface extraction problem posed in Section 1.4) intersects two wall face edges. The first condition states that contact line cells are obviously wall cells; the second allows for a 0.5-isosurface to be extracted at the cell; and the third ensures that a contact line exists, since the other two conditions are necessary but not sufficient (e.g., just before the impact of a drop on a solid surface, the 0.5-isosurface in a wall cell may be parallel to the surface).

The total force applied in the vicinity of each contact line cell i is defined as

$$\mathbf{f}_{cl,i} = \sigma \left(\cos\theta_{d,i} - \cos\theta_i\right) \frac{\lambda_i}{V_i} \mathbf{n}'_{\mathrm{iso},i}, \tag{3.53}$$

where $\theta_{d,i} = \theta_{d,i}(u_{\mathrm{cl},i}, \theta_a, \theta_r, \theta_e)$ is the value provided by a DCA model as a function of the contact line velocity at the contact line cell i, $u_{\mathrm{cl},i}$, and the static contact angles; θ_i the calculated DCA at the contact line cell i; $\mathbf{n}'_{\mathrm{iso},i}$ the projection to the wall of the unit vector normal to the 0.5-isosurface at cell i pointing out of the liquid; and λ_i the length of the contact line at cell i. The DCA model proposed by Kistler [75], briefly described in Section 1.5, is used in the proposed CLFM because it has been found to generally provide the best results for the range of conditions considered in this work.

The contact angle θ_i is calculated through the dot product of the unit vector normal to the interface at the contact line cell i, $\mathbf{n}_{\mathrm{iso},i}$, pointing to liquid side, and the unit vector normal to the wall, $\mathbf{n}_w$,

$$\mathbf{n}_{\mathrm{iso},i} \cdot \mathbf{n}_w = \cos\theta_i. \tag{3.54}$$

The 0.5-isosurface extraction at the contact line cells is carried out using the isoap method developed by López et al. [87], which, making use of its code publicly available at [90], has been implemented into OpenFOAM. Volume fractions are first interpolated from cell centers to mesh vertices using an inverse distance weighting method (similar expression to Eq. (3.48)). Then, the isoap method is applied to obtain an isosurface at

each contact line cell. In this method, the isosurface is approximated locally at each cell by a polygonal surface using a general polygon tracing procedure and the following additional considerations: the isovertices are located at the cell edges connecting cell vertices with assigned F_k values above and below 0.5; at most, one isovertex is inserted at each cell edge; and the isovertices inserted at cell edges are sequentially joined by line segments, forming polygons that may be non-planar. The isovertices ordering is the most complex step of the algorithm, and further details can be consulted in [87].

At a cell where only a single isosurface is extracted, this isosurface is triangulated in such a way that each triangle is formed by joining two consecutive isovertices with the center of the isovertices $\mathbf{x}_{c,\mathrm{iv}}$, computed using Eq. (3.41). The centroid of each triangle is computed and the isosurface center $\mathbf{x}_{\mathrm{iso}}$ is calculated as a weighted average of the triangle centroids using the areas of the triangles as weights. Then, the unit vector normal to the isosurface pointing to the liquid side, $\mathbf{n}_{\mathrm{iso},i}$, is obtained using Eq. (3.49) by making $\mathbf{n}^l = \mathbf{n}_{\mathrm{iso},i}$.

The length of the contact line at each contact line cell, λ_i, can be calculated by simply computing the distance between the isovertices lying at the wall, i.e. the length of the intersection line between the 0.5-isosurface and the wall face. In the rare situation in which more than a single isosurface is extracted in a contact line cell, the unit normal vector is calculated from the volume fraction gradient and the length of the contact line is set equal to the cell size.

It is clear that the contact line force distribution is discontinuous, since it is only concentrated at the contact line. However, it has been found that the concentration of this force on contact line cells may promote an unrealistic break-up of the lamella when the force value is sufficiently high. In order to avoid this non-physical effect, the contact line force is convolved using a kernel K to smooth the discontinuity. The kernel chosen here is the following monotonic eighth-degree polynomial function proposed by Williams et al. [175],

$$K(\mathbf{x}, r_k) = \begin{cases} C_k(r_k)[1 - (|\mathbf{x}|/r_k)^2]^4 & \text{if } |\mathbf{x}| \leq r_k, \\ 0 & \text{if } |\mathbf{x}| > r_k, \end{cases} \tag{3.55}$$

where $\mathbf{x}$ is the vector joining the two points considered in the convolution, r_k the kernel radius, and $C_k(r_k)$ a constant that normalizes the kernel for each contact line cell i as follows:

$$C_k(r_k) \sum_j [1 - (|\mathbf{x}_{\mathrm{iso},i} - \mathbf{x}_j|/r_k)^2]^4 = 1, \tag{3.56}$$

where $\mathbf{x}_j$ is the vector joining the centre of the isosurface at cell i and the centroid of cell j. Typically, the value of the kernel radius is set so that, at least, all the point neighbors at the wall of the contact line cells lie inside the kernel action; e.g., for a uniform cubic mesh of cell size Δx, the kernel radius is chosen as $2\Delta x \leq r_k \leq 3\Delta x$.

The contact line velocity $u_{\mathrm{cl},i}$ involved in the calculation of the $\theta_{d,i}$ value used in Eq. (3.53) is obtained as follows. In every contact line cell i, $u_{\mathrm{cl},i}$ is assumed to be equal to the fluid velocity vector interpolated to the 0.5-isosurface center and projected to the wall, $\mathbf{u}_{\mathrm{iso},i}$, and then dotted with the projection of the interface normal vector pointing out of the liquid, $\mathbf{n}'_{\mathrm{iso},i}$. It can be written as

$$u_{\mathrm{cl},i} = \left[\mathbf{u}_{\mathrm{iso},i} - \left(\mathbf{n}_w \cdot \mathbf{u}_{\mathrm{iso},i}\right)\mathbf{n}_w\right] \cdot \mathbf{n}'_{\mathrm{iso},i}. \tag{3.57}$$

With this definition, when $u_{\mathrm{cl},i}$ is positive, the contact line is considered to advance and, when negative, to recede. In cells that do not contain a contact line but where the angle is required, $\mathbf{u}_{\mathrm{iso},i}$ is substituted by the velocity at the cell center, $\mathbf{u}_i$, and $\mathbf{n}'_{\mathrm{iso},i}$ by the unit vector calculated from the volume fraction gradient pointing out of the liquid and projected to the wall.

Note that the force defined in Eq. (3.53) involves a DCA prescribed by a model such as those described in the next section, instead of the static or equilibrium angles, as in the models proposed by Malgarinos et al. [102] and Boelens and de Pablo [15], respectively. Thus, in the proposed model the contact line force only acts when the contact angle differs, not from the corresponding static value, but from the value prescribed by the contact angle model as a function of the contact line velocity, which has been found to improve the numerical predictions for the droplet impact outcome, as shown below.

3.4 Solution procedure

In the previous sections, it has been described the procedure used to discretize the governing equations with the FVM, as well as the VOF methods used to track the interface. The description of the algorithm selected to deal with the pressure-velocity coupling has been also addressed. The numerical procedure used in this work for the solution of the two-phase flow problem is the following:

1. Initialization of all the variables: velocity, pressure and volume fraction distributions must be initialized in the discretized physical domain, considering also the boundary conditions. Care must be taken in the volume fraction initialization procedure, since a poor estimation of this distribution can yield to inaccurate solutions.

2. CFL number and time step calculation: the CFL number is computed for each cell to find the $\mathrm{CFL}_{\mathrm{max}}$ through the entire domain using Eq. (3.17). Then, the time step is calculated according to the CFL restriction previously chosen.

3. Calculate the new volume fraction distribution: solve the discretized advection equation either algebraically (MULES) or geometrically (isoAdvector or gVOF) using the old time, or initial, volume fraction and velocity distributions.

4. Update the fluids properties distributions and the mass flux: according to the new volume fraction distribution, update the density and viscosity associated to each cell (Eq. (3.6)), as well as the mass flux at the faces (Eq. (3.52)). Update the boundary conditions if required (see Section 3.5.3) and calculate surface tension and contact line forces.

5. Obtain the pressure and velocity distributions: using the values updated in the previous step, calculate the velocity and the pressure fields, and the volumetric fluxes, using the PISO algorithm. Note that every time that the pressure or the velocity fields are required, a system of algebraic equations must be solved (see Section 3.5.1 for more details on this procedure).

6. If the final time is reached, end the procedure, if not, go back to step 2.

3.5 Other computational details

3.5.1 Solution of the system of algebraic equations

Once the discretization of the equations has been carried out and the proper boundary conditions over the boundary faces have been applied, a system of algebraic equations of the form $\mathbf{A}\boldsymbol{\phi} = \mathbf{b}$ is generated, where $\boldsymbol{\phi}$ is the vector of unknowns located at the centroids of the mesh elements, $\mathbf{A}$ is a sparse matrix containing the coefficients as the result of the discretization procedure and the mesh geometry, and vector $\mathbf{b}$ contains all sources, constants, boundary conditions, and non-linearizable terms. Due to the non-linear nature of the fluid-mechanics problems, the coefficients resulting from their linearization process are generally solution dependent. Therefore, direct methods for the solution of the system of equations are not suitable for this purpose. Instead, iterative methods are needed, which require less computational resources as will be discussed next.

In a direct method, matrix $\mathbf{A}$ is inverted to obtain the vector solution $\boldsymbol{\phi}$ in one step as $\boldsymbol{\phi} = \mathbf{A}^{-1}\mathbf{b}$. When this matrix is large, as is usual in CFD applications, these methods are very expensive computationally, since they require a large amount of arithmetic operations. Some of the most relevant methods are [111, 171]: Gauss elimination, which comprises the forward elimination and backward substitution steps; LU decomposition, which decomposes the matrix $\mathbf{A}$ into two matrices of upper and lower coefficients such as $\mathbf{LU} = \mathbf{A}$ and then solves two systems of equations using forward and backward substitutions; the tridiagonal matrix algorithm (TDMA), which can only be used if an structured mesh has been chosen for the discretization but is much more efficient than the two previous methods.

On the other side, iterative methods compute a series of solutions from an initial guess that, under certain conditions, such as that matrix $\mathbf{A}$ must be diagonally dominant, converge to the exact solution. The simplest point-iterative method is the

Jacobi method, which from the initial guess solves explicitly the system to obtain a new estimate that is used as initial guess for the next iteration until some convergence criterion is satisfied. An improved point-iterative method with better convergence characteristics is the Gauss-Seidel method, that uses the information from the current iteration as initial guess for computing subsequent values of $\boldsymbol{\phi}$ for that iteration. However, despite its simplicity, these methods have a low rate of convergence [111]. Therefore, some other techniques for convergence improvement were developed. One of the most employed is the preconditioning procedure, in which a preconditioning matrix $\mathbf{P}$ is defined such that the system $\mathbf{P}^{-1}\mathbf{A}\boldsymbol{\phi} = \mathbf{P}^{-1}\mathbf{b}$ has the same solution as the original one, but the spectral properties, directly related to the rate of convergence of the method, of the coefficient matrix $\mathbf{P}^{-1}\mathbf{A}$ are improved. Another technique is the incomplete LU decomposition (ILU), in which now matrices $\mathbf{L}$ and $\mathbf{U}$ have the same nonzero structure than lower and upper parts of $\mathbf{A}$ yielding $\mathbf{A} = \mathbf{LU} + \mathbf{R}$, where $\mathbf{R}$ is the residual of the factorization procedure and is used in the iterative process in order to obtain the solution. Many variants of the ILU factorization exist, the simplest is based on taking the pattern of zero elements in the combined $\mathbf{L}$ and $\mathbf{U}$ matrices to be exactly the pattern of zero elements in the original matrix $\mathbf{A}$, namely the ILU(0) method. If the matrix to be decomposed is positive definite, the method is known as incomplete Cholesky decomposition, and the factorization is made only for the lower or upper parts. Combining preconditioning and ILU factorization yields a class of efficient preconditioners. One of the most employed is the diagonal ILU (DILU), in which only the the diagonal elements are modified, and, thus, only one extra diagonal of storage is required.

Another set of iterative procedures are the gradient methods, initially developed for situations where the coefficient matrix is symmetric positive definite to reformulate the problem as a minimization of a quadratic function, which leads to the solution of the system of equations [111]. In the steepest descent method, the solution of the minimization problem is to be found at the minimum of a paraboloid function, following the fastest rate of descent, i.e., in the negative direction of the function gradient. Despite this method ensures convergence, its rate is low due to oscillations around local minima forcing the method to search in the same direction repeatedly. This behaviour can be avoided in the conjugate gradient method (CG) if every new search is in a different direction from the directions of previous searches, which must satisfy certain conditions. Furthermore, the rate of convergence can be increased using preconditioning (PCG). If matrix $\mathbf{A}$ is not symmetric, a transformation to a symmetrical one is needed if CG is going to be used. When this method is applied to the transformed system of equations, two sequences of CG-like vectors are obtained, what is called bi-conjugate gradient method (BiCG). Preconditioning might also be used in this method (PBiCG).

Finally, it is known that the convergence rate of iterative methods reduces as the mesh size increases since these methods cause rapid reduction in errors components with short wavelengths but does not in long-wavelength components, which tend to decay very slowly as the iteration count increases [171]. In order overcome such weakness, multigrid methods have been developed, in which long-wavelength error components are transformed to short wavelength components decreasing the mesh size. This approach might involve the use of mesh geometric information or just a direct agglomeration of the finer mesh elements, as in the algebraic multigrid method (AMG).

3.5.2 Computational mesh

In OpenFOAM, the mesh is treated as if it was of polyhedral unstructured type, i.e., no previous assumptions about the cells geometry are made. The code reads several text files in which the coordinates of the points and the connectivity of the mesh is provided. Each of the nodes, faces, and cells have a global index assigned to identify them. The file `faces` contains the ordered list of faces, in which each face entry provides the number of nodes and the list of sequently ordered nodes of the face. The file `points` contains the ordered list of vectors with the x, y, and z coordinates of all mesh nodes. The file `owner` contains the ordered list of indices of owner cells such that each face entry provides the cell that owns that face. The file `neighbour` contains the ordered list of indices of neighbor cells for all internal faces, i.e., the index of the cell that shares the internal face with its corresponding owner cell (note that an internal face is shared only by two cells). The file `boundary` contains a set of entries specifying the boundaries of the mesh along with the number of faces and the index of the first face in the face list for each boundary.

3.5.3 Boundary conditions

Due to that the mesh is treated without any geometry assumption, non-orthogonality, as already introduced in Section 3.1.2, must be accounted for. On a boundary cell Ω, vector $\mathbf{d}_{\Omega b}$ joins the cell centroid with the boundary face b centroid. Using the boundary face area vector, $\mathbf{S}_b$, the vector joining the cell centroid and the boundary face can be defined as perpendicular to the boundary face following

$$\mathbf{d}_b = \frac{\mathbf{S}_b}{|\mathbf{S}_b|}\frac{\mathbf{d}_{\Omega b}\cdot\mathbf{S}_b}{|\mathbf{S}_b|}. \tag{3.58}$$

Two different types of boundary conditions are usually employed:

- Dirichlet: in which the value of the variable at the boundary face is specified. For the convection term, this type of boundary condition consists on setting $\mathbf{u}_f = \mathbf{u}_b$ in Eq. (3.7). For the diffusion term, which reads

$$\mu_b \mathbf{S}_b \cdot (\nabla \mathbf{u})_b, \tag{3.59}$$

the product $\mathbf{S}_b \cdot (\nabla \mathbf{u})_b$ is calculated from the known face value $\mathbf{u}_b$ and the cell centroid value $\mathbf{u}_\Omega$ as

$$\mathbf{S}_b \cdot (\nabla \mathbf{u})_b = |\mathbf{S}_b| \frac{\mathbf{u}_b - \mathbf{u}_\Omega}{|\mathbf{d}_b|}. \tag{3.60}$$

For the pressure, the Dirichlet boundary condition consists just on setting the value of the pressure at the boundary face, p_b, and then relate it to the pressure gradient used in the pressure Eq. (3.28), similarly to the diffusion term, as

$$\mathbf{S}_b \cdot (\nabla p)_b = |\mathbf{S}_b| \frac{p_b - p_\Omega}{|\mathbf{d}_b|}. \tag{3.61}$$

- Neumann: in which the value of the dot product of the gradient and the unit face normal is prescribed at the boundary. For the convection term, $\mathbf{u}_b$ is calculated form the value at the cell centroid and the prescribed gradient as

$$\mathbf{u}_b = \mathbf{u}_\Omega + |\mathbf{d}_b| \, g_{b,\mathbf{u}}, \tag{3.62}$$

where $g_{b,\mathbf{u}} = \nabla \mathbf{u} \cdot \mathbf{S}_b / |\mathbf{S}_b|$. The diffusion term is simply

$$\mu_b \, |\mathbf{S}_b| \, g_{b,\mathbf{u}}. \tag{3.63}$$

Similarly to the convection term, if the normal component of the pressure gradient $g_{b,p} = \nabla p \cdot \mathbf{S}_b / |\mathbf{S}_b|$ is set, the pressure at the boundary face is calculated as

$$p_b = p_\Omega + |\mathbf{d}_b| \, g_{b,p}, \tag{3.64}$$

and the gradient g_b is used in the pressure equation.

The combination of these boundary conditions gives rise to some other physical boundary conditions: inlet, where the velocity field is prescribed and the pressure gradient is set to zero; outlet, where the pressure field is prescribed and the velocity gradient is zero; no-slip wall, in which the velocity is set equal to the wall velocity and the pressure gradient is zero since there is no flux through the solid wall; symmetry, in which the component of the gradient normal to the symmetry plane of the dependent variables is zero (except for the velocity since its normal component is positive at one side of the symmetry plane and negative at the other) and the components parallel to it are projected from the inside of the domain, i.e., from the centroids of the cells containing the boundary.

In two phase flows, boundary conditions for the volume fraction must also be considered. In the CSF framework, the interface curvature is specified at the solid surface as a boundary condition. The orientation of the interface normal vector is prescribed using the contact angle, so that the interface at the wall adopts the prescribed angle. The unit vector normal to the interface given by the volume fraction gradient, $\mathbf{n}_0$, is

corrected through

$$\mathbf{n} = a\,\mathbf{n}_0 + b\,\mathbf{n}_w, \tag{3.65}$$

where a and b are scalar coefficients given by

$$a = \frac{\cos\theta^* - \cos\theta\cos(\theta^* - \theta_0)}{1 - \cos^2\theta_0},\ b = \frac{\cos(\theta^* - \theta_0) - \cos\theta_0\cos\theta^*}{1 - \cos^2\theta_0}, \tag{3.66}$$

where θ_0 is the uncorrected contact angle given by $\mathbf{n}_0 \cdot \mathbf{n}_w = \cos\theta_0$ and θ^* is the prescribed contact angle, which satisfies $\mathbf{n} \cdot \mathbf{n}_w = \cos\theta^*$.

The prescribed contact angle, θ^*, is used to calculate the curvature at the contact line in Eq. (3.21). This prescribed angle is sometimes taken to be constant, as in static contact angle models ($\theta^* = \theta_s$), or it can be assumed to be a function of certain flow parameters, such as the contact line velocity ($\theta^* = \theta_d(u_{\mathrm{cl}})$). The latter formulation is adopted by the DCA models described in Section 1.3, which, in this work, have been implemented in OpenFOAM. In these implementations, the contact line velocity is calculated as the component parallel to the wall of the fluid velocity normal to the interface.

In addition, to get the value of the volume fraction at the boundary face that corresponds with that given by the prescribed contact angle, the component of the volume fraction gradient in the direction perpendicular to the wall is calculated as

$$(\nabla F)_w \cdot \mathbf{n}_w = |\nabla F|_w \cos\theta^*, \tag{3.67}$$

where $(\nabla F)_w$ is the volume fraction gradient at the wall face.

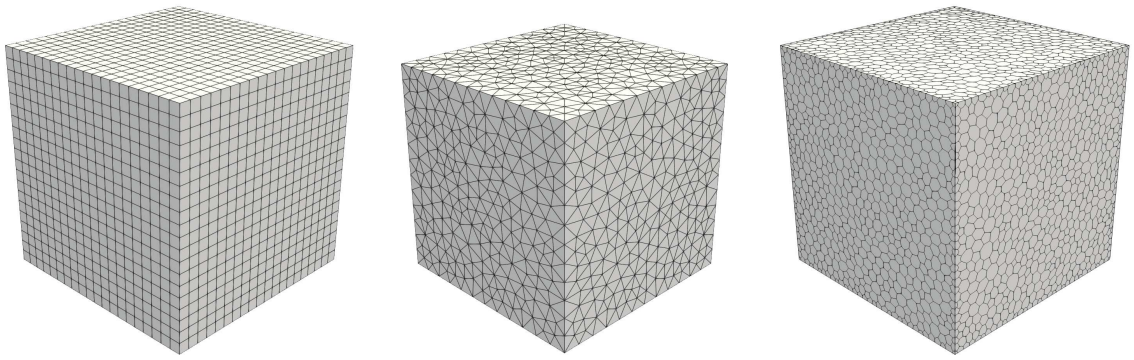
Chapter 4

Accuracy and efficiency of the geometric VOF methods

In this chapter, the geometric VOF methods described in Section 3.2.2 are compared in a consistent and systematic study through the simulation of reconstruction and advection tests with prescribed and non-prescribed velocity fields. Also, the efficiency of the proposed alternative implementation of the CLCIR method, described in Section 3.2.3, is tested. Results obtained when this implementation is coupled with the advection step of isoAdvector are also provided. The algebraic method MULES is left out of this comparison since it has been already compared to isoAdvector and other algebraic methods by Roenby et al. [136].

4.1 Computational details

In order to test the performance of the methods on different meshes, three different types are considered. Figure 4.1 shows examples for these types in a unit-cubic domain. Hexahedral meshes (Fig. 4.1(a)) consist of a set of rectangular parallelepipedic cells, arranged in a structured way. For its construction, the `blockMesh` tool is used, in which the user specifies the coordinates of the points that enclose the physical domain to be discretized and the number of cells on each spatial direction. This tool also allows to define multiple regions with different geometries and mesh refinements. However, in these tests, only structured hexahedral meshes are constructed. Unstructured tetrahedral meshes (Fig. 4.1(b)) are formed by a set of tetrahedral cells of different sizes. These meshes are obtained using the program `tetGen` v1.5 [150]. The generated files with this program with extensions `.ele`, `.node` and `.face` are converted to OpenFOAM's format using the `tetGenToFoam` tool. Unstructured polyhedral meshes of convex polyhedral cells (Fig. 4.1(c)) are constructed from a tetrahedral mesh obtained with `tetGen` and converted to OpenFOAM's format with the `tetGenToFoam` tool. Then, using the `polyDualMesh` utility, the cells centers surrounding a tetrahedral cell vertex are connected to construct a polyhedral cell. It should be noted that



(a) Hexahedral (b) Tetrahedral (c) Unstructured polyhedral

Figure 4.1: Examples of the different 3D mesh types with around 20^3 cells in a unit-cubic domain.

the number of points in the tetrahedral root mesh must be equal to the desired number of cells in the unstructured polyhedral mesh. The faces of the cells generated with this method are generally non-planar, thus, every face is triangulated by joining its center with two consecutive vertices using a program developed specifically for this purpose. For example, for an unstructured polyhedral mesh with 64^3 cells, a tetrahedral mesh of 262122 ($\approx 64^3$) points is generated with the command `tetgen -pq1.2a0.00000127 cube.poly` where the file `cube.poly` defines the domain, and the value `0.00000127` sets the maximum volume for the tetrahedra. Then, the tetrahedral mesh is converted to OpenFOAM's format, the unstructured polyhedral mesh is constructed using the command `polyDualMesh 60 -overwrite` and the mesh is triangulated. Note that in Fig. 4.1(c), the faces in the domain boundary are not yet triangulated since the unstructured polyhedral mesh is represented before the triangulation procedure to show more clearly the shape of the faces.

It should be noted that the size of the hexahedral meshes is expressed as the number of cells on each spatial direction, n, and, therefore, the total number of cells can be calculated as n^3. The size for the tetrahedral and unstructured polyhedral meshes used in this thesis is approximately equal to the cubic root of the total number of cells. In Table 4.1, the total number of points, faces and cells of the meshes are presented. Also, the average number of faces, points and cell-node neighbors per cell is provided, as well as the average number of faces, points and cell-node neighbors per internal cell (a cell whose faces are all internal). This last distinction between cells allows to see more clearly the real amount of neighbors, faces and points per cell. It is worth mentioning that the number of faces and points per cell remains independent of the mesh size for hexahedral and tetrahedral meshes, whilst in unstructured polyhedral meshes changes not monotonically for the internal cells and increases with mesh size for the overall average. The number of cell-node neighbors for the internal cells is independent of n in hexahedral meshes and also almost independent in unstructured polyhedral meshes.

Table 4.1: Average number of points, faces and cell-node neighbors per cell ($\overline{n}$) and internal cell ($\overline{n}_{in}$) for the different 3D meshes and sizes used in the tests. Also, the total number of points (n_p), faces (n_f) and cells (n_c) are provided.

n	$\overline{n}_{p,in}$	$\overline{n}_p$	$\overline{n}_{f,in}$	$\overline{n}_f$	$\overline{n}_{n,in}$	$\overline{n}_n$	n_p	n_f	n_c
					Hexahedral meshes				
10	8	8	6	6	26	21	1 331	330	1 000
20	-	-	-	-	-	23.4	9 261	25 200	8 000
32	-	-	-	-	-	24.3	35 937	101 376	32 768
40	-	-	-	-	-	24.7	68 921	196 800	64 000
64	-	-	-	-	-	25.2	274 625	798 720	262 144
80	-	-	-	-	-	25.3	531 441	1 555 200	512 000
128	-	-	-	-	-	25.6	2 146 689	6 340 608	2 097 152
160	-	-	-	-	-	25.7	4 173 281	12 364 800	4 096 000
256	-	-	-	-	-	25.8	16 974 593	50 528 256	16 777 216
					Tetrahedral meshes				
10	4	4	4	4	67.8	53.1	320	2271	999
20	-	-	-	-	73.5	63.4	1 982	17 229	7 981
32	-	-	-	-	74.8	68.7	6 766	68 636	32 790
40	-	-	-	-	74.7	69.4	12 979	133 069	63 965
64	-	-	-	-	76.5	73.2	47 374	536 692	262 139
80	-	-	-	-	75.1	72.5	93 742	1 056 634	518 135
128	-	-	-	-	77.3	75.6	352 489	4 245 287	2 097 390
160	-	-	-	-	75.9	74.4	687 720	8 164 336	4 038 731
256	-	-	-	-	77.7	76.8	2 687 095	33 443 678	16 620 595
					Unstructured polyhedral meshes				
10	43.3	33.7	82.9	63.3	15.8	11	13 379	36 054	1 012
20	41.9	36.2	79.9	68.5	15.3	12.8	105 620	290 502	8 077
32	41.4	37.4	78.8	70.7	15.1	13.4	432 805	1 195 863	32 756
40	41.3	38.1	78.5	72.1	15	13.4	853 761	2 634 735	64 076
64	41	38.9	78.1	73.9	15	14.1	3 538 197	9 819 651	262 122
80	41	39.3	77.9	74.5	15	14.3	6 929 638	19 247 250	510 961
128	40.8	39.7	77.7	75.5	14.9	14.5	28 523 863	79 293 605	2 086 918
160	40.8	39.9	77.6	75.8	14.9	14.6	57 097 689	158 781 022	4 164 794

Note that the number of nodes and faces per cell in unstructured polyhedral meshes is high due to that the faces have been triangulated, increasing the total numbers of nodes and faces.

Two-dimensional domains are also considered in the non-prescribed velocity tests using pseudo 2D meshes with a single cell in one spatial direction. For hexahedral meshes, in which the cells are square prisms, the `blockMesh` utility is also used. For 2D tetrahedral meshes, composed by triangular prisms, gmsh v4.4.1 and the `gmshToFoam` tool are used. For 2D unstructured polyhedral meshes, gmsh v4.4.1, and the `polyDualMesh` and `extrudeMesh` tools are used. Since the resulting cells are polygonal prisms, no triangulation is needed. For the 3D non-prescribed velocity tests, statically refined meshes are used, generated with the `snappyHexMesh` tool. These meshes are constructed from a uniform Cartesian root mesh whose cells are successively divided up to a certain level of maximum refinement in regions where required. More specifically, several concentric regions are defined in which the root mesh is refined by up to several levels, as described in detail below, while maintaining

the same highest resolution at the interface. To avoid instabilities due to changes in mesh resolution, a graded octree mesh is used.

Other computational details regarding the compilers and the hardware used for running the tests are provided in the following. The OpenFOAM code, written in `C++`, was compiled using gcc 9.3.0 with the `-o3` optimization flag. The `gVOF` package, written in `FORTRAN` and implemented in OpenFOAM as a shared library (see Section 3.2.3), was first compiled using gfortran 9.3.0 with the `-O3` optimization flag and then as a shared library using gcc 9.3.0 with the `-o3` optimization flag. All reconstruction and advection test cases were run on a 2.50 GHz Intel Xeon W with 96 GB of DDR4. The tests for studying the efficiency of the alternative implementation were run on a 2.50 GHz Intel Xeon Gold with 256 GB of DDR4.

4.2 Volume fraction initialization

The volume fraction initialization is the process involved in the computation of the volume fraction of the initial liquid volume contained in each computational cell. Therefore, this operation computes the correspondent distribution of volume fractions for a certain initial geometry shape on a mesh. Results obtained in multiphase-flow numerical simulations are very sensitive to the accuracy of the volume fraction initialization since they are directly related to the surface tension contribution, as shown in Eqs. (3.19) and (3.23), or in the fluid properties in Eq. (3.6). As stated in [95], the lack of accuracy of this initialization procedure can give rise to numerical instabilities in the simulations. Thus, an accurate method for the volume initialization is required.

At the time of writing this work, there are two utilities available in OpenFOAM for this purpose: `setFields` and `setAlphaField`. The former initializes cell volume fractions only with full or empty values, i.e., 1 or 0 respectively, which clearly does not represent well the initial interface, especially for non-planar shapes. The latter utility uses a widely employed method based on defining implicit functions to describe the shape of the regions to be initialized, which is much more precise than the former method. The implicit function, $f_{\text{imp}}(x, y, z) = 0$, is used to approximate the signed distance from the interface to every mesh node by simply substituting their x, y, and z coordinates in the implicit function (note that computing the true distance for certain shapes is not a trivial task [95]).

Another method used in this work, available in the routine `initfgrid` of the `gVOF` package, is the method developed by López et al. [101] and extended to non-convex cells by López et al. [95]. This accurate procedure is based on a recursive local mesh refinement to compute the liquid volume bounded by a convex or non-convex, polygonal or polyhedral cell and a given implicitly-defined liquid interface. Each interfacial cell is subdivided using a superimposed sub-mesh of size n_{sc}^3 of hexahedral cells of size $\Delta x/n_{sc} \times \Delta y/n_{sc} \times \Delta z/n_{sc}$, where $\Delta x = x_{\max} - x_{\min}$, $\Delta y = y_{\max} - y_{\min}$, and

$\Delta z = z_{\max} - z_{\min}$ are the maximum cell sizes along the coordinate axes. The volume fraction of each interfacial cell is calculated as the ratio between the summation of the liquid volumes of the sub-regions, obtained from the computation of its truncation with the approximated interface, and the interfacial cell volume.

Despite the improvement yielded by the `setAlphaField` method compared to the original `setFields`, it has been found that `setAlphaField` generally gives an initialization error two orders of magnitude larger than the method proposed by López et al. [95] for $n_{sc} = 10$. In order to show the assessment of this two methods, four shapes have been initialized in a domain of size $1 \times 1 \times 1$. These shapes are defined by the following implicit functions:

$$f_{\text{imp}}(x, y, z) = 0.25^2 - \left[(x-0.5)^2 + (y-0.5)^2 + (z-0.5)^2\right], \tag{4.1}$$

$$f_{\text{imp}}(x, y, z) = -0.2^2 + \left[(x-0.5)^2 + (y-0.5)^2 + (z-0.5)^2\right], \tag{4.2}$$

$$f_{\text{imp}}(x, y, z) = 0.1^2 - \left\{0.2 - \left[(x-0.5)^2 + (y-0.5)^2\right]^{0.5}\right\}^2 - (z-0.5)^2, \tag{4.3}$$

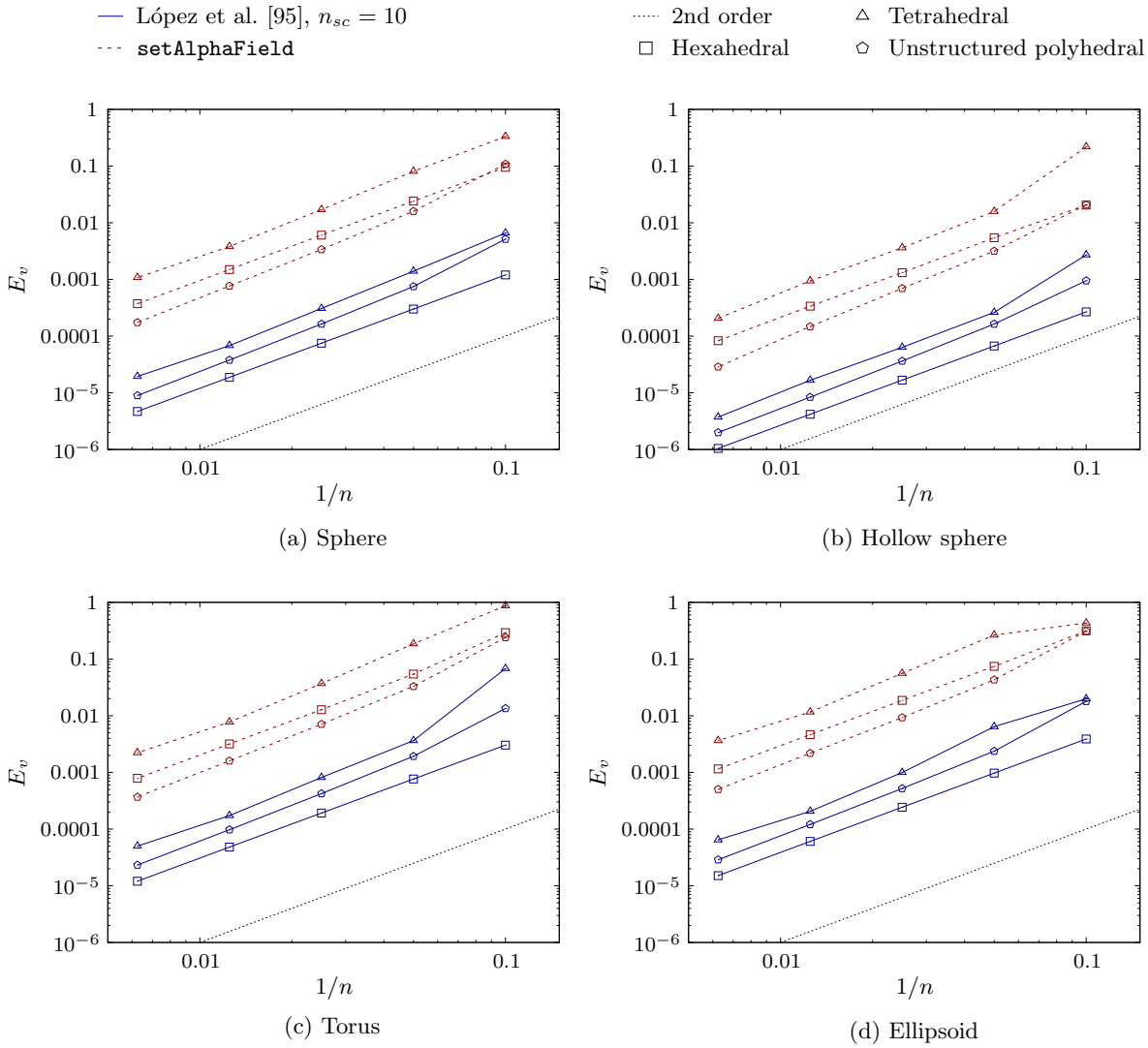
$$f_{\text{imp}}(x, y, z) = 1 - \left[\frac{(x-0.5)^2}{0.3^2} + \frac{(y-0.5)^2}{0.15^2} + \frac{(z-0.5)^2}{0.1^2}\right], \tag{4.4}$$

where Eq. (4.1) defines a sphere of radius 0.25, Eq. (4.2) a sphere of radius 0.2, Eq. (4.3) a torus, and Eq. (4.4) an ellipsoid. The convex sphere of Eq. (4.2) is used to obtain a hollow sphere of radius 0.4 with a spherical core of radius 0.2 computing the minimum distance for each node, i.e., $\min[f_{\text{imp},R=0.4}(\mathbf{x}), f_{\text{imp},R=0.2}(\mathbf{x})]$, where $f_{\text{imp},R=0.4}(x, y, z)$ is equal to Eq. (4.1) but with radius value of 0.4 instead of 0.25, and $f_{\text{imp},R=0.2}(x, y, z)$ to Eq. (4.2).

Figure 4.2 shows the initialization error as a function of the mesh size for different meshes. This error is defined as

$$E_v = \frac{|V_{\text{init}} - V_{\text{exact}}|}{V_{\text{exact}}}, \tag{4.5}$$

where V_{init} is the initialized volume and V_{exact} is the exact volume. It can be seen that both methods yield second-order convergence for all mesh types in all geometric shapes. For hexahedral and tetrahedral meshes, the initialization error obtained with the method of López et al. [95] with $n_{sc} = 10$ is two orders of magnitude lower than that obtained with the `setAlphaField` tool, whilst for unstructured polyhedral meshes this difference is reduced to one order of magnitude. This behavior can be explained as follows. The construction of the isopolygon carried out by the `setAlphaField` tool, which is formed by the cut points at the cell edges with zero signed distance to the interface, is performed using the signed distance value stored at each mesh node. As the total number of nodes in an unstructured polyhedral mesh is higher than in a hexahedral mesh, considering the same number of cells, more information about the spatial distribution of the signed distance function is available, and, also,



(a) Sphere

(b) Hollow sphere

(c) Torus

(d) Ellipsoid

Figure 4.2: Initialization error as a function of the mesh size in the initialization of the volume fraction for different mesh types and interface shapes.

the distance between two consecutive nodes is smaller. Thus, since the cut points are obtained using a linear interpolation between two consecutive vertex, the error introduced by this approximation is reduced. However, in the method of López et al. [95], the superimposed Cartesian mesh applied to hexahedral meshes results in a great reduction of the initialization error, as it could be thought as a local refinement (note that increasing n_{sc} can reduce the initialization error dramatically).

4.3 Reconstruction tests

In order to compare the accuracy of the reconstruction procedures, the reconstruction error, E_{rec}, defined as the volume between the exact interface and its approximate representation

$$E_{\text{rec}} = \sum_i 2 \left| V_i \, F_i - V_{\text{cut},i} \, F_{\text{cut},i} \right| , \tag{4.6}$$

is used, where $V_{\text{cut},i}$ is obtained truncating the cell i with its corresponding reconstructed PLIC, and $F_{\text{cut},i}$ is the volume fraction obtained when the volume fraction initialization procedure by López et al. [95], already described in Section 4.2, is applied to the truncated polyhedron resulting from the cell truncation with the PLIC plane. This error measurement procedure is similar to that described by Aulisa et al. [8] and is available in the routine `recerr` of the `gVOF` package.

For the sake of clarity, the reconstruction methods compared in this section are only CLCIR, ELCIR, gradAlpha and plicRDF, since these are the methods that give rise to the better results in terms of efficiency and accuracy. In the plicRDF method the maximum number of iterations is set equal to 5. With the purpose of comparing the computational efficiency of the methods, the consumed cpu-time in the reconstruction step, t_{rec}, is measured. The volume fraction tolerance, F_{tol}, is the same for all methods, with a value of 10^{-8}.

4.3.1 Sphere

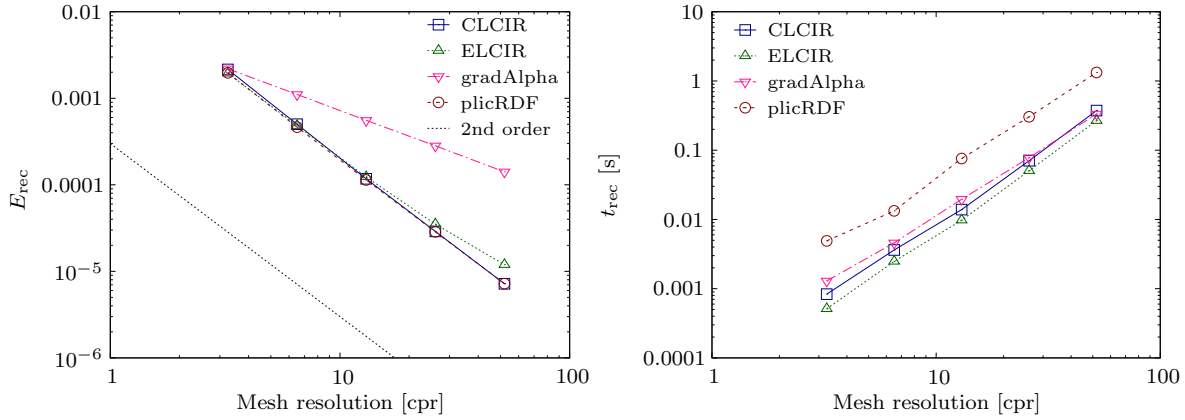
This test consists on reconstructing a sphere of diameter 0.65, which, to avoid artificial regularities in the results due to mesh dependence, is centered at $(0.525, 0.464, 0.516)$ in a unit-cubic domain. Figure 4.3 shows the reconstruction error and the consumed cpu-time for the compared methods as a function of mesh resolution, measured in cells per radius (cpr) of the sphere for hexahedral, tetrahedral and unstructured polyhedral meshes. As depicted in Fig. 4.3(a), for hexahedral meshes, CLCIR and plicRDF methods show the best results in terms of convergence and reconstruction error. For the rest of meshes, the results obtained with these two methods are practically equal for all mesh resolutions. ELCIR shows second-order convergence and reconstruction errors like those of CLCIR and plicRDF for low and medium mesh resolutions, whilst the gradAlpha scheme only yields first-order convergence and shows the largest values for E_{rec}. In terms of computational efficiency, plicRDF consumes the largest cpu-time, an order of magnitude higher than that obtained with the CLCIR method for almost all mesh resolutions. The gradAlpha method consumes a cpu-time like that of the CLCIR method for high mesh resolutions, although at low mesh resolutions it is a 35% larger, and the ELCIR method gives the smallest t_{rec}. For tetrahedral meshes (Fig. 4.3(b)), CLCIR gives slightly smaller reconstruction errors than plicRDF except for the finest mesh size, where plicRDF produces slightly smaller errors. ELCIR and gradAlpha produce very similar errors and convergence orders, smallers than the remaining two methods. In this case, ELCIR also takes the smallest time to reconstruct the interface for all mesh resolutions whilst plicRDF the largest, again an order of magnitude larger than the times consumed by CLCIR. These differences in the reconstruction time between CLCIR and plicRDF are mainly due to that the procedure used by CLCIR to calculate the PLIC position, CIBRAVE, is about 2 to 4 times more efficient than that implemented in the plicRDF method, and, as the latter repeats the reconstruction

procedure several times per reconstruction step, the differences are more evident. Also, since the reconstructed distance function in a cell is obtained as a weighted average of the distances from the cell center to its own interface and the interfaces in its cell-node neighbors, and, as tetrahedral cells have a higher amount of cell-node neighbors compared to an hexahedral cell, the time spent in reconstructing the distance function also increases.

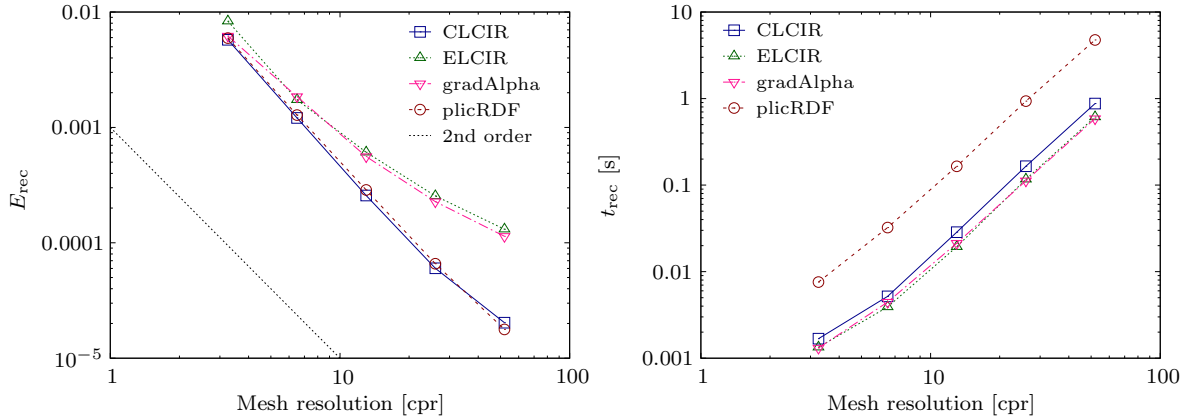
On unstructured polyhedral meshes (Fig. 4.3(c)), CLCIR and plicRDF yield similar errors for low resolutions although for higher mesh resolutions, plicRDF shows reconstruction errors lower than CLCIR, due to the reduction in convergence order of the latter. The gradAlpha scheme is the most efficient method, although it produces the highest errors for medium-high resolutions. CLCIR consumes a slightly lower cpu-time for the finest mesh and, although for coarser meshes CLCIR is faster than plicRDF, the great difference shown in the other mesh types is now reduced to only a 35%. The increment in the cpu-time shown by CLCIR in this kind of meshes is mainly due to the increase consumed by the procedure to construct the local 0.5-isosurface and the PLIC positioning, since polyhedral cells have a significant higher number of points than tetrahedral or hexahedral meshes. In this case, plicRDF spends more time in the calculation of the PLIC position than in reconstructing the distance function since the number of point neighbors is considerably lower than in tetrahedral meshes. For example, for $n = 80$, solving the VCE problem consumes the 88% of the reconstruction time and the RDF construction only a 6.5%. For the same resolution but in tetrahedral meshes, the percentages are 22% and 61%, respectively.

4.3.2 Hollow sphere

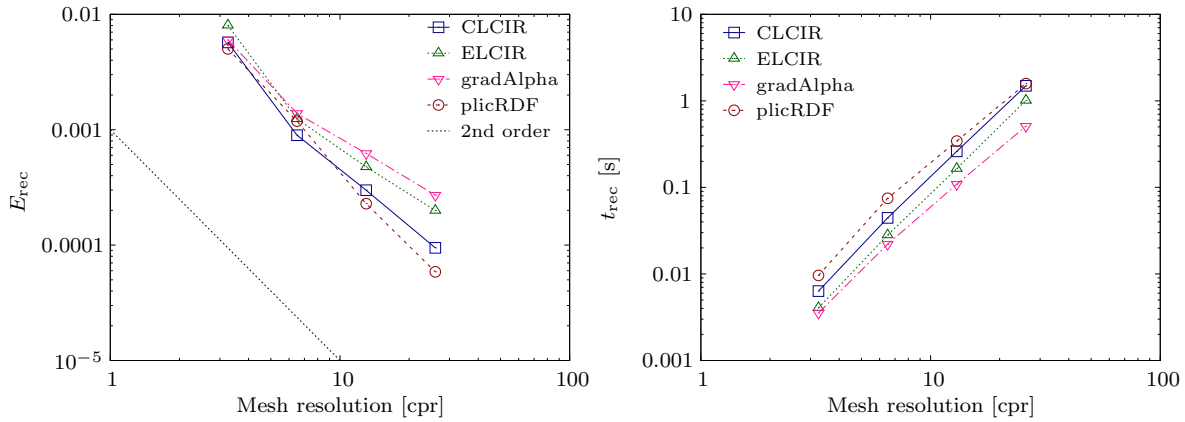
In the test proposed by Liovic et al. [85], a sphere of radius 0.4 with a spherical core of radius 0.2 is centered at (0.525, 0.464, 0.516) in a unit domain. Fig. 4.4 shows the reconstruction errors and consumed cpu-times as a function of mesh resolution for the different mesh types considered. As in the previous test, plicRDF and CLCIR give rise to very similar results in terms of convergence and reconstruction error for hexahedral meshes, although CLCIR yields slightly smaller values in E_{rec}, and, again, gradAlpha shows first-order convergence. For tetrahedral meshes (Fig. 4.4(b)), the results are very similar to those obtained for the sphere reconstruction test when comparing the accuracy. However, in terms of computational efficiency, gradAlpha consumes now cpu-times very similar to those of the CLCIR method, and ELCIR is the fastest method. In this type of meshes, CLCIR is now 6.6-7.6 times faster than plicRDF. Also, for unstructured polyhedral meshes (Fig. 4.4(c)), the results are close to those obtained in the sphere reconstruction test, although it should be noticed how the difference in consumed cpu-times for the plicRDF and CLCIR methods, which is about a 46%, is now maintained for all mesh resolutions, and that the number of iterations required to reach convergence of the plicRDF method in these reconstruction tests is, in general,



(a) Hexahedral meshes



(b) Tetrahedral meshes



(c) Unstructured polyhedral meshes

Figure 4.3: Reconstruction error, E_{rec}, and consumed cpu-time, t_{rec}, for different mesh types and sizes for the sphere reconstruction test.

higher than in a dynamic test, since plicRDF uses the previous normal as initial guess, decreasing the number of iterations required to reach the desired convergence. Therefore, the cpu-times spent in the reconstruction step in such tests, would be lower than that presented in the reconstruction tests.

4.4 Advection tests

In this section, three tests with different prescribed velocity fields are used to compare the accuracy and efficiency of the different methods. In the `gVOF` package, the FMFPA flux polyhedra construction algorithm is selected for the advection step due to a less expensive computation cost compared to the EMFPA method without a significant change in accuracy (for sufficiently low CFL numbers). For the reconstruction step, the CLCIR algorithm is chosen due to the better overall performance, as shown in the previous comparison. This combination of methods is named as FMFPA-CLCIR. For the isoAdvector advection scheme, the plicRDF reconstruction algorithm is selected, since is the most accurate of the methods implemented by Scheufler and Roenby [145], and this combination is named as isoAdvector-plicRDF.

The geometric error is estimated with an L_1 error norm defined as

$$E_g = \sum_i V_i \left| F_i - F_{e,i} \right|, \tag{4.7}$$

where $F_{e,i}$ is the exact volume fraction at the final instant of the test $t = T$ (in this work is obtained with the routine `initfgrid` of the `gVOF` package). The order of convergence can be then determined from

$$\mathcal{O} = \frac{\ln[E_g(2n)/E_g(n)]}{\ln(1/2)}, \tag{4.8}$$

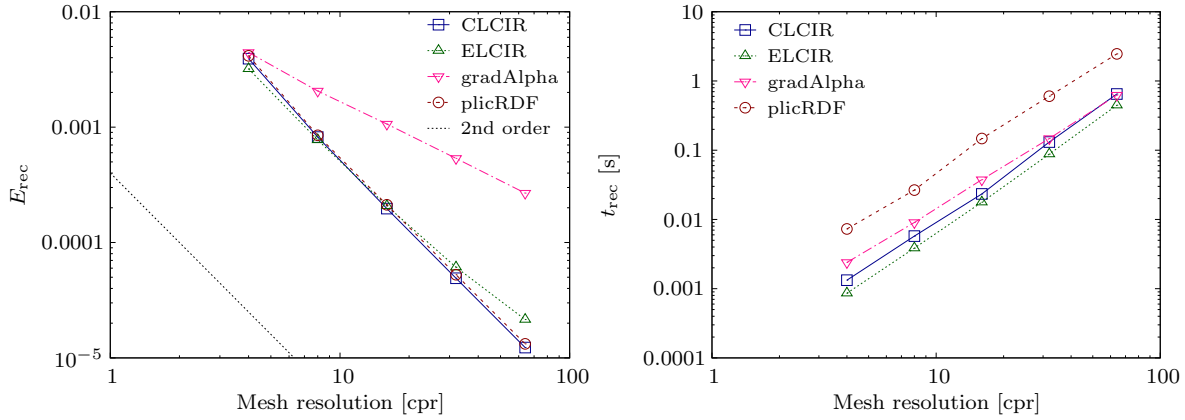
where $E_g(n)$ and $E_g(2n)$ are the errors obtained using two different mesh sizes with n^3 and $(2n)^3$ cells, respectively.

The volume conservation error is quantified as

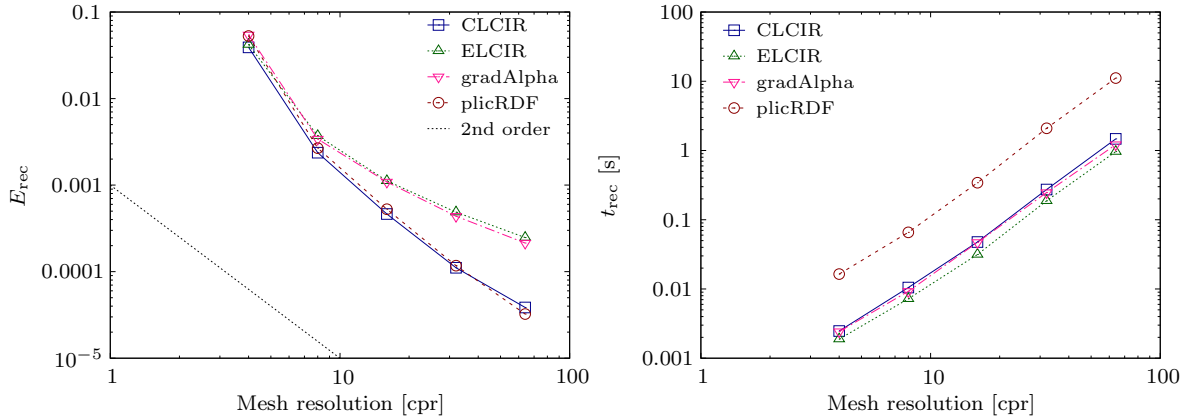
$$E_{\text{vol}} = \left| \sum_i V_i \left(F_i - F_{e,i} \right) \right|, \tag{4.9}$$

where the volume fractions are obtained at the final instant of the test. The boundedness of the solution at instant t is measured through the error

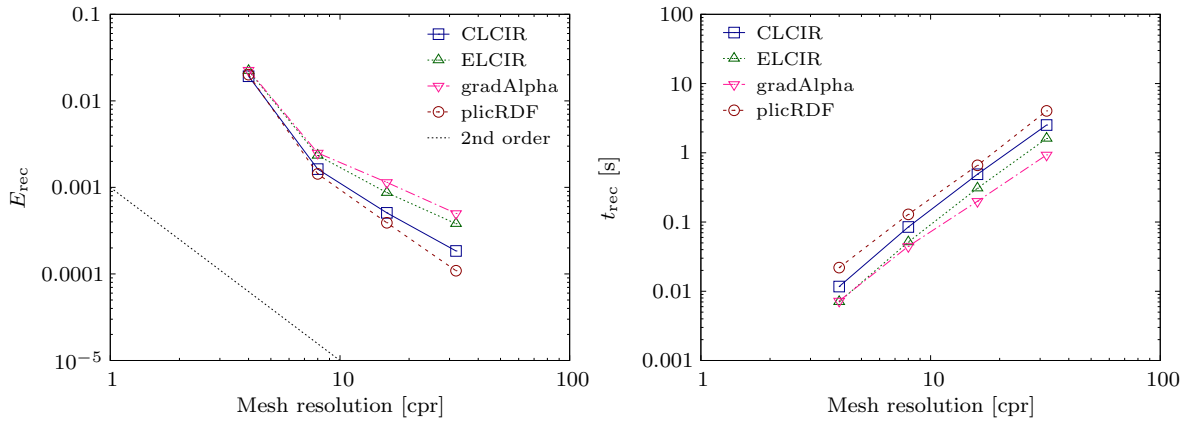
$$E^t_{\text{bound}} = \max \left\{ E^{t-\Delta t}_{\text{bound}}, \max \left[\left| \min_i \left(V_i \, F_i \right) \right|, \max_i \left(V_i \, F_i \right) \right] \right\}, \tag{4.10}$$



(a) Hexahedral meshes



(b) Tetrahedral meshes



(c) Unstructured polyhedral meshes

Figure 4.4: Reconstruction error, E_{rec}, and consumed cpu-time, t_{rec}, for different mesh types and sizes for the hollow sphere reconstruction test.

where $E_{\text{bound}}^{t-\Delta t}$ is the boundedness error at the immediate precedent instant $t - \Delta t$. This definition allows to capture the unboundedness of the solution throughout the complete simulation.

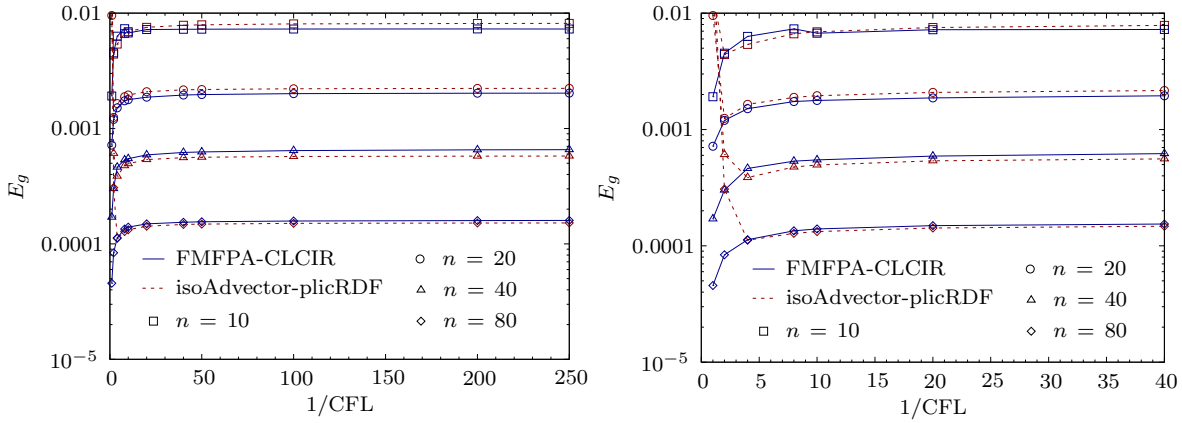
To compare the computational efficiency of the algorithms, the consumed cpu-time, t_{adv}, defined as the average time consumed by the advection procedure, and the consumed cpu-time introduced in Section 4.3, t_{rec}, which now is defined as the average time consumed by the reconstruction procedure, are measured. Also, the total consumed cpu-time per time step, t_{tot}, is defined as the summation of the advection and reconstruction times. These times are provided relative to the corresponding smallest total time value, hence, its relative values $\tilde{t}_{\text{adv}}$, $\tilde{t}_{\text{rec}}$, and $\tilde{t}_{\text{tot}}$ are used instead.

In the isoAdvector method, the parameter `nAlphaBounds` is set equal to 5, which applies 5 times a conservative bounding step per advection-reconstruction step, redistributing the fluid of unbounded cells to preserve volume conservation [136]. Besides, isoAdvector offers the possibility to force the bounding of the volume fraction value, but this procedure is kept unset for the sake of consistency in the comparison. The volume fraction tolerance is the same for both combinations, with a value 10^{-12}.

4.4.1 Simple translation

A sphere of radius 0.5 centered at (0.25, 0.25, 0.25) is translated by means of a steady and uniform flow with velocity components (1, 1, 1) for a period of time $T = 0.5$, in a cube domain of size $1 \times 1 \times 1$. Figure 4.5 shows the geometric error as a function of the CFL number for the two combinations in an hexahedral mesh with different sizes. At high CFL numbers, the error obtained with isoAdvector-plicRDF is larger than that obtained with the FMFPA-CLCIR methods for all mesh sizes, increasing the difference between them as the mesh resolution increases. As the CFL decreases, the error stabilizes around a constant value for each method, reaching the mesh resolution dependent value described by Harvie and Fletcher [58]. For $n = 10$ and $n = 20$ the error of FMPFA-CLCIR at low CFL numbers is a 11-14% lower than that obtained with isoAdvector-plicRDF, whereas for $n = 40$ the error of the latter is a 11% lower than that of the former. For the finest mesh resolution both combinations give a similar geometric error.

It should be noted that the CFL calculation method defined by Eq. (3.17), results in higher errors in FMFPA-CLCIR than that obtained with the CFL calculation method defined by Eq. (3.18). The latter method yields larger time step sizes, and, since the geometric error obtained with FMFPA-CLCIR decreases with increasing time step sizes, these errors are smaller. This trend in the geometric error was explained by Harvie and Fletcher [58]: the authors argued that as the time step size decreases, more steps are required to complete a given duration, therefore, more interface reconstructions steps are carried out (one per time step). Since each reconstruction step introduces a discrete amount of error, if the number of reconstructions is increased so it is the total



(a) (b)

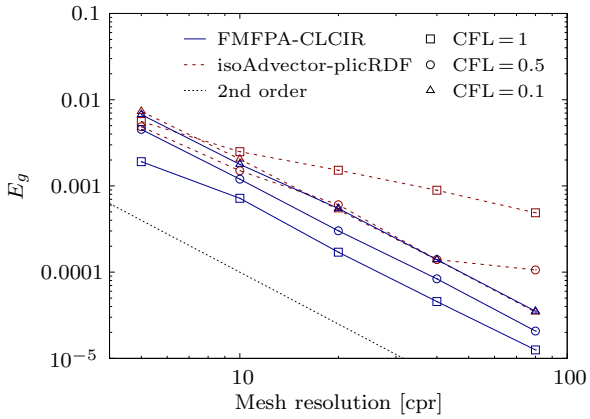
Figure 4.5: L_1 error norm as a function of the CFL number obtained for the translation test using four different hexahedral mesh sizes,

amount of error introduced in the simulation. However, the error obtained with the isoAdvector algorithm does not follow this trend. Instead, it decreases with decreasing time step sizes, which could indicate that the error introduced by the advection step dominates over the reconstruction error for high CFL numbers.

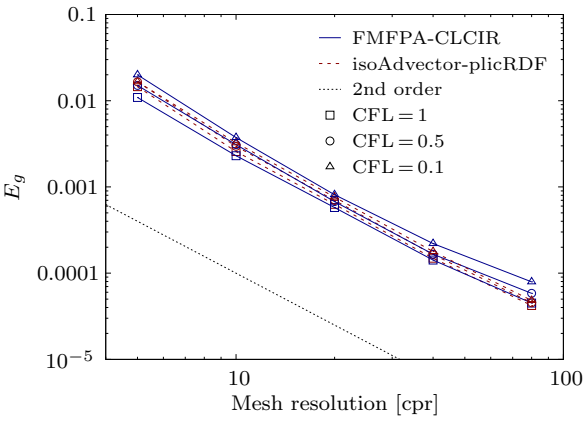
Figure 4.6 shows the L_1 error norm as a function of mesh resolution obtained for different mesh types and three CFL numbers. Fig. 4.6(a) depicts the L_1 error norm for hexahedral meshes. The convergence rate of the isoAdvector-plicRDF methods increases as the CFL decreases achieving second order for CFL around 0.1, whereas the convergence of the FMFPA-CLCIR methods is second order for all the range of CFL numbers considered. These differences on the convergence rates yield to large differences on the error norm for high mesh resolutions and high CFL numbers, being about an order of magnitude. For tetrahedral meshes (Fig. 4.6(b)), the results are less dependent on the CFL number, and both combinations show second-order convergence in all the range of the CFL numbers. However, for low CFL numbers at high mesh resolutions, FMFPA-CLCIR yield higher error norms than that obtained with the other methods. For unstructured polyhedral meshes (Fig. 4.6(c)) both combinations give very similar errors for low CFL numbers and high mesh resolutions, although for the coarsest mesh the FMFPA-CLCIR methods yield smaller errors. For CFL = 1, the isoAdvector-plicRDF methods yield a smaller error for the coarsest mesh, whereas as the mesh resolution increases, FMFPA-CLCIR give lower error norms.

4.4.2 3D deformation

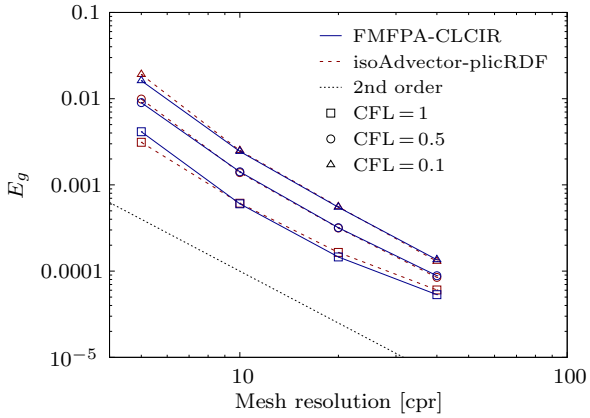
The well-known benchmark test proposed by Enright et al. [45] consists on a sphere of radius 0.15 initially centered at (0.35, 0.35, 0.35) within a unit-cube domain, which



(a) Hexahedral meshes



(b) Tetrahedral meshes



(c) Unstructured polyhedral meshes

Figure 4.6: L_1 error norm as a function of mesh resolution obtained for the translation test using three different mesh types and CFL numbers.

is deformed in a solenoidal velocity field, $\mathbf{u} = (u, v, w)$, given by

$$\begin{aligned} u(x,y,z,t) &= 2\sin^2(\pi x)\sin(2\pi y)\sin(2\pi z)\cos(\pi t/T), \\ v(x,y,z,t) &= -\sin(2\pi x)\sin^2(\pi y)\sin(2\pi z)\cos(\pi t/T), \\ w(x,y,z,t) &= -\sin(2\pi x)\sin(2\pi y)\sin^2(\pi z)\cos(\pi t/T), \end{aligned} \tag{4.11}$$

where a period $T = 3$ is used.

Table 4.2 shows the errors and consumed cpu-times for the two considered combination of methods. The cpu-times are provided relative to the smallest total time value, which in this case is given by $t_{\text{tot}} = 23.9$ ms obtained using the isoAdvector-plicRDF methods for an hexahedral mesh with $n = 32$. For all mesh types and sizes the L_1 error norm obtained with FMFPA-CLCIR is always smaller than that obtained with the isoAdvector-plicRDF methods, except for the coarsest unstructured polyhedral mesh, where isoAdvector-plicRDF yield an error a 3.7% lower. Differences for the rest of mesh types and sizes range from the 4.9% lower error obtained with FMFPA-CLCIR for tetrahedral meshes with $n = 32$, to the 56.3% lower error also obtained with FMFPA-CLCIR for hexahedral meshes with $n = 256$. For hexahedral and tetrahedral meshes, as the mesh size increases the differences in the geometric error also increase, due to that the convergence order is higher for the FMFPA-CLCIR methods, although both combinations reach second-order convergence for the finest resolution. For unstructured polyhedral meshes, differences between the compared methods are reduced and the convergence rate of isoAdvector-plicRDF is slightly higher for $n = 128$. This behavior might be caused by a better performance of the advection step in the latter method, since isoAdvector is based on the advection of an isosurface, and as this is constructed with smaller interpolation errors in unstructured polyhedral cells than in hexahedral or tetrahedral cells (for a given size a polyhedral cell requires a greater amount of nodes, e.g., for $n = 128$ the average number of points per polyhedral cell is 39.7, as shown in Table 4.1), the advection error should be lower as well.

The error in volume conservation for hexahedral meshes is several orders of magnitude lower using the FMFPA-CLCIR methods, except for $n = 128$. For the rest of mesh types FMFPA-CLCIR yield a higher error, which although small ($\lesssim 10^{-6}$), is several orders of magnitude higher than that obtained with isoAdvector-plicRDF ($\lesssim 10^{-12}$). This difference is probably due to the use of the bounding procedure in the isoAdvector advection stepg, which reduces the volume conservation error as well as the geometric error. The boundedness error is very similar for all meshes, although for some resolutions FMFPA-CLCIR yield an error several orders of magnitude lower. In terms of computational efficiency, as the mesh size increases, the differences between both combinations in the average total time, t_{tot}, decrease. For the coarsest meshes, isoAdvector-plicRDF yield the lowest values and for hexahedral meshes, as the mesh resolution increases, the cpu-time consumed by FMFPA-CLCIR is more similar to

that consumed by isoAdvector-plicRDF. However, for all mesh types, the total times of isoAdvector-plicRDF are smaller. In hexahedral meshes, the time consumed by the advection is of the same order for both combinations whereas the time consumed by the reconstruction step is much lower for the CLCIR method, partially due to the iterative nature of the plicRDF algorithm. For the finest tetrahedral and unstructured polyhedral meshes, the reconstruction times consumed by the plicRDF algorithm are, respectively, 7.5 and 3.7 times higher than that consumed by the CLCIR method, whereas the advection times consumed by isoAdvector are 10.5 and 4.9 times lower than that spent by FMFPA.

The great increase in the consumed cpu-time for the FMFPA algorithm in tetrahedral and unstructured polyhedral meshes with respect to hexahedral meshes is mainly due to the face advection flux calculation. In this algorithm, the advection of the fluid is carried out using face flux polyhedra, which are first constructed using the velocities interpolated at the cell nodes and then truncated with the reconstructed interface at the cell and at the neighbor cells surrounding the face. Thus, if the number of neighbor cells that contain the interface increases, the number of truncation operations will also increase. For an internal tetrahedral cell, the average number of cell-node neighbors is between 74 and 78 for the mesh sizes considered in this test (see Table 4.1), whilst for an hexahedral cell, this number is only 26 and not size-dependent. Therefore, as the number of interfacial cells increases with mesh refinement, the number of truncations also increases, and, consequently, the consumed cpu-time rises. On the contrary, as isoAdvector does not perform this kind of operations for the face flux calculation, the time consumed by the advection step is considerably smaller. Note that as the number of faces of a tetrahedral cell is lower than that of an hexahedral cell, the consumed cpu-time is smaller in the latter.

For the unstructured polyhedral meshes considered in this test, the average number of cell-node neighbors per internal cell is 15, whereas the average number of faces and points per internal cell is from 40.8 to 41.4 and from 77.7 to 78.8, respectively, which is a great increase compared to those of hexahedral meshes (6 and 8) or tetrahedral meshes (4 and 4). Then, in a polyhedral cell, the number of truncations for a given face is substantially smaller since the number of neighbors is very small compared to other mesh types, but, as the number of flux polyhedra is highly increased due to the great amount of faces, the total amount of truncation operations is thus increased. This increase in the average number of faces per polyhedral cell also increases the time consumed by the isoAdvector algorithm because it is based on the calculation of the area described by the interface-face intersection movement within the time interval considered.

Figures 4.7 and 4.8 depict the PLIC interfaces obtained with the two compared combinations at the intermediate ($t = 1.5$) and final ($t = 3$) instants of the test where differences on the geometric error can now be clearly noticed. The PLICs are printed in

Table 4.2: Errors and consumed cpu-times for the 3D deformation test using a CFL = 0.5 for different mesh types and resolutions.

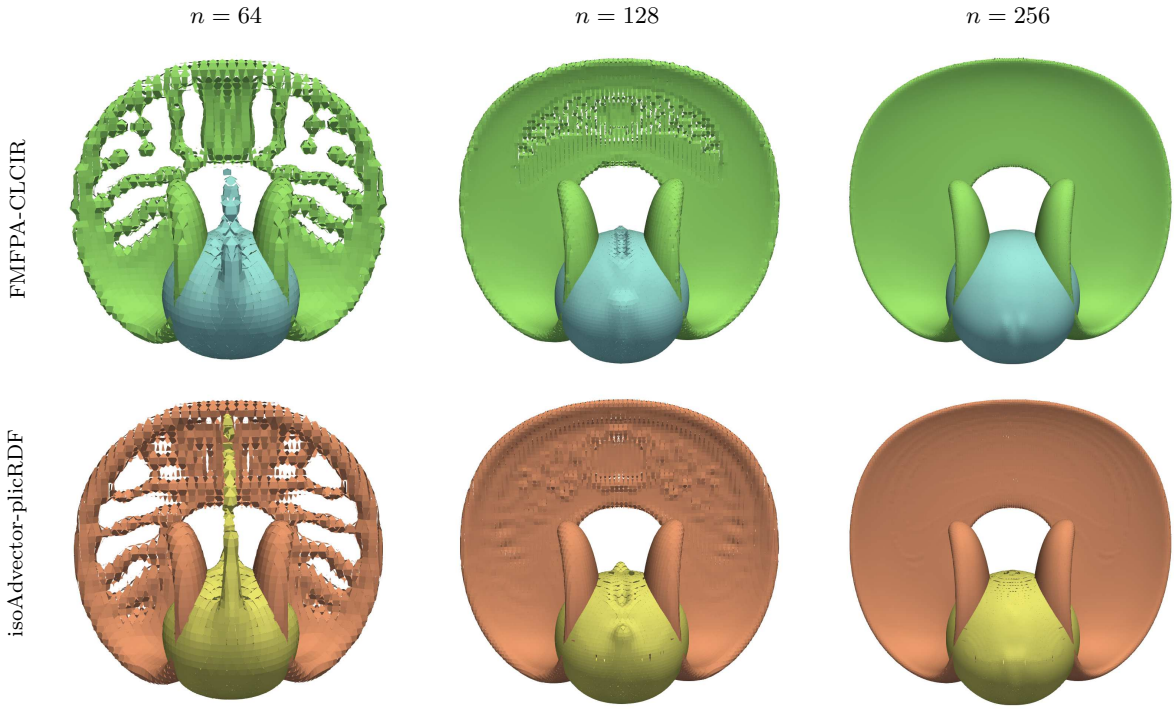
Methods	n	E_g	$\mathcal{O}$	E_{vol}	E_{bound}	$\tilde{t}_{\text{tot}}$	$\tilde{t}_{\text{adv}}$	$\tilde{t}_{\text{rec}}$
Hexahedral meshes								
FMFPA-CLCIR	32	6.485e-03	-	5.204e-17	1.274e-18	1.94	1.78	0.16
isoAdvector-plicRDF	32	7.966e-03	-	8.726e-16	2.170e-17	1.00	0.18	0.82
FMFPA-CLCIR	64	2.070e-03	1.65	7.980e-17	8.008e-19	9.90	9.16	0.75
isoAdvector-plicRDF	64	3.018e-03	1.40	5.938e-15	2.015e-09	5.57	1.43	4.14
FMFPA-CLCIR	128	4.308e-04	2.26	1.299e-08	7.201e-10	56.84	53.18	3.66
isoAdvector-plicRDF	128	7.063e-04	2.10	1.878e-14	1.603e-10	35.64	12.57	23.06
FMFPA-CLCIR	256	6.131e-05	2.81	9.284e-15	2.926e-19	374.91	353.03	21.88
isoAdvector-plicRDF	256	1.061e-04	2.74	1.720e-13	1.924e-11	241.13	108.29	132.85
Tetrahedral meshes								
FMFPA-CLCIR	32	1.326e-02	-	6.980e-06	1.123e-08	17.56	17.03	0.54
isoAdvector-plicRDF	32	1.394e-02	-	1.298e-15	4.887e-06	1.88	0.23	1.65
FMFPA-CLCIR	64	4.231e-03	1.65	3.290e-06	3.054e-10	77.90	75.07	2.83
isoAdvector-plicRDF	64	5.858e-03	1.25	2.475e-15	1.181e-06	19.68	2.90	16.78
FMFPA-CLCIR	128	9.386e-04	2.17	3.346e-07	2.106e-10	396.65	379.87	16.79
isoAdvector-plicRDF	128	1.464e-03	2.00	4.672e-15	1.263e-07	126.15	23.44	102.71
FMFPA-CLCIR	256	1.583e-04	2.57	2.157e-07	9.493e-11	2 694.72	2 563.02	131.70
isoAdvector-plicRDF	256	3.621e-04	2.02	2.618e-15	1.872e-08	1 121.66	244.59	877.07
Unstructured polyhedral meshes								
FMFPA-CLCIR	32	1.052e-02	-	2.516e-07	8.873e-09	56.71	54.00	2.71
isoAdvector-plicRDF	32	1.013e-02	-	6.523e-16	4.481e-07	9.12	3.02	6.10
FMFPA-CLCIR	64	3.570e-03	1.56	3.121e-09	4.467e-09	279.05	258.57	20.49
isoAdvector-plicRDF	64	4.173e-03	1.28	1.631e-16	5.486e-08	162.03	38.97	123.06
FMFPA-CLCIR	128	6.226e-04	2.52	4.321e-08	2.376e-08	1 488.23	1 347.43	140.80
isoAdvector-plicRDF	128	6.945e-04	2.59	2.116e-16	8.783e-09	801.88	275.46	526.42

`.vtk` format using the `printplic` routine of the `gVOF` package. For all mesh types and medium and high resolutions, the final shape obtained with FMFPA-CLCIR is closer to the theoretical sphere. For the finest unstructured polyhedral mesh in Fig. 4.8, the final shape obtained with isoAdvector-plicRDF show a small region of liquid detached from the final sphere.

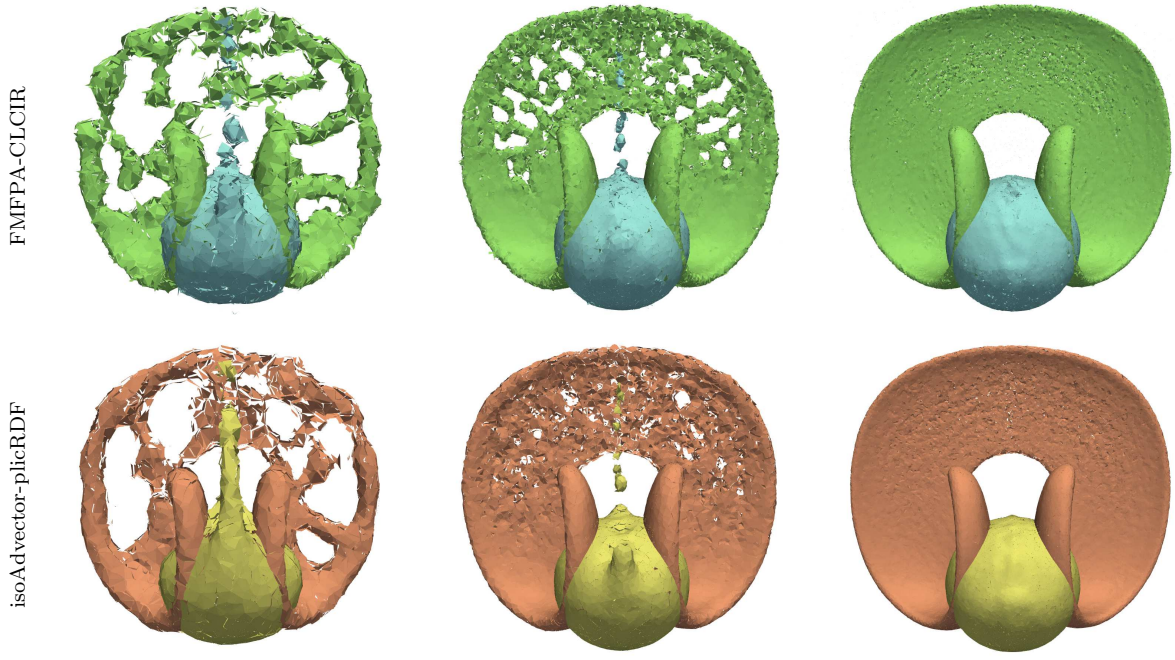
4.4.3 3D shearing

In this test proposed by Liovic et al. [85], a sphere of fluid of radius 0.15, initially centered at (0.5, 0.75, 0.25) in a domain of size $1 \times 1 \times 2$, is deformed in the velocity field defined as

$$\begin{aligned} u(x,y,z,t) &= \sin^2(\pi x)\sin(2\pi y)\cos(\pi t/T), \\ v(x,y,z,t) &= -\sin^2(\pi y)\sin(2\pi x)\cos(\pi t/T), \\ w(x,y,z,t) &= \left\{1 - 2\left[(x-0.5)^2 + (y-0.5)^2\right]^{1/2}\right\}^2 \cos(\pi t/T), \end{aligned} \tag{4.12}$$



(a) Hexahedral meshes



(b) Tetrahedral meshes

Figure 4.7: PLIC interfaces for the 3D deformation test at $t = 1.5$ (green and orange) and $t = 3$ (blue and yellow) using different mesh types and resolutions. Results obtained with CFL=0.5 for the FMFPA-CLCIR (first row for each mesh type) and isoAdvector-plicRDF (second row for each mesh type) methods.

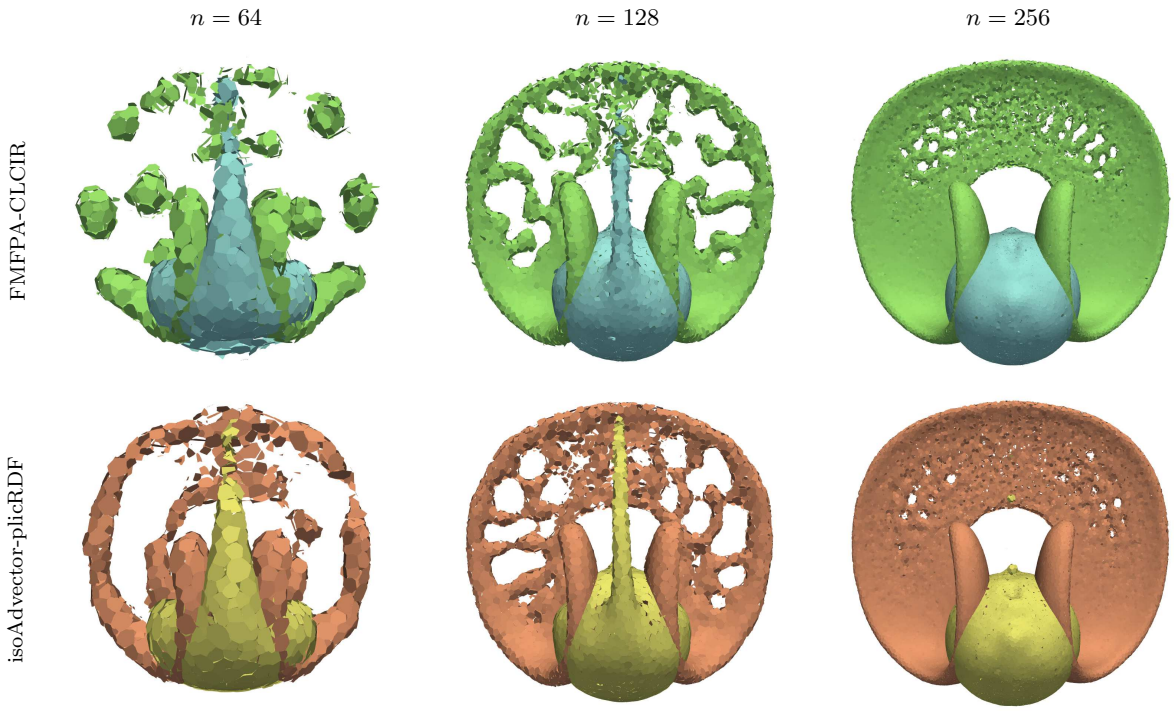


Figure 4.8: Same results as in Fig. 4.7, but for unstructured polyhedral meshes of different resolutions.

where $T = 3$. Table 4.3 shows the errors and consumed cpu-times using the two combinations compared in the test. The cpu-times are also provided relative to the smallest total time value, which in this test is given by $t_{\text{tot}} = 21.8$ ms obtained using the isoAdvector-plicRDF methods for an hexahedral mesh with $n = 32$. Now, for all mesh types and sizes, the L_1 error norm obtained with FMFPA-CLCIR is always smaller than that obtained with the isoAdvector-plicRDF method, with differences ranging from 1.2% for an unstructured polyhedral mesh with $n = 128$ to 68.9% for the finest tetrahedral mesh. Both combinations yield second-order convergence in hexahedral and unstructured polyhedral meshes, but for tetrahedral meshes, the isoAdvector-plicRDF methods do not reach second-order convergence. The error in volume conservation is now always several orders of magnitude smaller in the isoAdvector-plicRDF methods, whilst the boundedness error is of the same order for both combinations in all meshes and resolutions.

The trend in the computational efficiency and the order of magnitude of the total consumed cpu-times is generally the same as those commented in the previous test for hexahedral and tetrahedral meshes. However, the differences are reduced in tetrahedral meshes due to the decrease in the advection time in the FMFPA-CLCIR methods. If the velocities defined by Eqs. (4.11) and (4.12) are compared, it can be seen that the magnitudes of each component are smaller in this test, which implies that the volumetric fluxes are also smaller although the time step is higher in this case (note that the CFL is the same). The direction of the velocity components is also different and combined with the previous makes the flux polyhedra intersect less interface cells

Table 4.3: Errors and consumed cpu-times for the 3D shearing flow test using a CFL = 0.5 for different mesh types and resolutions.

Hexahedral	n	E_g	$\mathcal{O}$	E_{vol}	E_{bound}	$\tilde{t}_{\text{tot}}$	$\tilde{t}_{\text{adv}}$	$\tilde{t}_{\text{rec}}$
Hexahedral meshes								
FMFPA-CLCIR	32	3.509e-03	-	7.027e-07	1.173e-08	1.90	1.74	0.17
isoAdvector-plicRDF	32	4.485e-03	-	9.177e-16	1.515e-09	1.00	0.20	0.80
FMFPA-CLCIR	64	9.686e-04	1.86	8.530e-08	8.896e-10	9.95	9.15	0.80
isoAdvector-plicRDF	64	1.276e-03	1.81	5.586e-15	3.138e-10	5.77	1.56	4.21
FMFPA-CLCIR	128	2.372e-04	2.03	1.028e-08	6.932e-11	60.02	55.79	4.23
isoAdvector-plicRDF	128	3.240e-04	1.98	2.839e-14	2.259e-10	55.46	19.89	35.57
FMFPA-CLCIR	256	5.421e-05	2.13	1.078e-09	4.649e-12	406.33	379.48	26.84
isoAdvector-plicRDF	256	7.690e-05	2.07	8.065e-14	5.695e-11	282.14	125.36	156.77
Tetrahedral meshes								
FMFPA-CLCIR	32	6.184e-03	-	1.900e-05	1.195e-08	13.81	13.44	0.37
isoAdvector-plicRDF	32	8.680e-03	-	6.852e-16	4.118e-06	2.21	0.28	1.93
FMFPA-CLCIR	64	1.537e-03	2.01	6.499e-07	3.979e-10	60.34	58.00	2.34
isoAdvector-plicRDF	64	2.857e-03	1.60	1.952e-15	6.250e-07	16.27	2.85	13.41
FMFPA-CLCIR	128	4.032e-04	1.93	8.923e-09	8.491e-11	316.24	302.71	13.53
isoAdvector-plicRDF	128	9.290e-04	1.62	4.118e-15	5.857e-08	130.93	26.14	104.80
FMFPA-CLCIR	256	1.041e-04	1.95	1.941e-09	5.332e-11	1 771.46	1 686.99	84.47
isoAdvector-plicRDF	256	3.344e-04	1.47	7.615e-16	3.854e-09	1 222.58	270.78	951.80
Unstructured polyhedral meshes								
FMFPA-CLCIR	32	5.348e-03	-	9.613e-08	2.795e-09	45.60	41.50	4.10
isoAdvector-plicRDF	32	5.934e-03	-	5.274e-16	4.262e-07	10.04	2.99	7.05
FMFPA-CLCIR	64	1.569e-03	1.77	4.565e-09	1.917e-10	221.74	201.94	19.80
isoAdvector-plicRDF	64	1.644e-03	1.85	3.990e-16	4.765e-08	197.73	53.18	144.55
FMFPA-CLCIR	128	4.176e-04	1.91	2.720e-08	3.310e-08	1 413.17	1 252.91	160.26
isoAdvector-plicRDF	128	4.225e-04	1.96	9.147e-15	4.617e-09	869.30	291.17	578.13

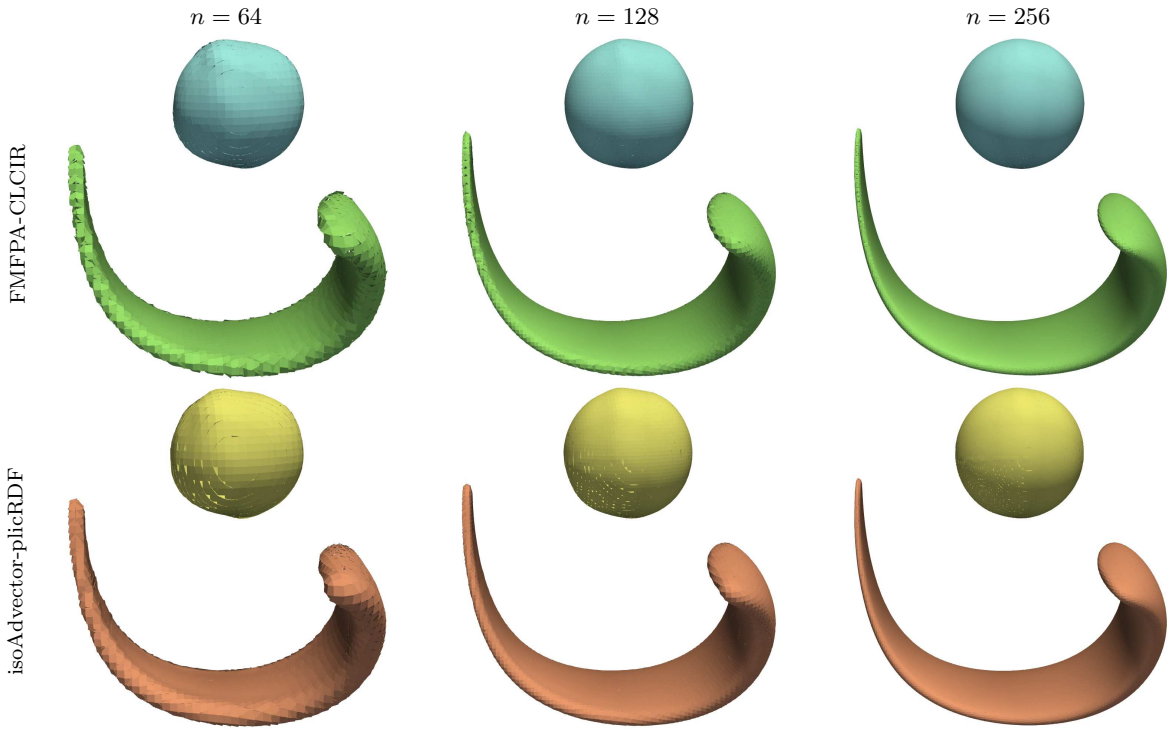
resulting in a reduction in the total number of truncation operations and decreasing the computational time in the advection step.

Figure 4.9 depicts the PLIC interfaces for this test. For the coarsest meshes shown, it can be seen how the shape obtained at the end of the test by FMFPA-CLCIR is closer to the theoretical sphere. For higher mesh resolutions, these differences are also very clear in hexahedral and tetrahedral meshes. However, for the finest unstructured polyhedral mesh, both combinations give very similar intermediate and final shapes.

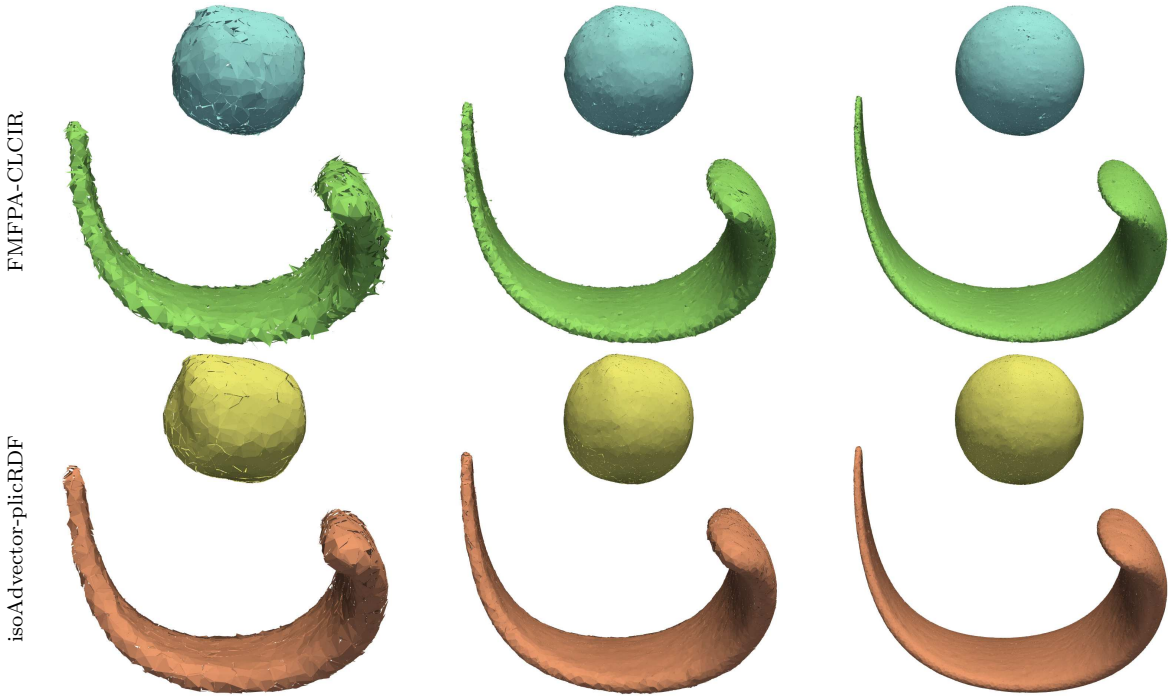
4.5 Non-prescribed velocity tests

4.5.1 Single bubble rising

This test case was described by Hysing et al. [63], and recently used by Gamet et al. [50] on the validation of several reconstruction schemes coupled with the isoAdvector advection step. A gas bubble of initial diameter $D_0 = 0.5$ m with physical properties $\rho_g = 1\text{kg m}^{-3}$ and $\mu_g = 0.1\text{Pa s}$, is surrounded by a liquid of properties $\rho_l = 1000\text{kg m}^{-3}$



(a) Hexahedral meshes



(b) Tetrahedral meshes

Figure 4.9: PLIC interfaces for the 3D shearing flow test at $t = 1.5$ (green and orange) and $t = 3$ (blue and yellow) for different mesh types and resolutions. Results obtained with CFL=0.5 using the FMFPA-CLCIR (first row for each mesh type) and isoAdvector-plicRDF (second row for each mesh type) methods.

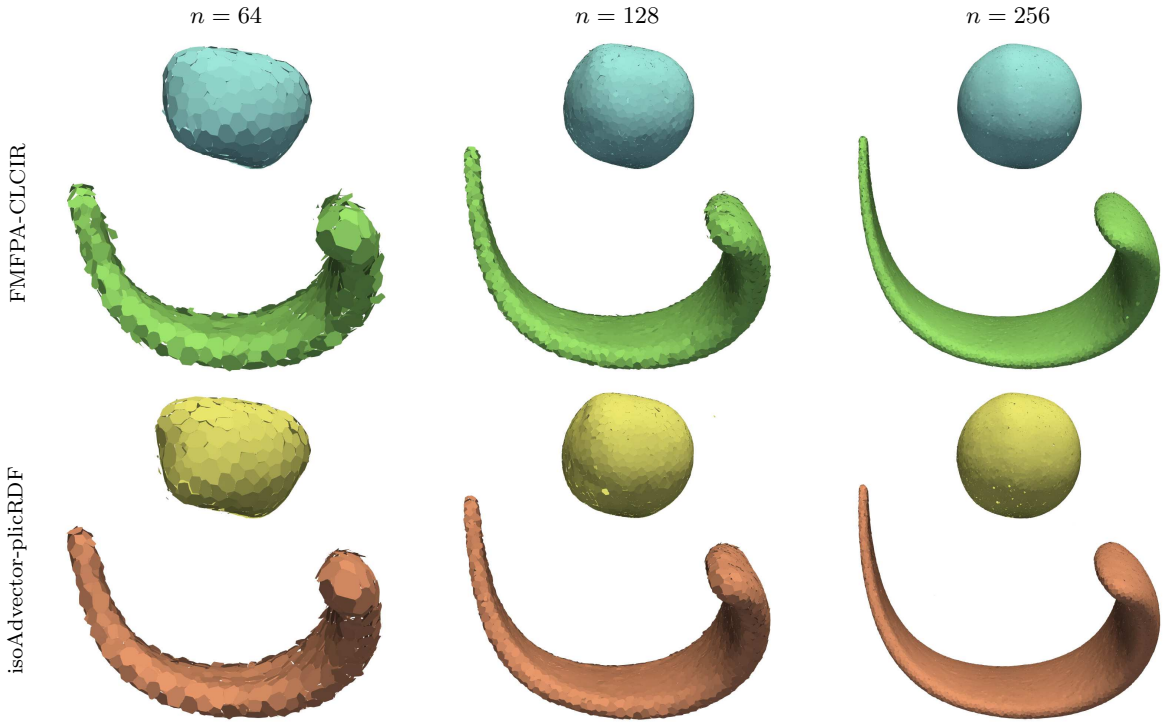


Figure 4.10: Same results as in Fig. 4.9, but for unstructured polyhedral meshes of different resolutions.

and $\mu_l = 10\,\mathrm{Pa\,s}$. The surface tension is $\sigma = 1.96\,\mathrm{N\,m^{-1}}$ and the gravity vector $\mathbf{g} = (0, 0, -0.98)\,\mathrm{m\,s^{-2}}$. Under these conditions, as argued by Hysing et al. [63], the estimated Reynolds number, based on the gas velocity and liquid viscosity and density, is $\mathrm{Re} = 35$, whereas the Weber number is $\mathrm{We} = 125$. Therefore, the expected outcome lies between the skirted and dimpled ellipsoidal-cap regimes, where the bubble breakup can occur [28].

The physical domain size is $2D_0 \times 4D_0$ for the 2D case, and $2D_0 \times 2D_0 \times 4D_0$ for the 3D case. The bubble center is initially placed at (D_0, D_0) and (D_0, D_0, D_0) for the 2D and 3D cases, respectively. Hexahedral, tetrahedral and unstructured polyhedral meshes, generated as described in Section 4.1, with $n = 40$, 80, 160, 320, and 640 are considered in the 2D tests, where n is the number of cells in the x direction. For the 3D test, only an hexahedral mesh with $n = 80$ is considered, and generated with the `snappyHexMesh` tool by imposing on a root mesh of size $10 \times 10 \times 20$ a cylindrical region of diameter $1.4D_0$ centered at the bubble center with its axis parallel to the z direction in the range $0.4D_0 \leq z \leq 3.7D_0$.

The discretization and numerical schemes used in this test are the same for all the methods. For the temporal discretization, the Crank-Nicolson scheme with a blending coefficient, sets the weight to bound the Crank-Nicolson scheme with the first-order implicit Euler scheme, of 0.9 is used. For the gradient terms, the Gauss scheme is selected, and for the convective terms, a TVD scheme with a Sweby limiter function is used. For the boundary conditions, lateral walls are considered as slip walls, meanwhile

top and bottom boundaries are defined as no-slip walls. For the pressure and volume fraction, the zero normal gradient boundary condition is used in all the walls.

For the resolution of the Navier-Stokes equations, the PISO algorithm with the momentum predictor step and 3 iterations is used with a single non-orthogonal corrector step for only tetrahedral and unstructured polyhedral meshes. The condition CFL < 0.075 is used for all meshes and resolutions. A study of the influence of this condition in hexahedral meshes has been carried out, as shown in the following. For the solution of the linear systems of equations, the GAMG is used for the pressure terms, whereas for the velocity, an iterative solver with the simplified diagonal incomplete Cholesky (DIC) smoother is selected. The tolerance for the volume fraction is set as $F_{\text{tol}} = 1 \times 10^{-8}$.

In order to compare quantitatively the results obtained with both combinations, the bubble velocity, the circularity (2D) and sphericity (3D) are computed. The bubble velocity is calculated as the following weighted average through the entire domain,

$$\mathbf{u}_b = \frac{\sum_i \mathbf{u}_i \left(1 - F_i\right) V_i}{\sum_i (1 - F_i) \, V_i}, \tag{4.13}$$

using the volume of gas $(1 - F_i) \, V_i$ as weight. Circularity and sphericity are calculated as

$$\frac{[V_b/(\pi \Delta y)]^{1/2}}{A_b/(2\pi \Delta y)}, \quad \text{and} \quad \frac{[3V_b/(4\pi)]^{2/3}}{A_b/(4\pi)}, \tag{4.14}$$

respectively, where Δy is the mesh size in the direction perpendicular to the xz plane in the 2D simulations (note that 2D cells are polygonal prisms, thus Δy corresponds to their height), V_b the bubble volume calculated as $V_b = \sum_i (1 - F_i) \, V_i$, and A_b the bubble area. This last term is obtained from the 0.5-isosurface area, which is calculated from the isosurface extracted using the isoap method. This area calculation procedure is used for the two compared combinations, so that the circularity and sphericity at $t = 0$ ms are obviously the same for both methods.

Figure 4.11 shows the vertical component of $\mathbf{u}_b$, namely rise velocity u_b, as a function of time for a 2D hexahedral mesh with $n = 160$. As the CFL decreases, the velocity converges to the same solution for both combinations with almost any difference. It can be seen that the variation in the solution between the two smallest CFL numbers is very small (less than 1%). Thus, these results manifest that using the condition $\text{CFL}_{\text{max}} < 0.075$ does not suppose a significant decrease in accuracy, whilst the time step is increased by a factor of 2 with respect to the lowest CFL number, saving computational time.

Figure 4.12 depicts the time evolution of the rise velocity and circularity for different 2D mesh types and resolutions, and Fig. 4.13 shows the 0.5-isosurfaces for the three finest mesh sizes at the final instant of the test. For hexahedral meshes, the isoAdvector-plicRDF methods yield slightly higher velocities for coarser meshes, but

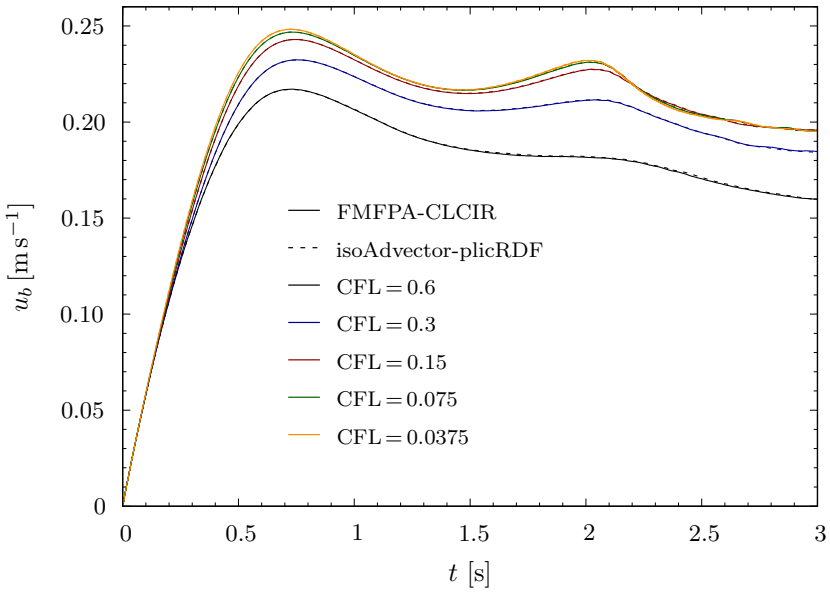


Figure 4.11: Time evolution of the bubble rise velocity for different CFL numbers and a 2D hexahedral mesh with $n = 160$. Comparison between isoAdvector-plicRDF and FMFPA-CLCIR.

as the mesh size is increased, both combinations give very similar results. Differences in the circularity are more clear and as the mesh is refined both combinations converge to very similar solutions. For $n = 40$, the secondary bubbles detach almost at the same time (around $t \approx 2.64$ ms) in both combinations. These similarities are shown in Fig. 4.13(a) for all resolutions. This implies that for hexahedral meshes and the CFL number considered the results are very similar for the advection-reconstruction methods compared here, and these results are mainly dependent on other factors, such as the tolerances used in the PISO algorithm or the error introduced by the discretization schemes.

For tetrahedral meshes (Fig. 4.12(b)), the differences in the rise velocity and circularity are now more evident. For all mesh resolutions, isoAdvector-plicRDF yields from $t = 0.7$ ms a higher value of the rise velocity. For $n = 40$, the time evolution obtained with isoAdvector-plicRDF shows several oscillations from $t = 1.9$ ms and in that obtained with FMFPA-CLCIR these oscillations also appear but with a smaller amplitude. For $n = 160$, the results obtained with both combinations are very similar, whilst for the remaining sizes the coincidences are not so good as for hexahedral meshes. In the time evolution of the circularity, the isoAdvector-plicRDF methods give rise, in general, to higher values for all mesh resolutions. The coincidence in the results for $n = 160$ can also be seen in the 0.5-isosurface shown in Fig. 4.13(b), where for this resolution both combinations yield very similar shapes. For $n = 320$ and 640, the interfaces are slightly different for both combinations, and they show a more apparent asymmetry than in hexahedral meshes.

For unstructured polyhedral meshes (Fig. 4.12(c)), the differences between both combinations in the rise velocity are now very small for all resolutions. For $n = 640$,

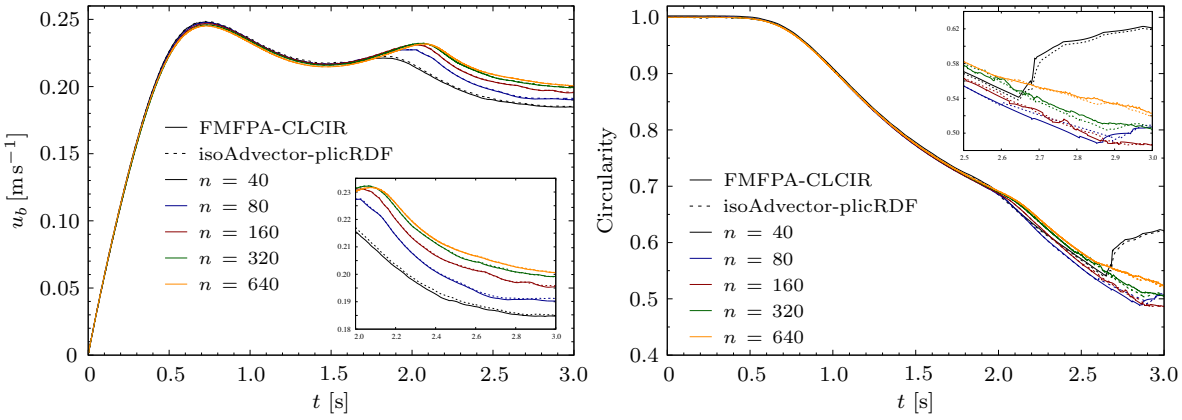
both combinations yield almost the same solution. However, in the time evolution of circularity, at low resolutions and after $t = 2.5$ ms the results are slightly different, although both combinations follow a similar trend. If the attention is now focused on the isosurface contours depicted in Fig. 4.13(c), it can be seen that although the results are again very similar, the FMFPA-CLCIR methods yield more evolved secondary bubbles, i.e., bubbles that seem to detach a few instants before the same bubbles that appear in the results provided by the isoAdvector-plicRDF methods.

Figures 4.14 and 4.15 show the results obtained in the 3D hexahedral mesh with $n = 80$. The rise velocity time evolution (Fig. 4.14(a)) shows that, again, both combinations yield very similar solutions, although after $t = 0.5$ s, the velocity obtained with isoAdvector-plicRDF is slightly higher. Figure 4.15 shows the 0.5-isosurfaces at different instants which are extracted with ParaView [119] for visualization purposes. Only a half of the bubble for each method is represented in order to show more clearly the differences between the numerical results. From $t = 1$ s to $t = 2.5$ s, both combinations yield very similar results. At instant $t = 3$ s, the holes that appear in the bubble tail are due to the lack of mesh resolution. The undulations formed at the lower part of the tail, that remain at $t = 3.5$ s, are very similar in both combinations, as well as is the shape of the tail breakup. The top part of the bubble shows almost no difference between both combinations. These results were somehow expected since the rise velocity and circularity evolutions are very similar.

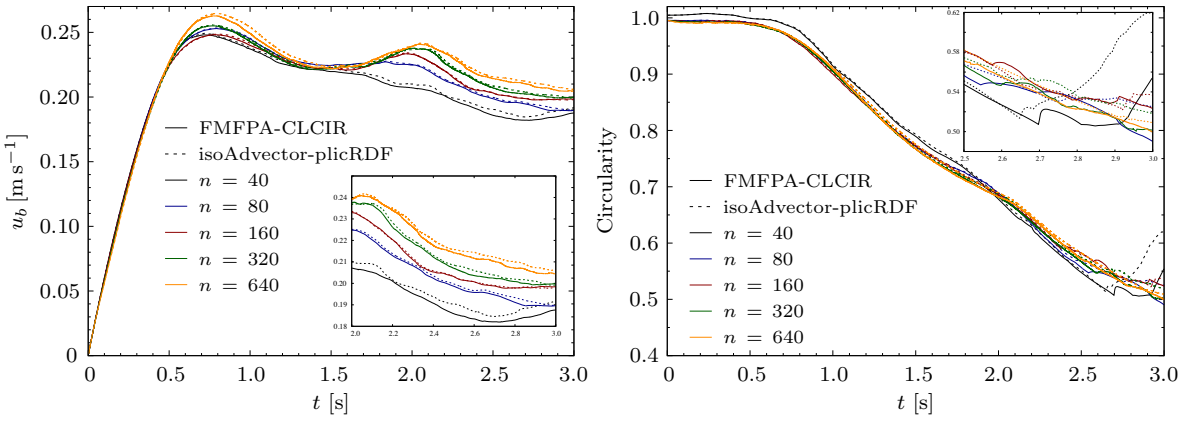
4.5.2 Drop impact on a deep pool

The experimental results of this test are obtained from [60]. It consists on the impact of a water drop of initial diameter $D_0 = 2.9$ mm on a deep pool of the same liquid at a velocity $U = 1.55\,\mathrm{m\,s^{-1}}$. The water has physical properties $\rho_l = 1000\,\mathrm{kg\,m^{-3}}$, $\mu_l = 1\,\mathrm{mPa\,s}$ and $\sigma = 72.6\,\mathrm{mN\,m^{-1}}$. Hence, the Weber and Froude $(U^2(g\,D_0)^{-1})$ numbers are We $= 96$ and Fr $= 85$. The pool has an initial depth of $4.5D_0$ and the drop center is initially placed at $(0, 0, 5.1D_0)$ to avoid possible differences in the velocity impact between both combinations. The velocity field is initialized by imposing the velocity vector $(0, 0, -U)$ at cells whose volume fraction is greater than 0 and the PISO algorithm calculates the initial pressure field based on this initial velocity field.

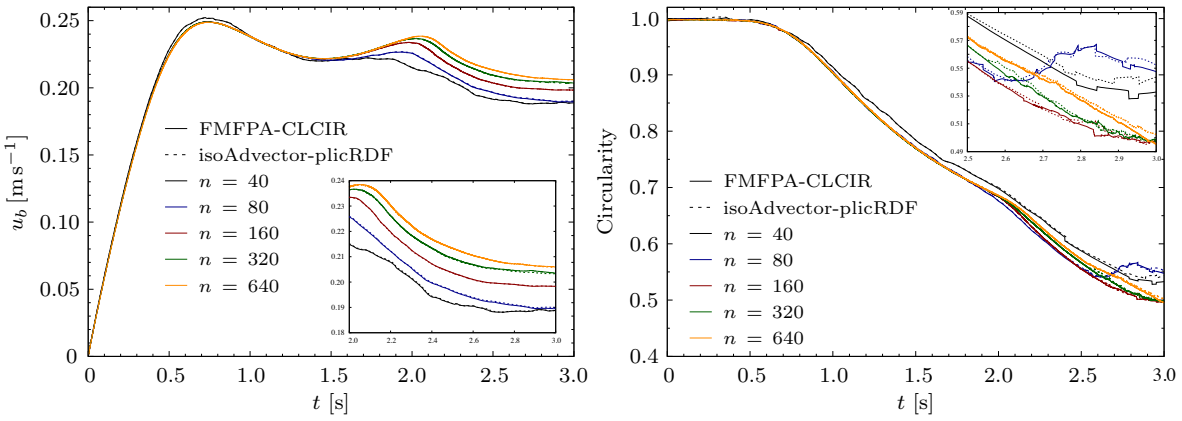
Making use of the symmetry of the problem, only a quarter of it is solved using a physical domain of $3.5D_0 \times 3.5D_0 \times 7D_0$, which is discretized with a statically refined root mesh of $7 \times 7 \times 14$ cells. Three concentric cylindrical regions centered at the coordinate origin, with their axes parallel to the z direction with 4 refinement levels are imposed to the root mesh: the first cylindrical region has a diameter of $0.345D_0$ and is imposed for $2.07D_0 \leq z \leq 4.14D_0$, the second has a diameter of $1.035D_0$ and is imposed for $4.14D_0 \leq z \leq 5D_0$, and the last has a diameter of $0.345D_0$ and is imposed for $5D_0 \leq z \leq 6.03D_0$. The equivalent mesh resolution is 16 cpr, which is maintained near the interface throughout the entire simulation. Fig. 4.16 shows the



(a) Hexahedral meshes

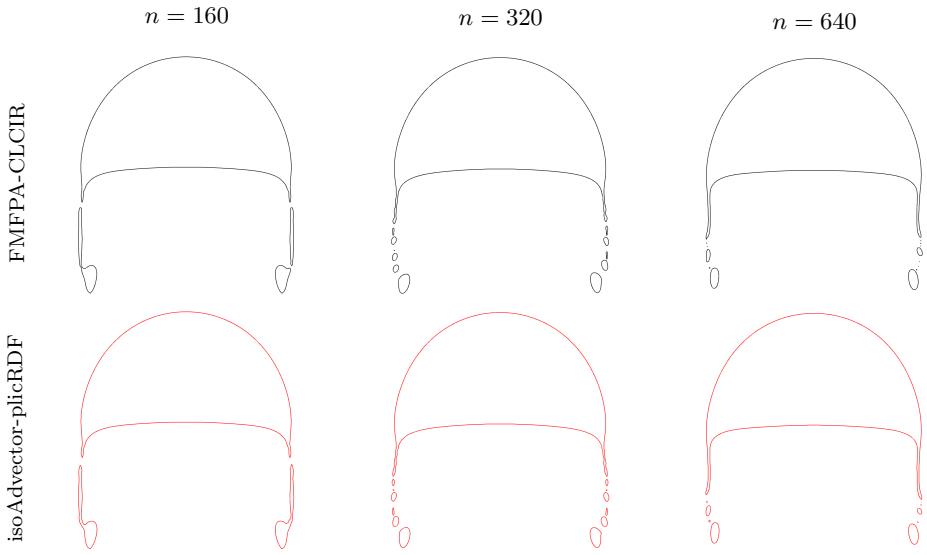


(b) Tetrahedral meshes

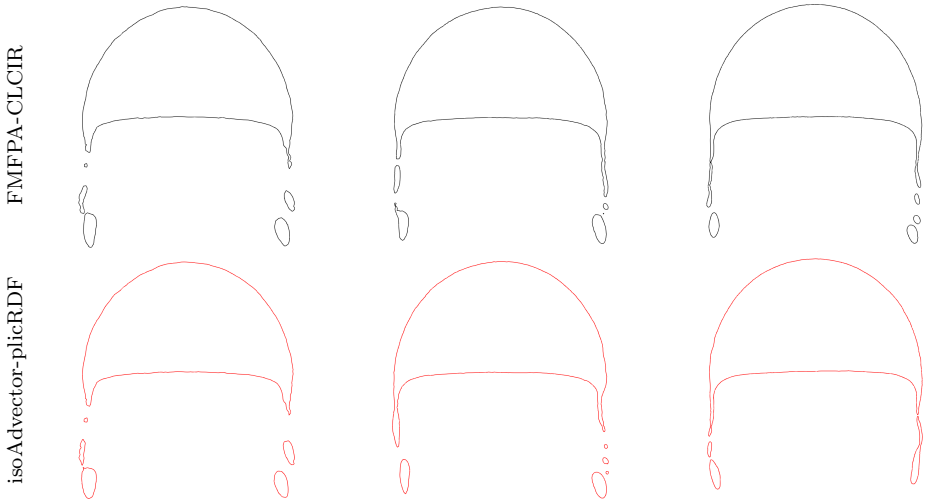


(c) Unstructured polyhedral meshes

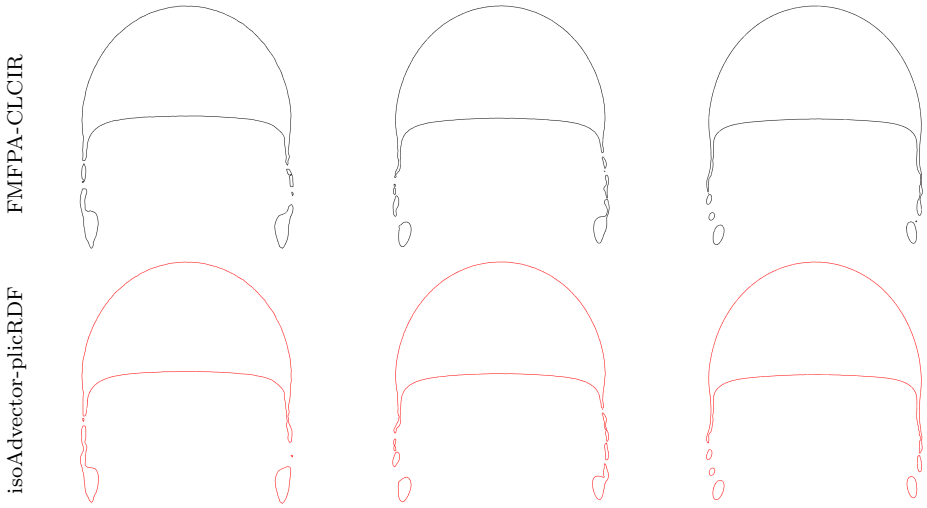
Figure 4.12: Rise velocity and circularity as a function of time for different 2D mesh types and resolutions using the FMFPA-CLCIR and isoAdvector-plicRDF methods. The insets show a detailed view of the last instants.



(a) Hexahedral meshes

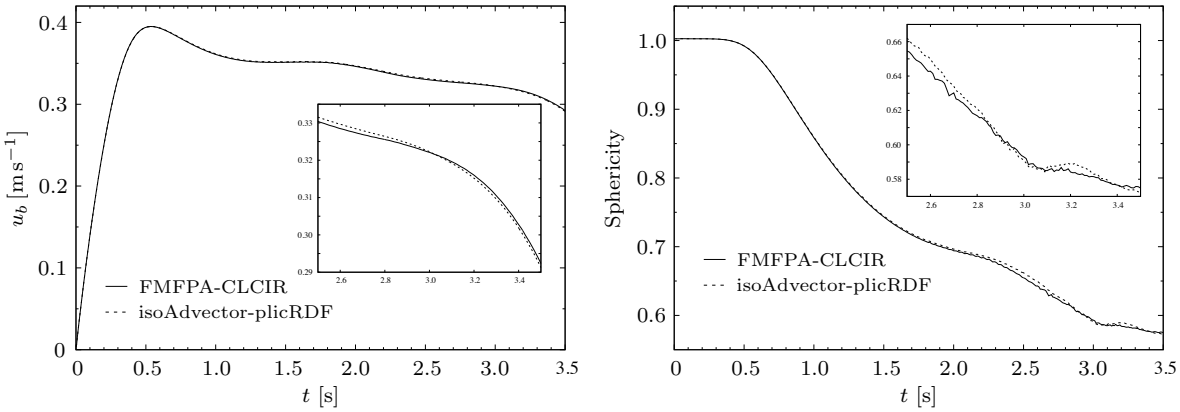


(b) Tetrahedral meshes



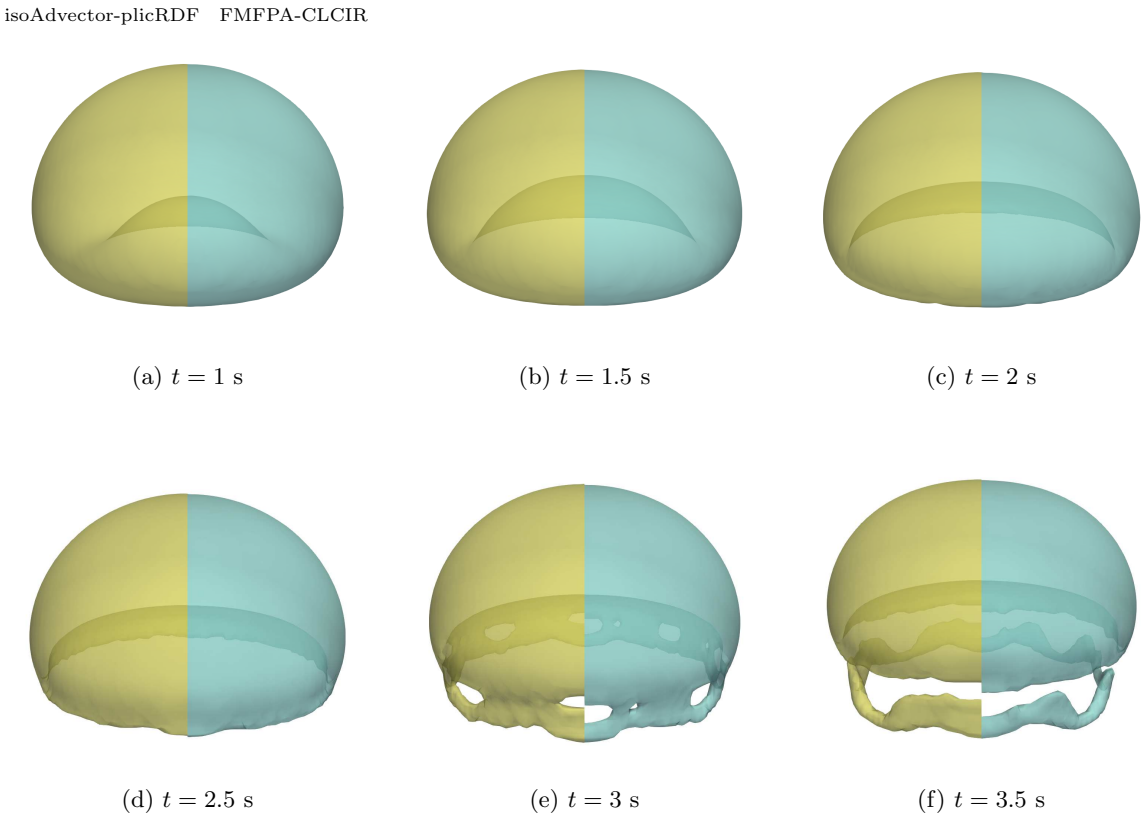
(c) Unstructured polyhedral meshes

Figure 4.13: 0.5-isosurface contours of the bubble at $t = 3$ s for different 2D mesh types and resolutions. Comparison between the FMFPA-CLCIR (black contours) and isoAdvector-plicRDF (red contours) methods.



(a) (b)

Figure 4.14: Rise velocity (a) and sphericity (b) as a function of time for a 3D hexahedral mesh with $n = 80$ using the FMFPA-CLCIR and isoAdvector-plicRDFmethods. The insets show a detailed view of the last instants.



(a) $t = 1$ s (b) $t = 1.5$ s (c) $t = 2$ s

(d) $t = 2.5$ s (e) $t = 3$ s (f) $t = 3.5$ s

Figure 4.15: 0.5-isosurface contours of the bubble at different instants obtained using the isoAdvector-plicRDF (yellow) and FMFPA-CLCIR (blue) methods for an hexahedral mesh with $n = 80$. Only a half of the complete bubble is represented.

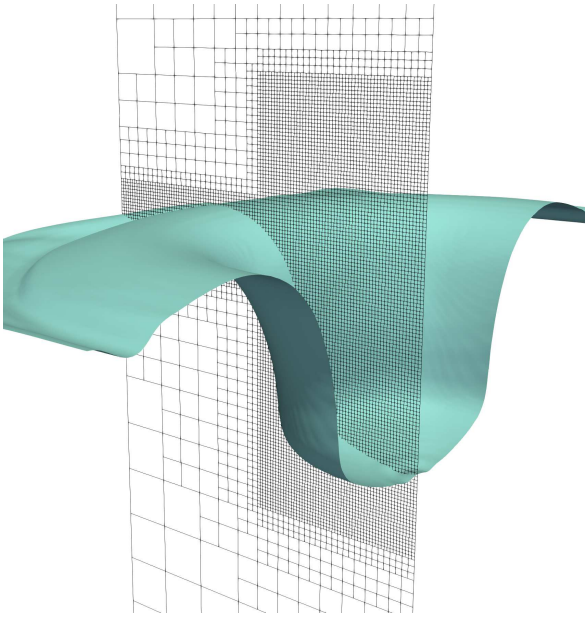


Figure 4.16: Results for the drop impact on a deep pool obtained using the FMFPA-CLCIR methods. Detail of the statically refined mesh with three cylindrical regions of four refinement levels on each one and the 0.5-isosurface contour at $t^* = 7.3$.

statically refined mesh, where the three cylindrical regions used to refine the mesh near the interface can be observed, and the 0.5-isosurface at the non-dimensional time $t^* = tU/D_0 = 7.3$ obtained using the FMFPA-CLCIR methods. It should be noted that, although only a quarter of the problem has been solved, a symmetry operation has been performed in the ParaView program to show a more complete view of the results.

In order to simulate a quarter of the drop impact, the boundaries normal to the x and y directions passing through the coordinate origin are considered as symmetry planes. The remaining boundaries are considered as walls, applying the no-slip boundary condition for the velocity. For the volume fraction and the pressure, the corresponding gradients normal to the walls are set to zero. In the discretization of the gradient terms, the Gauss scheme is used, whereas for the convective terms, a TVD scheme with a Sweby limiter function is chosen. For the temporal discretization, the first-order implicit Euler scheme is chosen. In the resolution of the systems of algebraic equations, the GAMG solver with a DIC smoother is used for the pressure and velocity terms. For the resolution of the Navier-Stokes equations, the PISO algorithm is used with 3 corrections and the condition $\mathrm{CFL}_{\max} < 0.2$ is set for the calculation of the variable time step. The tolerance for the volume fraction is set to 10^{-6} in both combinations.

Figure 4.17 shows a comparison between the experimental and the numerical results obtained using the FMFPA-CLCIR and isoAdvector-plicRDF methods, for which the 0.5-isosurfaces, are depicted in blue and yellow colors, respectively. Both combinations give very similar solutions until $t^* = 8.7$. Then, the numerical simulations are retarded

about a dimensionless time of 0.9 with respect to the experimental results; difference which is maintained over the rest of the simulation. Therefore, in order to compare the numerical and experimental results, the instant at which the former coincide with the latter for the experimental time $t^* = 10.6$ is found, which corresponds with the instant represented in Fig. 4.17(e) at which the bubble detaches from the free surface. Then, the instant represented in Fig. 4.17(d) is that preceding the bubble detachment.

At instants $t^* = 11.1$ and 11.4, the crater stretches and an air bubble is entrapped, which appears fully detached from the crater at $t^* = 11.5$. Both combinations can reproduce well the dynamics of the bubble entrapment although FMFPA-CLCIR yield a deeper air entrapment at $t^* = 11.4$, and thus, the air bubble position at the subsequent instants is deeper from the free surface than in the results provided by isoAdvector-plicRDF. Also, at $t^* = 11.4$, a secondary bubble entrapped is predicted by the FMFPA-CLCIR methods, which is not observed experimentally, although, as the secondary bubble is very small compared to the main bubble, the lack of resolution in the experimental images might be the cause for not observing this phenomenon. At $t^* = 13.6$, the isoAdvector-plicRDF methods predict a secondary droplet detaching from the Rayleigh jet, which is neither observed in the experimental results. The Rayleigh jet is thinner and taller in the numerical results obtained with isoAdvector-plicRDF, which suggests a higher vertical velocity that probably causes the droplet detachment.

Figure 4.18 shows the volume conservation error as a function of the dimensionless time for the deep pool impact. Three curves are shown: one for the FMFPA-CLCIR methods and two for the isoAdvector-plicRDF methods. In the last case, the bounding procedure is used with 5 iterations (bounding) and not used (no bounding) by making `nAlphaBounds` equal to 0 and setting the snap tolerance equal to the volume fraction tolerance in order to avoid unphysical results, i.e., if the calculated volume fraction in a cell is $F < F_{\text{tol}}$ or $F > 1 - F_{\text{tol}}$, the resulting volume fraction value is snapped to 0 or 1, respectively. It is clear that the bounding procedure drops dawn the volume conservation error since the over/undershoots produced in the advection are redistributed, keeping this error almost constant during all the simulation. However, if this procedure is not executed during the calculation, the error obtained with isoAdvector-plicRDF is closer to that obtained with FMFPA-CLCIR, nearly in the same order of magnitude. The increased order of magnitude of this error obtained for both combinations compared to the advection tests might arise from the fact that in this test the velocity field is velocity-divergence free only up to a certain tolerance, which in this case is set to 10^{-7}. This generates more over/undershoots in the advection step.

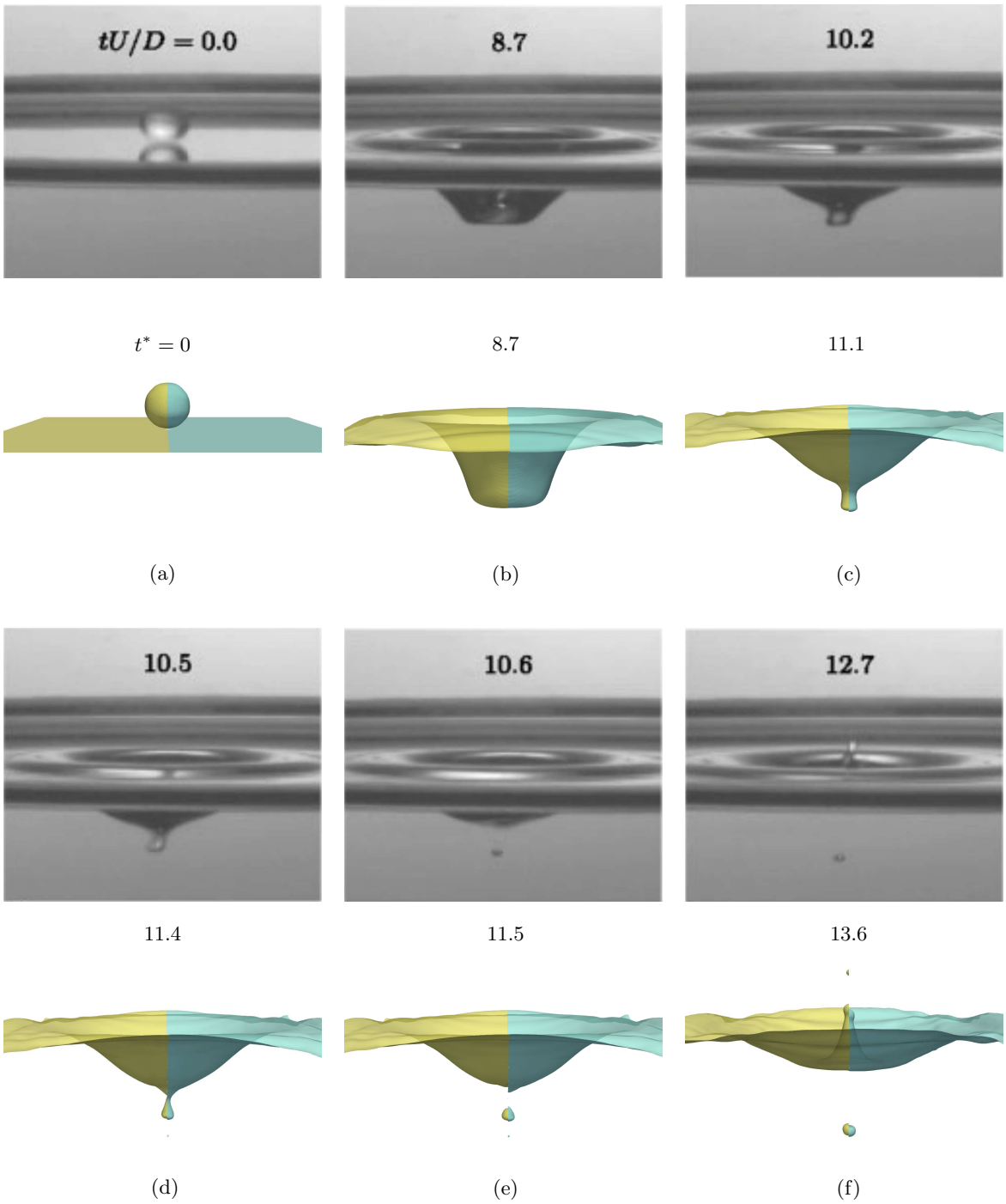


Figure 4.17: Comparison between the experimental results obtained from [60] (first and third rows) and the numerical results showing the 0.5-isosurface (second and fourth rows) obtained in the deep pool impact test using the FMFPA-CLCIR (blue) and isoAdvector-plicRDF (yellow) methods.

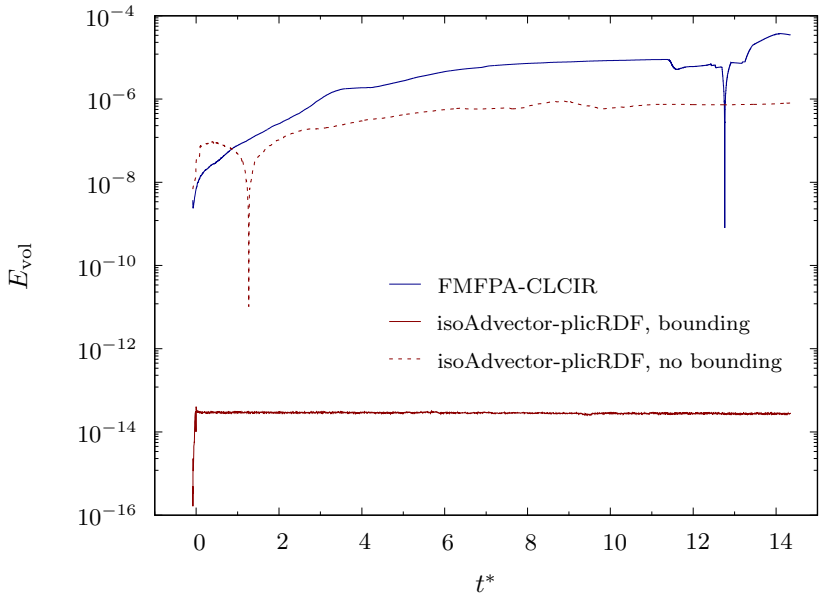


Figure 4.18: Time evolution of the volume conservation error in the deep pool impact test obtained using the FMFPA-CLCIR and isoAdvector-plicRDF methods in an hexahedral mesh with $n = 80$.

4.6 Alternative implementation analysis

4.6.1 Efficiency analysis

In order to compare the differences in efficiency of the two parallelization applications implemented (OpenMP and MPI, as described in Section 3.2.3) in the CLCIR method, several simulations are carried out measuring the consumed cpu-time during the reconstruction step. It is important to note that the results obtained with the MPI application are very dependent on the method used to decompose the domain as well as the case itself, since through the simulation the interfacial cells can all lie in a single processor or can be distributed among a certain amount of the total number of processors used, n_{proc}. For this study, the simple decomposition is chosen, in which the domain is subdivided in as many subdomains as processors are used, specifying for each direction the number of subdomains, written in the form $(n_{\text{sd},x}, n_{\text{sd},y}, n_{\text{sd},z})$, where the total number of subdomains, n_{sd}, is equal to the number of processors, following

$$n_{\text{sd}} = n_{\text{proc}} = n_{\text{sd},x} \times n_{\text{sd},y} \times n_{\text{sd},z}. \tag{4.15}$$

For example, for $n_{\text{proc}} = 16$, the valid decompositions would be $(1, 1, 16)$, $(1, 2, 8)$, $(2, 2, 4)$, $(4, 4, 1)$, and their corresponding permutations, i.e., there are 15 different possibilities. Depending on the computational domain, and the characteristics of the case that is being solved, such as the expected symmetry of the problem, the time evolution, etc., the total amount of cells and the number of interfacial cells per processor may vary, thus, it is difficult to make a general estimation of the performance using MPI parallelization. Besides, it also depends on the hardware used, which influences

the communication between processors. In any cases, a roughly estimation of the efficiency of the MPI and OpenMP parallelizations is carried out using two different tests.

The first test consists on the reconstruction of a sphere of radius 0.325 whose center is placed at $(0.5, 0.5, 0.5)$ in a unit-cubic domain, which is discretized using an hexahedral mesh with $n = 64$, 128, and 256. The two different parallelizations of the CLCIR method are used and the number of processors is varied as $n_{\text{proc}} = 1$, 2, 4, 8, 16, 32, and 64. Figure 4.19 shows the speedup ratio, $t_{\text{rec}}(1)/t_{\text{rec}}(n_{\text{proc}})$, and efficiency, $t_{\text{rec}}(1)/[t_{\text{rec}}(n_{\text{proc}}) \times n_{\text{proc}}]$, where $t_{\text{rec}}(1)$ is the execution time when using only one processor, as a function of the number of processors n_{proc} used for the three mesh sizes and for the OpenMP and MPI parallelizations of the CLCIR method.

For $n = 64$ and 256 and all number of processors, the MPI parallelization yields higher speedup values, while for $n = 128$ and $n_{\text{proc}} = 16$, the OpenMP parallelization is faster (Fig. 4.19(a)). The speedup values obtained for low numbers of processors are similar, but as n_{proc} increases, the differences in the speedup also increase. The MPI parallelization also shows better efficiency values (Fig. 4.19(b)) although as n_{proc} increases, the efficiency decays faster than in the OpenMP implementation due to that the number of processors involved in the interface reconstruction (processors with interfacial cells) does not necessarily increase with the total number of processors, since the domain is divided in more subdomains but equally distributed. In this case, for all mesh resolutions, the number of processors that are actually used for the reconstruction in the MPI parallelization is 32 for $n_{\text{proc}} = 64$ and 24 for $n_{\text{proc}} = 32$, whereas for the rest is the same as their corresponding n_{proc}. This behavior can be improved if other different domain decomposition methods are used, such as the Scotch algorithm [25], or if the sub-region procedure is used, in which the domain is subdivided accordingly to a certain distribution, i.e., it allows to use more processors for the regions of interest and leave fewer processors for the regions which are not.

It has been found that the most time consuming step for the MPI parallelization of CLCIR is the construction of the cell-node neighbors stencil using a procedure that allows the communication of all processors neighboring the cells that have, at least, a face lying on a processor boundary. Since this procedure is only needed for the normal estimation using the least squares gradient technique (Youngs method) in cells where more than a single 0.5-isosurface has been extracted, and, usually, the number of interfacial cells reconstructed with such method is small compared to the total number of interfacial cells ($\simeq 0.1\%$ in the reconstruction of a sphere), it has been substituted by a procedure that does not allow the communication between processors but that is more efficient. Thus, cells in which the normal vector needs to be estimated with the least squares gradient and that have at least one of its faces in a processor boundary are treated as boundary cells, i.e., the volume fraction value of the neighbor cells that belong to a different processor than that owning the considered cell are neglected.

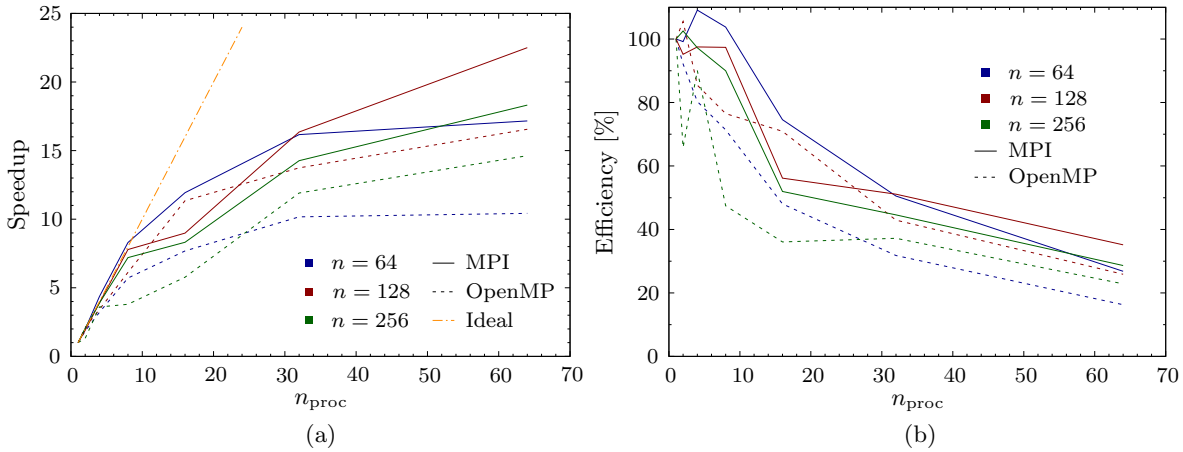


Figure 4.19: Speedup ratio (a) and efficiency (b) as a function of the number of processors for different hexahedral mesh sizes in the reconstruction of a sphere of radius 0.325 in a unit-cubic domain using different parallelizations of the CLCIR method.

This approximation leads to an important increase in efficiency (a reduction in the reconstruction time of about an order of magnitude) but also can lead to a decrease in accuracy if the amount of cells reconstructed with Youngs' method lying in the processors boundaries is important. However, this type of boundary cells has not been found in the tests carried out.

This enhancement in the implementation has been used in the previous test, but, in order to quantitatively show the improvement reached, Fig. 4.20 depicts the reconstruction times obtained using both MPI and OpenMP parallelizations along with the MPI parallelization taking into account the Youngs cells liying at the processors boundaries (MPI-Youngs parallel). It can be observed that the performance improvement obtained neglecting the boundary Youngs cells is in general, as mentioned before, about an order of magnitude and, as the number of processors increases, the time required for the reconstruction with the MPI parallelization is very similar to that obtained with the OpenMP. For $n_{\text{proc}} = 1$, the OpenMP implementation, which is fully written in `FORTRAN`, shows a better performance than the MPI implementation, partially written in `C++`, with reconstruction times about a 25% lower. This is mainly due to the data copy required in the integration of the `VOFTools` and `isoap` libraries into OpenFOAM, although a solution based on data referencing is being implemented at the moment, with promising results.

The ability of the CLCIR implementation to work on the simulation of a multiphase-flow problem in which the Navier-Stokes equations are solved using the parallelization of the complete code has also been tested. Since the interest is focused only on the efficiency of the CLCIR method, the interface is reconstructed every time step but it is not used in the advection step, to avoid the dependence of the results on the accuracy of the reconstruction. With this purpose, the reconstruction step has been integrated into the interFoam solver, which makes use of the algebraic MULES scheme,

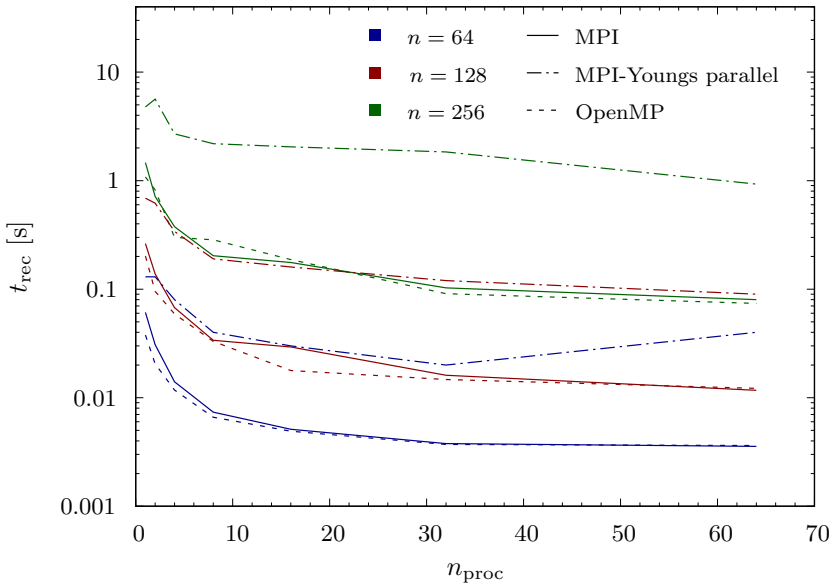


Figure 4.20: Reconstruction time as a function of the number of processors for different hexahedral mesh sizes in the reconstruction of a sphere of radius 0.325 in a unit-cubic domain using different parallelizations of the CLCIR method.

for solving the 3D rising bubble case simulated in Section 4.5.1, but using an uniform hexahedral mesh of $40 \times 40 \times 80$ cells and a CFL number of 0.2. As in the previous test, several number of processors have been used. Figure 4.21 shows the total cpu time consumed in the simulation of the test case and the total reconstruction time for both parallelizations of the CLCIR method relative to the total time consumed by the default interFoam solver on a single core, which is equal to 605 s. For the MPI, as the number of processors increases, the total simulation time is reduced and, for $n_{\text{proc}} = 32$, the time is almost an order of magnitude lower than that obtained for the sequential simulation ($n_{\text{proc}} = 1$). The reduction of the reconstruction time is less evident, but both times follow the same trend with a stabilization between $n_{\text{proc}} = 32$ and 64. As in the previous test, this is due to the subdomain decomposition and, therefore, to the number of processors involved during the simulation. A possible solution might be to use a different subdomain decomposition to increase the number of processors used in the reconstruction step.

For the OpenMP implementation, the total time remains almost constant for all the number of processors considered since, as stated at in Section 3.2.3, the solution of the Navier-Stokes equations consumes a huge amount of the total time, whereas the time spent in the interface reconstruction is less than the 20% of the total time. Therefore, as suggested initially, using this last implementation in OpenFOAM makes the use of CLCIR impractical, since its parallelization with OpenMP limits the rest of the calculation to a single core. However, it can be observed how increasing the number of cores decreases the reconstruction time with a higher rate than in the MPI implementation. This is due to that this parallelization application does not depend

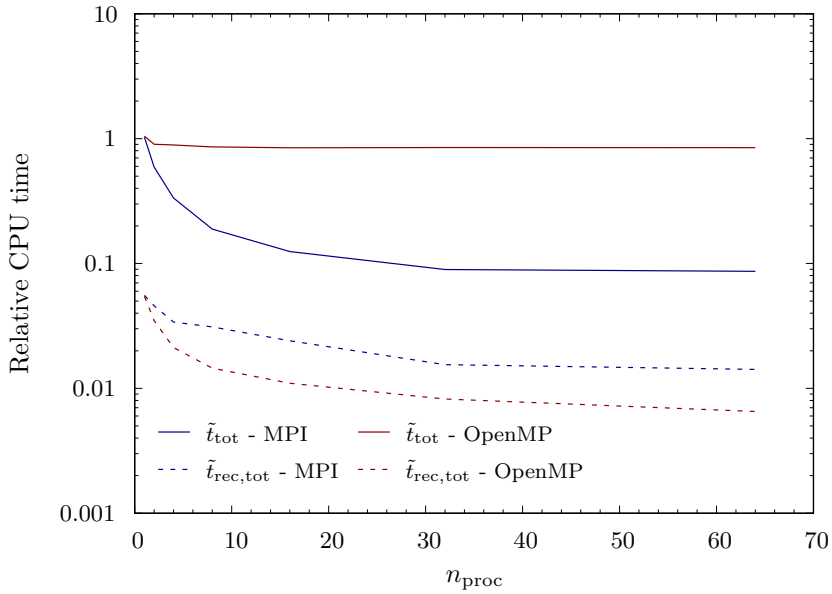


Figure 4.21: Reconstruction and total times as a function of the number of processors in the 3D rising bubble test. Reconstruction times correspond to the OpenMP and MPI implementations of the CLCIR reconstruction method, whereas the total time corresponds to the time consumed through the entire simulation.

on the domain decomposition and also to that if the number of interfacial cells is high for a given mesh, as it happens in this case due to the interface smearing in MULES, the MPI implementation is less efficient. Note also that the total time given by the OpenMP implementation for $n_{\text{proc}} = 1$ is slightly higher than that when using $n_{\text{proc}} > 1$. This is due to that some operations in OpenFOAM are prepared to work with the OpenMP application but, clearly, it does not result in a significant reduction of the calculation time and remains almost independent of n_{proc}.

4.6.2 Coupling CLCIR reconstruction with isoAdvector advection

In Section 4.3 it was shown that the plicRDF method is as accurate as the CLCIR method, whereas the latter is significantly faster for all mesh types and resolutions considered in the tests. In Section 4.4, the greater efficiency achieved by isoAdvector compared to the FMFPA method for tetrahedral and unstructured polyhedral meshes has been revealed, whilst for hexahedral meshes the differences are lower. However, in terms of advection accuracy, the results are less clear since these are affected by the error introduced by the reconstruction algorithm, which is different in each case. Thus, one may think that combining isoAdvector with CLCIR can be an interesting approach to, on one side, compare the true accuracy of the advection step, since both advection methods, isoAdvector and FMFPA, would make use of the same reconstruction method, and, on the other side, check wether the greater efficiency yielded by the CLCIR method speeds up the total time consumed during the advection and reconstruction steps.

The combination of these two schemes has been done making use of the proposed alternative implementation for the CLCIR method, described above, and the template class `reconstructionMethods` available in OpenFOAM v2106. In addition, in order to make CLCIR work along with isoAdvector, the PLIC center (assumed to be the interface center $\mathbf{x}_{\text{int}}$) must be calculated in the reconstruction step since it is required by the advection algorithm to estimate the interface movement within the time step (see Eq. (3.47)). Thus, the computation of the PLIC center is carried out by truncating the cell with its PLIC plane using the `VOFTools` library.

To compare the accuracy and efficiency of this combination of advection and reconstruction methods, the 3D deformation and the 3D shearing tests carried out in Sec. 4.4 are again performed. Tables 4.4 and 4.5 show the results obtained for each test. The new combination yields smaller geometric errors with respect to isoAdvector-plicRDF, between 5% and 35% smaller for all mesh types and resolutions, except for the coarsest and finest unstructured polyhedral mesh, where the geometric error slightly increases. The volume conservation and boundedness errors remain of the same order of magnitude, both with small improvements for distinct mesh types and resolutions. In terms of efficiency, the improvement is more evident: throughout all the tests, mesh types and resolutions, both advection and reconstruction times are systematically decreased with respect to isoAdvector-plicRDF, except for the finer unstructured polyhedral meshes, where the advection takes a slightly greater time. Despite of this, the total time consumed by isoAdvector-CLCIR is between 8% and 46% lower than that consumed by isoAdvector-plicRDF.

This behavior can be partially explained looking at the bounding procedure of isoAdvector. Table 4.6 shows, per time step and for different mesh types with $n = 64$, the number of interfacial cells, $\tilde{n}_{c,\text{int}}$; the number of cells in which the bounding procedure is applied, $\tilde{n}_{c,\text{bound}}$; the ratio of these cells with respect to the total number of cells, in percentage; and the number of iterations carried out by the bounding step until all cells are bounded or the maximum number of iterations is reached, $\tilde{n}_{iter,\text{bound}}$. For hexahedral meshes, the $\tilde{n}_{c,\text{bound}}$ given by isoAdvector-CLCIR is very low, about 78 times lower. The average number of iterations of the bounding procedure is also lower in isoAdvector-CLCIR, 2.4/2.8 iterations per time step against the 4.8 given by isoAdvector-plicRDF. Therefore, the time consumed in the advection step is reduced. For tetrahedral meshes, the $\tilde{n}_{c,\text{bound}}$ is slightly lower in isoAdvector-CLCIR, whereas $\tilde{n}_{iter,\text{bound}}$ is the same for both combinations, which corresponds to the specified maximum number of iterations. For unstructured polyhedral meshes, isoAdvector-CLCIR yields a greater $\tilde{n}_{c,\text{bound}}$. This does not increase the advection time for the considered mesh size, but for $n = 128$ it has a greater influence, as can be seen in tables 4.4 and 4.5.

The time consumed by CLCIR in this implementation is very similar to that in the FMFPA-CLCIR combination for tetrahedral meshes, whereas it is higher for hexahe-

Table 4.4: Same results as in Table 4.2, but including the results obtained with the new combination of the isoAdvector-CLCIR methods.

Methods	n	E_g	$\mathcal{O}$	E_{vol}	E_{bound}	$\tilde{t}_{\text{tot}}$	$\tilde{t}_{\text{adv}}$	$\tilde{t}_{\text{rec}}$
Hexahedral meshes								
FMFPA-CLCIR	32	6.485e-03	-	5.204e-17	1.274e-18	1.94	1.78	0.16
isoAdvector-plicRDF	32	7.966e-03	-	8.726e-16	2.170e-17	1.00	0.18	0.82
isoAdvector-CLCIR	32	7.162e-03	-	1.082e-15	2.915e-17	0.47	0.15	0.32
FMFPA-CLCIR	64	2.070e-03	1.65	7.980e-17	8.008e-19	9.90	9.16	0.75
isoAdvector-plicRDF	64	3.018e-03	1.40	5.938e-15	2.015e-09	5.57	1.43	4.14
isoAdvector-CLCIR	64	2.376e-03	1.59	1.029e-15	2.517e-18	2.76	1.07	1.69
FMFPA-CLCIR	128	4.308e-04	2.26	1.299e-08	7.201e-10	56.84	53.18	3.66
isoAdvector-plicRDF	128	7.063e-04	2.10	1.878e-14	1.603e-10	35.64	12.57	23.06
isoAdvector-CLCIR	128	4.844e-04	2.29	9.298e-16	7.251e-12	18.50	10.56	7.93
FMFPA-CLCIR	256	6.131e-05	2.81	9.284e-15	2.926e-19	374.91	353.03	21.88
isoAdvector-plicRDF	256	1.061e-04	2.74	1.720e-13	1.924e-11	241.13	108.29	132.85
isoAdvector-CLCIR	256	7.185e-05	2.75	1.150e-13	2.168e-11	133.75	90.22	43.53
Tetrahedral meshes								
FMFPA-CLCIR	32	1.326e-02	-	6.980e-06	1.123e-08	17.56	17.03	0.54
isoAdvector-plicRDF	32	1.394e-02	-	1.298e-15	4.887e-06	1.88	0.23	1.65
isoAdvector-CLCIR	32	1.314e-02	-	1.402e-15	3.936e-06	0.63	0.17	0.46
FMFPA-CLCIR	64	4.231e-03	1.65	3.290e-06	3.054e-10	77.90	75.07	2.83
isoAdvector-plicRDF	64	5.858e-03	1.25	2.475e-15	1.181e-06	19.68	2.90	16.78
isoAdvector-CLCIR	64	4.763e-03	1.46	5.959e-15	9.071e-07	4.43	1.57	2.86
FMFPA-CLCIR	128	9.386e-04	2.17	3.346e-07	2.106e-10	396.65	379.87	16.79
isoAdvector-plicRDF	128	1.464e-03	2.00	4.672e-15	1.263e-07	126.15	23.44	102.71
isoAdvector-CLCIR	128	1.218e-03	1.97	2.261e-14	1.549e-07	34.54	16.86	17.67
FMFPA-CLCIR	256	1.583e-04	2.57	2.157e-07	9.493e-11	2694.72	2563.02	131.70
isoAdvector-plicRDF	256	3.621e-04	2.02	2.618e-15	1.872e-08	1121.66	244.59	877.07
isoAdvector-CLCIR	256	3.496e-04	1.80	3.254e-14	1.818e-08	387.61	241.59	146.01
Unstructured polyhedral meshes								
FMFPA-CLCIR	32	1.052e-02	-	2.516e-07	8.873e-09	56.71	54.00	2.71
isoAdvector-plicRDF	32	1.013e-02	-	6.523e-16	4.481e-07	9.12	3.02	6.10
isoAdvector-CLCIR	32	1.119e-02	-	3.643e-17	4.471e-07	8.17	2.78	5.38
FMFPA-CLCIR	64	3.570e-03	1.56	3.121e-09	4.467e-09	279.05	258.57	20.49
isoAdvector-plicRDF	64	4.173e-03	1.28	1.631e-16	5.486e-08	162.03	38.97	123.06
isoAdvector-CLCIR	64	3.904e-03	1.52	9.038e-16	7.127e-08	58.44	29.12	29.32
FMFPA-CLCIR	128	6.226e-04	2.52	4.321e-08	2.376e-08	1 488.23	1 347.43	140.80
isoAdvector-plicRDF	128	6.945e-04	2.59	2.116e-16	8.783e-09	801.88	275.46	526.42
isoAdvector-CLCIR	128	6.945e-04	2.49	1.528e-14	7.748e-09	490.58	321.01	169.58

Table 4.5: Same results as in Table 4.3, but including the results obtained with the new combination of the isoAdvector-CLCIR methods.

Methods	n	E_g	$\mathcal{O}$	E_{vol}	E_{bound}	$\tilde{t}_{\text{tot}}$	$\tilde{t}_{\text{adv}}$	$\tilde{t}_{\text{rec}}$
Hexahedral meshes								
FMFPA-CLCIR	32	3.509e-03	-	7.027e-07	1.173e-08	1.90	1.74	0.17
isoAdvector-plicRDF	32	4.485e-03	-	9.177e-16	1.515e-09	1.00	0.20	0.80
isoAdvector-CLCIR	32	3.808e-03	-	6.158e-16	1.866e-17	0.46	0.13	0.33
FMFPA-CLCIR	64	9.686e-04	1.86	8.530e-08	8.896e-10	9.95	9.15	0.80
isoAdvector-plicRDF	64	1.276e-03	1.81	5.586e-15	3.138e-10	5.77	1.56	4.21
isoAdvector-CLCIR	64	1.076e-03	1.82	5.548e-15	6.096e-11	2.83	1.30	1.53
FMFPA-CLCIR	128	2.372e-04	2.03	1.028e-08	6.932e-11	60.02	55.79	4.23
isoAdvector-plicRDF	128	3.240e-04	1.98	2.839e-14	2.259e-10	55.46	19.89	35.57
isoAdvector-CLCIR	128	2.657e-04	2.02	2.586e-14	7.731e-11	20.73	12.92	7.81
FMFPA-CLCIR	256	5.421e-05	2.13	1.078e-09	4.649e-12	406.33	379.48	26.84
isoAdvector-plicRDF	256	7.690e-05	2.07	8.065e-14	5.695e-11	282.14	125.36	156.77
isoAdvector-CLCIR	256	6.180e-05	2.10	6.664e-14	8.711e-12	142.90	99.13	43.77
Tetrahedral meshes								
FMFPA-CLCIR	32	6.184e-03	-	1.900e-05	1.195e-08	13.81	13.44	0.37
isoAdvector-plicRDF	32	8.680e-03	-	6.852e-16	4.118e-06	2.21	0.28	1.93
isoAdvector-CLCIR	32	7.328e-03	-	6.939e-18	3.556e-06	0.55	0.17	0.38
FMFPA-CLCIR	64	1.537e-03	2.01	6.499e-07	3.979e-10	60.34	58.00	2.34
isoAdvector-plicRDF	64	2.857e-03	1.60	1.952e-15	6.250e-07	16.27	2.85	13.41
isoAdvector-CLCIR	64	2.232e-03	1.72	1.025e-15	4.903e-07	4.23	1.66	2.56
FMFPA-CLCIR	128	4.032e-04	1.93	8.923e-09	8.491e-11	316.24	302.71	13.53
isoAdvector-plicRDF	128	9.290e-04	1.62	4.118e-15	5.857e-08	130.93	26.14	104.80
isoAdvector-CLCIR	128	7.386e-04	1.60	1.101e-14	5.592e-08	33.91	17.98	15.93
FMFPA-CLCIR	256	1.041e-04	1.95	1.941e-09	5.332e-11	1 771.46	1 686.99	84.47
isoAdvector-plicRDF	256	3.344e-04	1.47	7.615e-16	3.854e-09	1 222.58	270.78	951.80
isoAdvector-CLCIR	256	3.022e-04	1.29	3.532e-14	4.172e-09	396.62	256.56	140.06
Unstructured polyhedral meshes								
FMFPA-CLCIR	32	5.348e-03	-	9.613e-08	2.795e-09	45.60	41.50	4.10
isoAdvector-plicRDF	32	5.934e-03	-	5.274e-16	4.262e-07	10.04	2.99	7.05
isoAdvector-CLCIR	32	5.468e-03	-	1.717e-16	3.955e-07	7.23	2.63	4.60
FMFPA-CLCIR	64	1.569e-03	1.77	4.565e-09	1.917e-10	221.74	201.94	19.80
isoAdvector-plicRDF	64	1.644e-03	1.85	3.990e-16	4.765e-08	197.73	53.18	144.55
isoAdvector-CLCIR	64	1.600e-03	1.77	1.044e-15	5.365e-08	57.00	30.93	26.07
FMFPA-CLCIR	128	4.176e-04	1.91	2.720e-08	3.310e-08	1 413.17	1 252.91	160.26
isoAdvector-plicRDF	128	4.225e-04	1.96	9.147e-15	4.617e-09	869.30	291.17	578.13
isoAdvector-CLCIR	128	4.382e-04	1.87	2.189e-14	4.891e-09	462.29	308.87	153.42

Table 4.6: Average number of cells and iterations per time step given by the bounding procedure of isoAdvector when coupled with plicRDF and CLCIR. Data obtained in the 3D deformation flow test for different mesh types with $n = 64$.

Mesh type	Methods	$\tilde{n}_{c,\text{int}}$	$\tilde{n}_{c,\text{bound}}$	$\tilde{n}_{c,\text{bound}}/n_c$ [%]	$\tilde{n}_{iter,\text{bound}}$
Hexahedral	isoAdvector-plicRDF	3688.5	753	0.287	4.8
	isoAdvector-CLCIR	3732.5	9.7	0.004	2.4
Tetrahedral	isoAdvector-plicRDF	4985.7	1128	0.430	5
	isoAdvector-CLCIR	5541.1	1039.6	0.397	5
Unstructured polyhedral	isoAdvector-plicRDF	3536.4	1285.9	1.345	5
	isoAdvector-CLCIR	4004.3	4735.7	1.807	5

dral and unstructured polyhedral meshes. This might be caused by the complexity of the truncated polyhedra obtained when cutting a hexahedral or polyhedral cell, since the number of its faces and points is usually greater in this kind of cells compared to that of tetrahedral cells, in which always two polyhedra with 4 faces are formed. Therefore, the time spent by copying the arrays containing the information of the truncated polyhedra, operations needed to communicate the `FORTRAN` libraries with the `C++` implementation, is greater in these type of meshes. Besides, as the number of interfacial cells per time step is also greater in isoAdvector-CLCIR than in FMFPA-CLCIR, the time consumed in the reconstruction increases.

Some comments regarding the advection methods can be extracted at the light of the results presented in tables 4.4 and 4.5. Considering that the error introduced by the reconstruction step is approximately the same for both methods (note that $\tilde{n}_{c,\text{int}}$ depends on the method for a given F_{tol} and the total error introduced can vary if $\tilde{n}_{c,\text{int}}$ is different), the remaining differences between FMFPA-CLCIR and isoAdvector-plicRDF in the measured errors are mainly due to the error yielded by the advection algorithms. Thus, taking into account the geometric errors, the FMFPA method seems to be slightly more accurate than isoAdvector, although it yields volume conservation errors several orders of magnitude higher in tetrahedral and unstructured polyhedral meshes. In this type of meshes, the velocity fields given in the tests are not solenoidal and, therefore, the conservation errors increase in both methods but, in isoAdvector, the redistribution algorithm helps to keep the volume error very low. The efficiency of the isoAdvector method is higher, since the operations carried out at the cell faces are much less computationally expensive than the truncation operations performed by FMFPA.

Chapter 5

Results and discussion

In this chapter, several static and dynamic tests are presented in order to asses the accuracy and robustness of the proposed contact line force model (CLFM) on a wide range of Reynolds and Weber numbers. The results are compared with experimental data obtained by other authors and with numerical results obtained with different contact angle models, which are based on solely applying as a boundary condition either the equilibrium contact angle (ECA-BC) or the dynamic contact angle (DCA-BC) given by one of the models described in Section 1.3. Also, the results of the 3D numerical simulation of a splashing drop are presented and compared with experimental results.

5.1 Axisymmetric drop impact

5.1.1 Drop released on a wall under zero-gravity conditions

A first assessment of the implemented CLFM is carried out through a test in which a "semi-spherical" drop of initial diameter D_0 is placed at rest on a flat surface, with an initial contact angle of 90°, in a zero-gravity environment [141]. The only forces acting are surface tension and the intermolecular forces between the wall and the fluids (characterized by the contact line force), which deform the drop until the equilibrium situation is reached. The shape of the drop in equilibrium calculated from volume conservation corresponds to a spherical cap of diameter is given by

$$D = D_0 \left[(1 - \cos\theta_e)(1 + \sin^2\theta_e - \cos\theta_e)/2\right]^{-1/3}, \tag{5.1}$$

height

$$h = D/2(1 - \cos\theta_e), \tag{5.2}$$

and radius of the base

$$r_b = D\sin\theta_e, \tag{5.3}$$

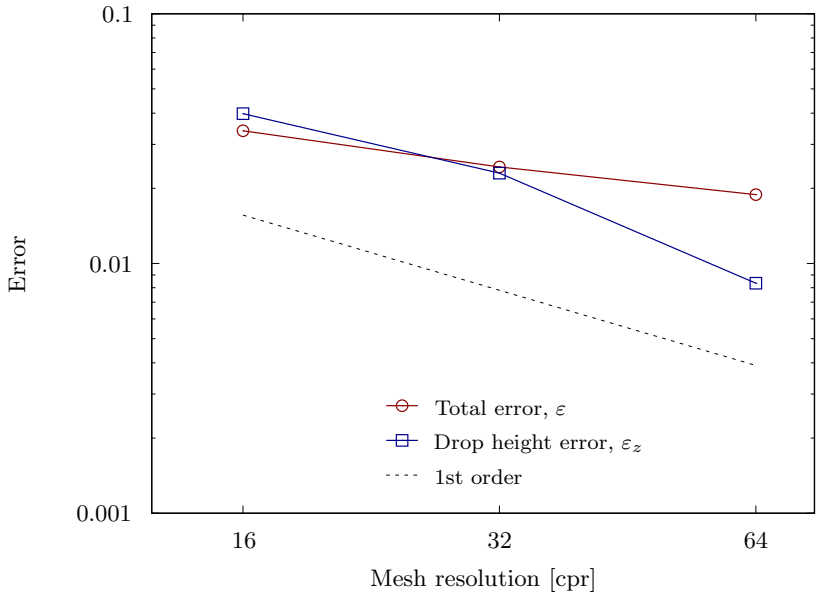


Figure 5.1: Test of a drop released on a wall. Errors for the drop shape at equilibrium as a function of mesh resolution, for an equilibrium contact angle of 60°.

where θ_e is the static contact angle at equilibrium (therefore, $\theta_{d,i} = \theta_e$ in Eq. (3.53)). The test is performed for a water drop ($\rho_l = 1000\ \mathrm{kg\,m^{-3}}$, $\mu_l = 1\,\mathrm{mPa\,s}$, $\sigma = 72\ \mathrm{mN\,m^{-1}}$), of initial diameter $D_0 = 1\,\mathrm{mm}$, surrounded by air ($\rho_g = 1.25\ \mathrm{kg\,m^{-3}}$, $\mu_g = 0.00182\ \mathrm{mPa\,s}$), on a computational domain of wedge shape with a meridian plane of size $2D_0 \times 2D_0$. Five different values of the equilibrium contact angle are considered: 30°, 60°, 90°, 120°, and 150°.

A mesh dependence analysis was first carried out. The total error between the theoretical and simulated shapes of the drop at equilibrium (the numerical results correspond to a long enough time of 100 ms) is defined as

$$\varepsilon = (\varepsilon_r + \varepsilon_z)/2, \tag{5.4}$$

where $\varepsilon_r = |r_b - r|/r_b$ and $\varepsilon_z = |h - z|/h$ are the errors for the radius of the base and height of the droplet, respectively, and z and r are measured from the 0.5-isosurface contour. Figure 5.1 shows the error convergence as a function of mesh resolution (cells per initial drop radius, cpr) for an equilibrium contact angle of 60°. Note that the convergence is first order for the drop height error, while the poor convergence of error ε_r, which remains nearly constant with mesh resolution, causes the convergence of the total error to deviate from the first order.

Figure 5.2 shows the numerical results for the equilibrium drop shape (0.5-isosurface) at the end of the simulation, when the final form has already been reached, obtained with a mesh of size 128×128 (32 cpr). Figure 5.2(a) depicts a comparison between the numerical results obtained for different static contact angles and the analytical solution. It can be seen how the contact angle obtained from the simulation better

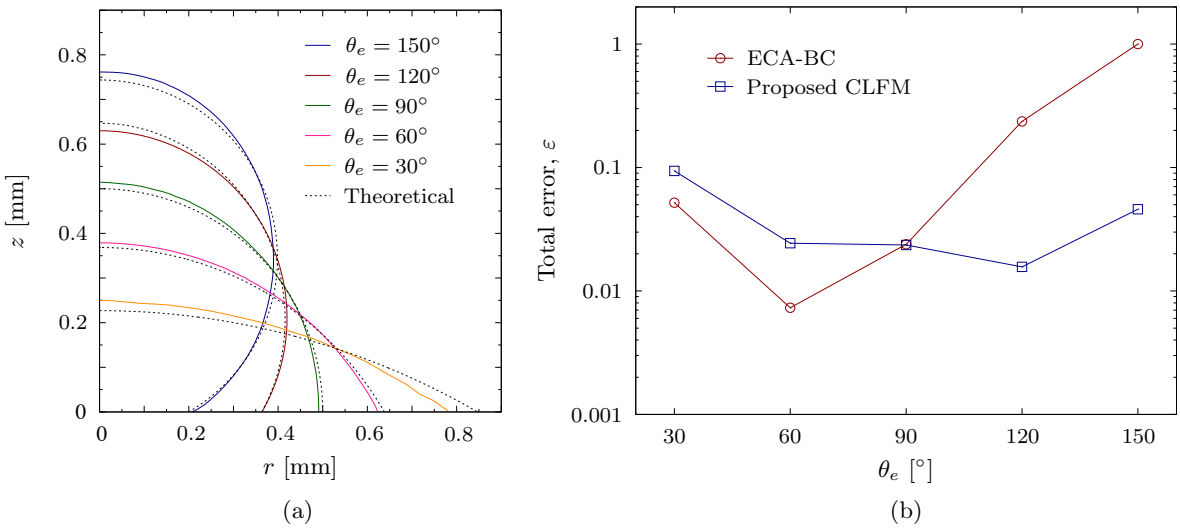


Figure 5.2: Numerical results obtained for the equilibrium drop shape, for different equilibrium contact angles and a mesh resolution of 32 cpr. (a) Comparison with theoretical shapes (dashed lines) and (b) total error, ε, as a function of θ_e.

matches that corresponding to the theoretical shape of the droplet as the value of the equilibrium contact angle increases except for high values of this angle. Figure 5.2(b) shows a comparison of the results for the total drop shape error obtained with the proposed CLFM and with solely applying the equilibrium contact angle as a boundary condition (ECA-BC), for different equilibrium contact angles. Note that the error obtained with CLFM is larger for equilibrium contact angles below 90°, and nearly equal to that obtained with ECA-BC for $\theta_e = 90°$, since for this angle, which coincides with the contact angle at drop release, the contact line force is zero. The error obtained with ECA-BC increases dramatically for higher equilibrium contact angles, and the drop even detaches from the solid surface when $\theta_e = 150°$, while with CLFM the error turns out to be of a similar order for the whole range of contact angles considered, and remains relatively low also for high equilibrium contact angles.

5.1.2 Drop depositions

To validate the proposed CLFM under dynamic conditions, several drop impacts onto solid surfaces have been simulated for a wide range of Re and We numbers and different wetting properties. For the sake of simplicity, a 2D axisymmetric domain will be used in these tests. Table 5.1 summarizes the conditions of the different impacts simulated. The air properties are assumed to be $\rho_g = 1.25\ \mathrm{kg\,m^{-3}}$ and $\mu_g = 0.0182\ \mathrm{mPa\,s}$.

In these tests, the boundary conditions, discretization schemes, and solution algorithms are kept the same. For the discretization of gradient terms, the Gauss scheme is used. For the convective term, the second-order upwind (LUD) scheme is used, which uses an explicit correction based on the local cell velocity gradient. In the volume fraction transport equation, a TVD scheme with a van Leer limiter function is used

Table 5.1: Physical properties and drop impact conditions.

Test	D_0 [mm]	U [m s^{-1}]	σ [mN m^{-1}]	Re	We	θ_e [°]	θ_a [°]	θ_r [°]	Ref.
1	1.5	0.93	20.14	2 455	43	32	-	-	[20]
2	2.5	0.23	73	575	1.81	85.5	120	65	[137]
3	2.45	4.1	63	106	802	15.1	17	13	[153]
4	2.45	1.64	73	4 010	90	99.7	105	95	[153]
5	2	1	70	2 334	27	57	110	30	[120]

for the convection term, and a central differencing scheme for the artificial compression term. For the temporal discretization, the first-order implicit Euler scheme is chosen. For the resolution of the Navier-Stokes equations, the PISO algorithm is used with 3 corrections. The condition $\mathrm{CFL}_{\max} < 0.2$ is set for the calculation of the variable time step. The GAMG solver with a DIC smoother is used for the pressure and velocity terms, whereas the DILU smoother is preferred for the volume fraction terms.

Regarding the boundary conditions, at the solid boundary, the no-slip condition is applied for the velocity and the pressure gradient is set to zero. At open boundaries, i.e., top and lateral planes, the total pressure is set to zero and the static pressure is calculated from the dynamic pressure using the velocity at the boundary, which is obtained from the continuity equation if there is inflow and from the zero velocity gradient condition if there is outflow. At the two planes normal to the azimuthal direction, vectorial quantities are rotated, preserving their magnitude, and scalar quantities are linearly interpolated. On the axis of symmetry, the pressure and velocity gradients are set to zero. The velocity field is initialized by imposing the corresponding velocity vector $(0, 0, -U)$ at cells whose volume fraction is greater than 0.

Test 1 is an impact of a heptane drop on a stainless steel surface [20]. An axisymmetric computational domain of size $2.5D_0 \times 2.5D_0$ is used, where the center of the droplet is initially set at a distance of $3D_0/2$ from the impact surface. Figure 5.3 shows a comparison between the numerical results for the evolution of the spread factor using a mesh of 100 cpr. Two sets of simulations are presented: one using the proposed CLFM and the other using the equilibrium contact angle boundary condition (ECA-BC). Note that only the equilibrium contact angle value is available ($\theta_e = 32°$) and there is no data on wettability hysteresis. To show the sensitivity of the results to the value of θ_e used in the simulations, we compare in Fig. 5.3, as an example, the results obtained for $\theta_e = 32°$ with those corresponding to a value not much different, of 40°. Note that small differences in θ_e produce appreciable differences in the results. In any case, given that the accuracy of the experimental value of θ_e used is unknown and considering that wettability hysteresis is not being taken into account, the differences are not too significant except perhaps towards the end of the spreading process, when the effect of θ_e is more appreciable. Figure 5.4(a) shows the dependence on the mesh size of the numerical results of Fig. 5.3 for $\theta_e = 40°$. Note that the best agreement with the experimental results is observed for the results obtained using the proposed CLFM

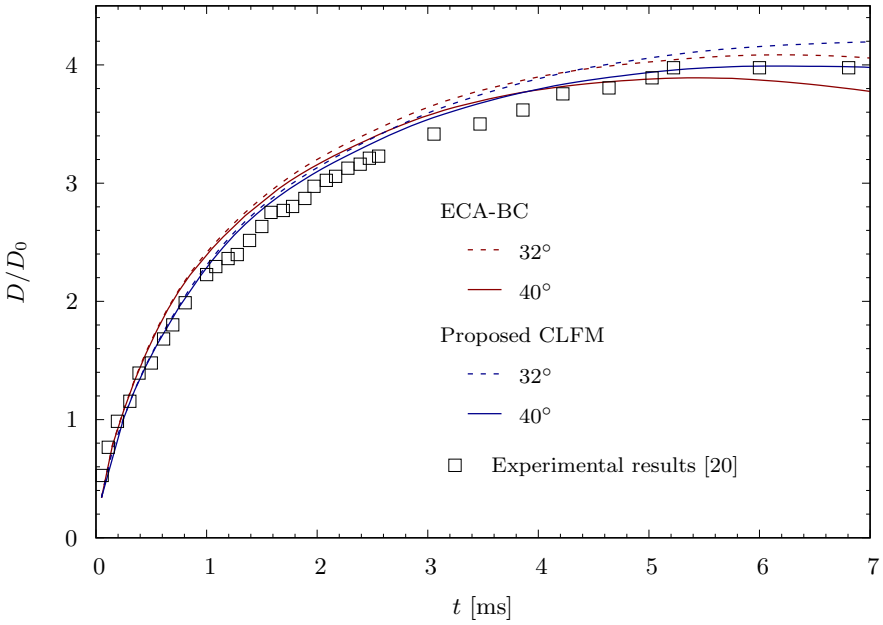


Figure 5.3: Evolution of the spread factor for Test 1 in Table 5.1. Results obtained with the proposed CLFM and with solely the boundary condition applied at the contact line based on the equilibrium contact angle (ECA-BC) for two different values of this angle. Comparison with the experimental results of Bussmann et al. [20].

and a mesh size of 100 cpr, as more quantitatively depicted in Fig. 5.4(b), which shows the convergence of the radial error, ε_r, as a function of the mesh resolution. This error is calculated, at $t = 7$ ms, as $\varepsilon_r = |r_e - r|/r_e$, where r_e is the experimental radius of the lamella. The convergence for the contact line force model is now closer to 2nd order than in the test of Fig. 5.2, whereas the solution does not converge when ECA-BC is used.

In the remaining tests (2 to 5) shown in Table 5.1, the numerical results obtained with the proposed CLFM are compared with those obtained using different DCA models to solely impose a contact angle dependent on the contact line velocity u_{cl} as a boundary condition at the contact line. For this purpose, several DCA models have been implemented in OpenFOAM, namely the Cox, Kistler, Shikhmurzaev and Jiang models described in Section 1.3. The results will also be compared with those obtained using the equilibrium contact angle to impose a boundary condition, without taking into account wettability hysteresis or any other dynamic effect on the contact angle. In all simulations, we used a computational domain of size $2D_0 \times 2D_0$, in which the drop center was set at a distance of $2D_0/3$ from the impact surface, and, unless otherwise stated, a mesh resolution of 50 cpr.

Test 2 corresponds to an impact of a water drop on a stainless steel substrate. Surface wettability for water is experimentally characterized by the advancing and receding contact angles $\theta_a = 120°$ and $\theta_r = 65°$, respectively. From these two values, an equilibrium contact angle $\theta_e = 85.5°$ was determined using the correlation proposed

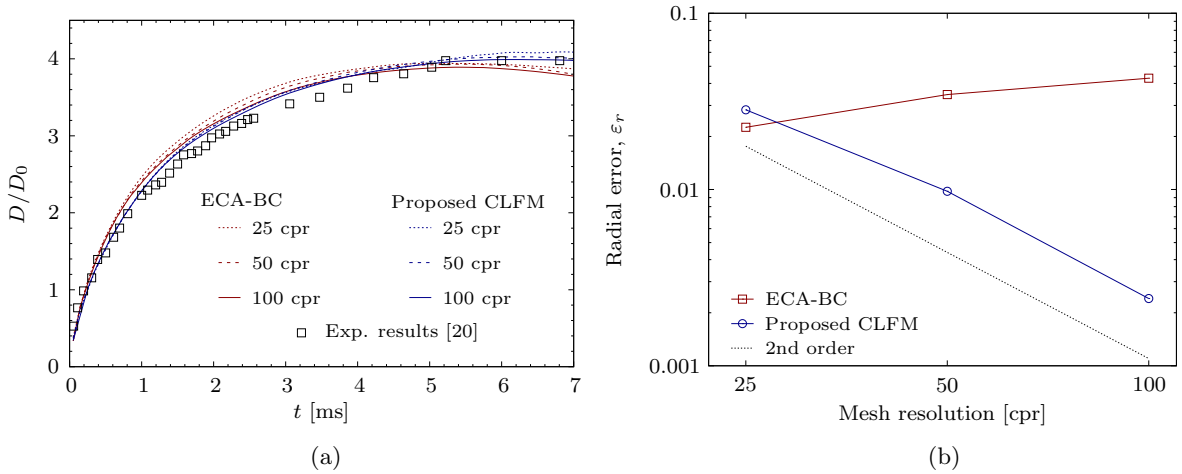


Figure 5.4: (a) Same results as in Fig. 5.3 (numerical results for $\theta_e = 40°$) and three different mesh sizes. (b) Radial spreading error as a function of mesh resolution.

by Tadmor [161]. The three values of θ_a, θ_r and θ_e are introduced into the DCA model used in CLFM to reproduce the effect of surface wettability hysteresis.

Figure 5.5 shows the time evolution of the spread factor predicted by the proposed CLFM and the alternative approach based on just imposing a boundary condition at the contact line with different values for the contact angle. Note that the results of CLFM agree with the experimental results appreciably better than those obtained using any of the four DCA models considered or the equilibrium contact angle to impose the boundary condition. CLFM is able to reproduce the drop receding after $t \approx 10$ ms better than the other models, thus yielding a better estimate of the spread factor, closer to the experimental data. Figure 5.6 shows that the radial error obtained with CLFM is substantially lower than those obtained with the other models during most of the time of the spreading process.

Test 3 is an impact of a glycerin drop on a glass substrate. Figure 5.7 shows the spread factor obtained with the different contact line models considered as a function of the dimensionless time $t^* = tU/D_0$, along with the experimental results reported in [153]. In the inertial regime, and before $t^* \approx 1$, all models predict the same evolution of the lamella diameter except the ECA-BC, which oscillates around the experimental data due to bubble entrapment at the advancing front rim. Subsequently, CLFM and Kistler DCA-BC, which in this case give results that are in close agreement with each other, provide the results that best fit the experimental data. Note, however, that the Shikhmurzaev DCA model, when used to impose the boundary condition on the contact line, and the proposed CLFM based on the same Shikhmurzaev model, whose results are also in this case in close agreement with each other, predict better than the Kistler model the behavior observed in the experiments that the contact line tends to stop, without receding.

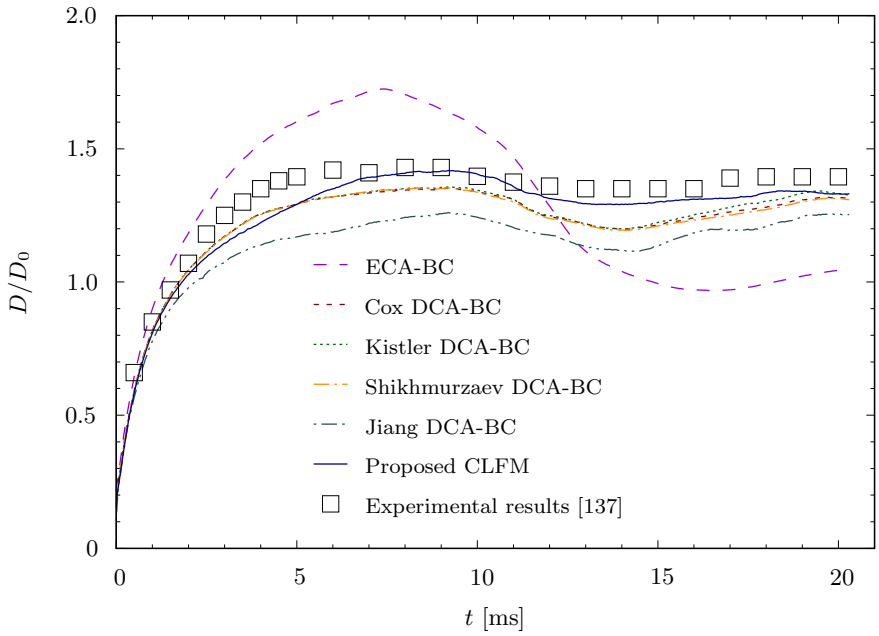


Figure 5.5: Evolution of the spread factor predicted by the proposed contact line force model, and by just imposing a boundary condition based on the equilibrium contact angle (ECA-BC) and several DCA models (DCA-BC), for Test 2. Comparison with experimental results.

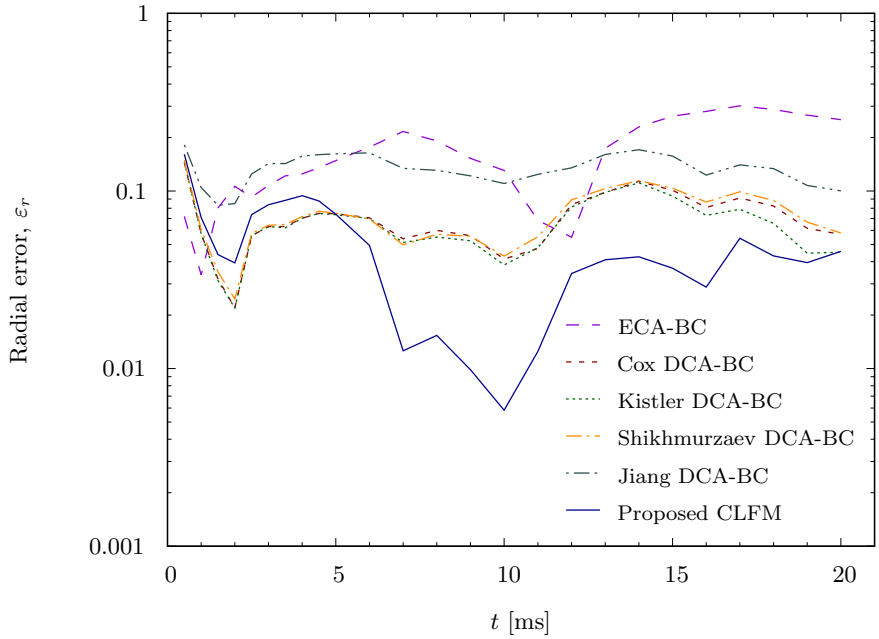


Figure 5.6: Evolution of the radial error obtained in the simulations of Fig. 5.5, for Test 2.

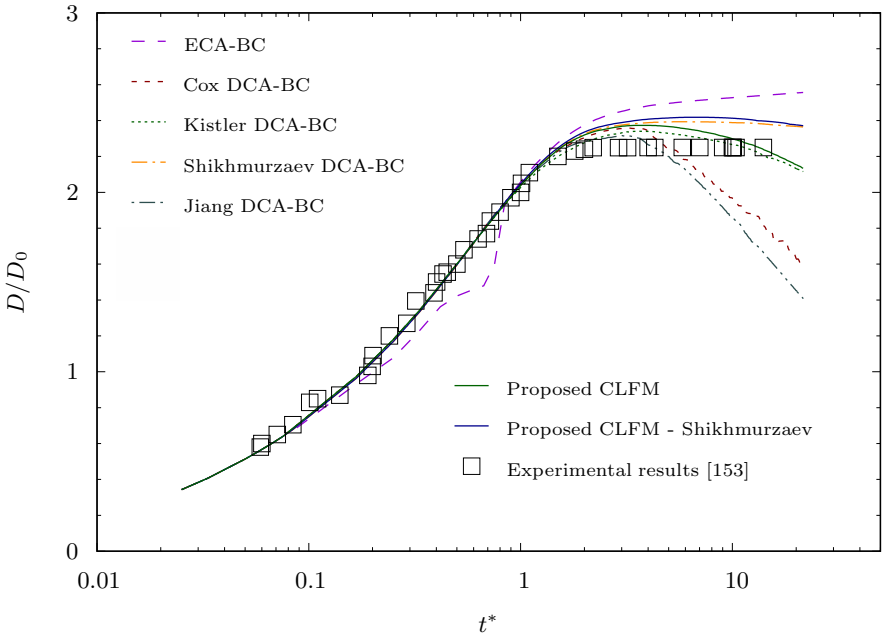
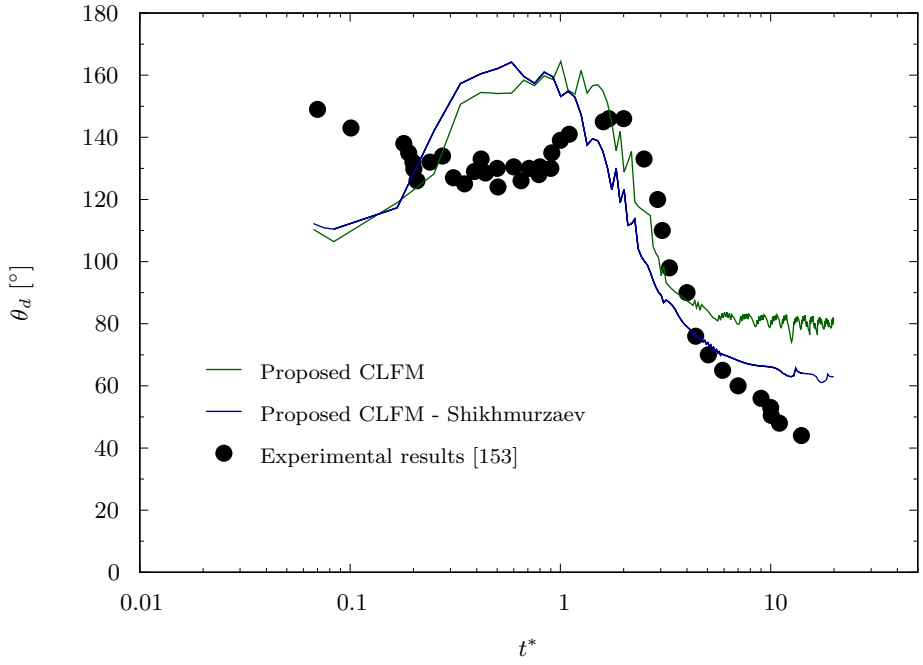
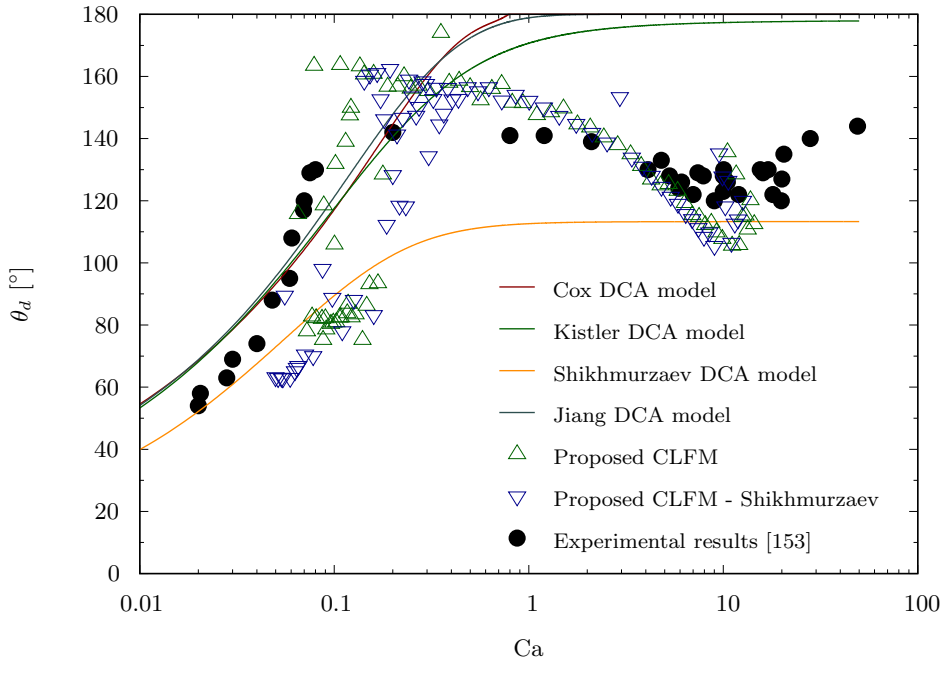


Figure 5.7: Results as in Fig. 5.5, for the drop impact of Test 3.

This better behavior at the end of the spreading process can be partly explained in view of Figs. 5.8(a) and 5.8(b), which show the results for the dynamic contact angle obtained with the proposed CLFM as a function of the dimensionless time and capillary number, respectively. It can be observed from Fig. 5.8(a) how in the first instants after impact the contact angles obtained numerically are below the experimental results. In later instants, the numerical results reach a maximum that is observed to be delayed in the experimental data. Later, in the interval $0.6 \lesssim t^* \lesssim 4$, the numerical results agree better with experimental data, although some oscillations are observed in the numerical results. Also for $t^* \gtrsim 4$, the numerical results for the Kistler model in the CLFM begin to oscillate around a mean value near 80°, and do not reproduce the absolute minimum observed in the experiments. This disagreement could be expected, since the measured receding contact angle is much higher than the corresponding static value, θ_r, used in the Kistler model inside CLFM. The perturbations in the dynamic contact angle, especially after $t^* \approx 4$, might be partly due to the very high ratio between the viscosities of the two fluids: $\mu_l/\mu_g \approx 6.5 \times 10^3$ for this test compared to $\mu_l/\mu_g \approx 55$ for Test 2, which makes the PISO algorithm unstable, thus requiring an increase in the number of iterations used to converge (from 3 to 20). They might also be related to possible difficulties in locating the contact line and determining the orientation of the interface on the wall when the contact angle is small. On the other hand, the results for $t^* \gtrsim 4$ obtained when CLFM is used with the Shikhmurzaev model show a better agreement with the experimental data up to $t^* \approx 6$, when the dynamic contact angle begins to stabilize around a mean value of 60°, still larger than the measured minimum. This improvement might be due to a better prediction of the



(a)



(b)

Figure 5.8: Evolution of the dynamic contact angle as a function of (a) the dimensionless time and (b) capillary number (lines correspond to different dynamic contact angle (DCA) models) obtained numerically in Test 3.

contact angle for low Ca numbers, reached when the contact line is almost stopped, as can be seen in Figure 5.8(b) for Ca $\lesssim 0.1$.

Figure 5.8(b) also shows the reasonable overall agreement between the dynamic contact angle obtained using the proposed CLFM with two different DCA models and the experimental data up to Ca ≈ 10, while the empirical correlations proposed in [30, 70, 75] (Eqs. (1.10), (1.7) and (1.9), respectively) fit the experimental results reasonably well only for small capillary numbers (Ca $\lesssim 0.3$). The computed θ_d values underpredict the experimental ones up to Ca ≈ 0.2 and are above them in an approximate range of Ca between 0.2 and 2, where the maximum values are reached. The decreasing trend of the experimental data for θ_d after Ca ≈ 1 and the subsequent increase for Ca $\gtrsim 10$ are also reasonably well reproduced. For higher capillary numbers (Ca $\gtrsim 20$), there are no simulation data since the contact line velocity does not exceed 11 $\mathrm{m\,s^{-1}}$ at any time. Despite the above mentioned differences, the overall agreement with the experimental data found in Test 3 is reasonably good.

Test 4 corresponds to the impact of a water drop on a wax substrate. It can be observed from Fig. 5.9 that, due to the higher Re number compared to the previous tests, all models are close to the experimental data, with small differences between them, until $t^* \approx 0.4$, when Jiang DCA-BC and ECA-BC begin to overpredict more clearly the diameter of the spreading lamella. Note that CLFM is the model that better predicts the evolution of the spread after $t^* \approx 0.7$, particularly the maximum spreading, whereas the Cox, Kistler and Shikhmurzaev models predict a delay of the maximum spreading instant and then a receding stage that is considerably slower than that observed in the experiments and predicted by CLFM (note that t^* is in logarithmic scale).

Test 5 corresponds to the impact of a drop on a stainless steel surface. The drop liquid is water with sodium dodecyl sulfate at a concentration of 100 ppm by weight. The surfactant addition results in a reduction of pure water viscosity ($\mu_l = 0.85$ mPa s) and density ($\rho_l = 945\ \mathrm{kg\,m^{-3}}$), a surface tension $\sigma = 70\ \mathrm{mN\,m^{-1}}$ and a wettability characterized by the empirical contact angles shown in Table 5.1 [120]. Figure 5.10 shows the numerical results for the evolution of the spread factor. It can be observed from the figure that, as in some of the previous tests, the results from ECA-BC and Jiang DCA-BC deviate from the experimental results with a very considerable error from the early stages of the impact. On the other hand, the rest of the models agree better with the experimental data, but still with a radial error of between 10 and 15%. Although the differences between the remaining models are small until $t = 5$ ms, after that instant, CLFM approaches better the experimental data, whereas Cox and Kistler models yield a higher radial error. The results obtained with CLFM and Kistler and Shikhmurzaev models for the spreading drop diameter at the last instant simulated are very close to the experimental value.

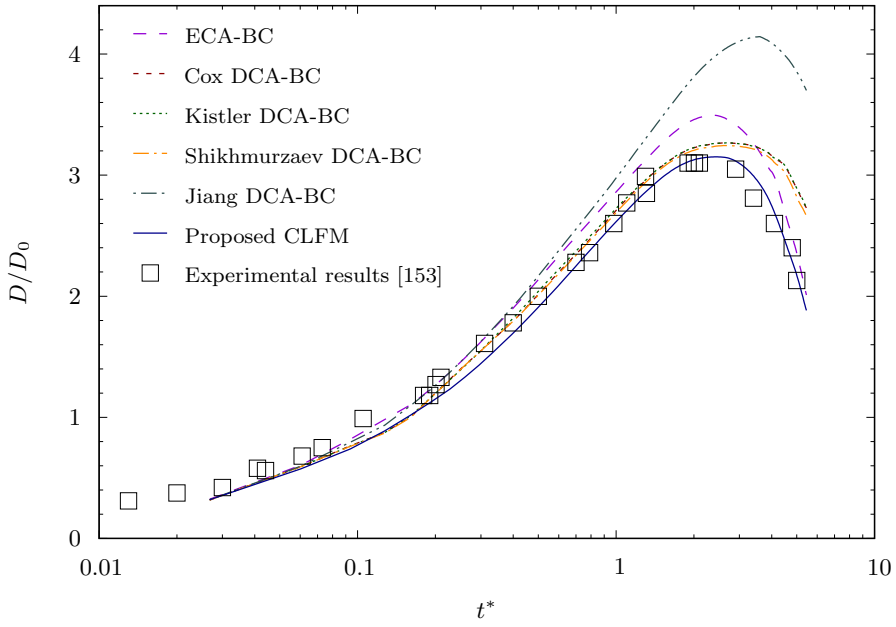


Figure 5.9: Evolution of the spread factor in Test 4. Comparison between numerical results obtained with different contact line models and experimental results.

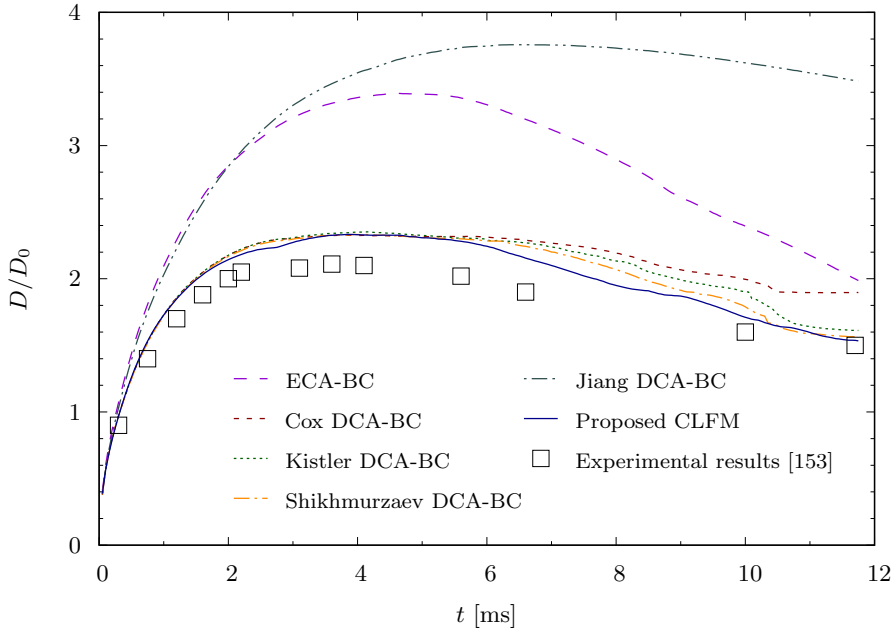


Figure 5.10: Results as in Fig. 5.5, for Test 5.

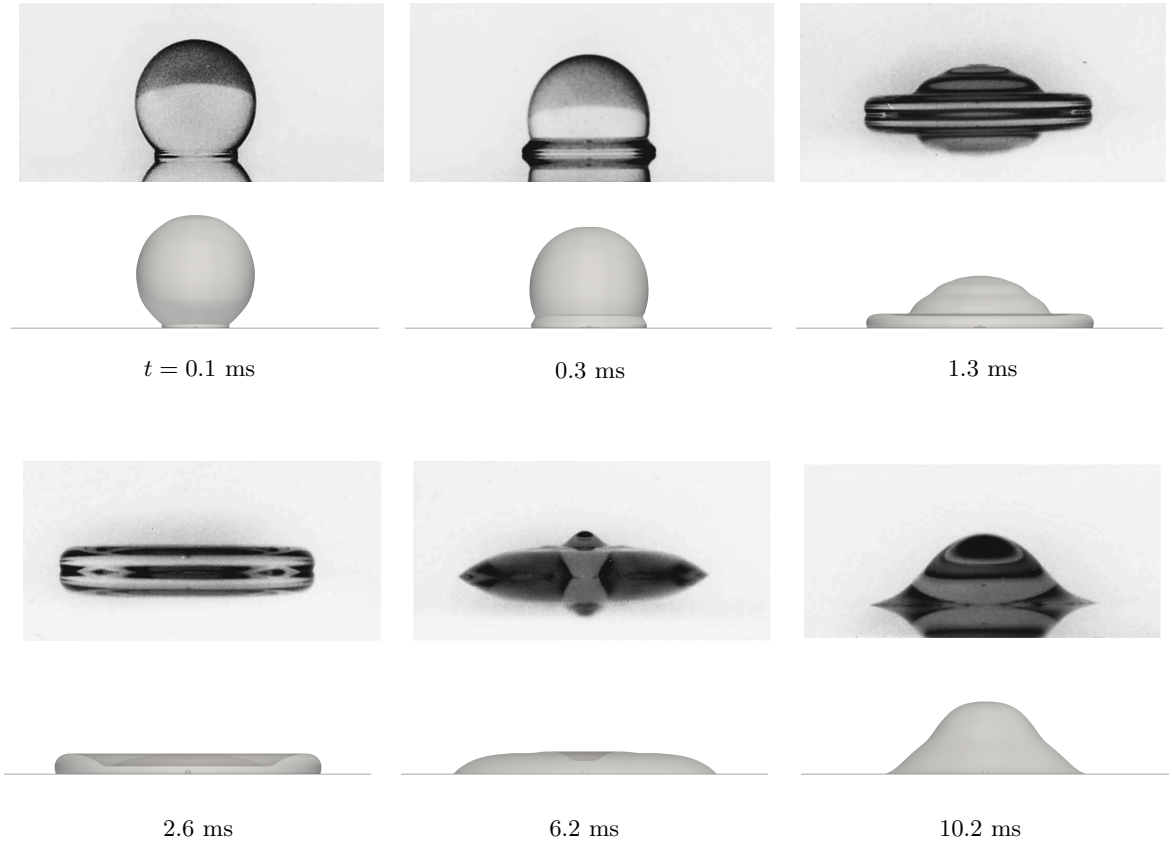


Figure 5.11: Numerical results obtained with the proposed contact line force model and a mesh resolution of 100 cpr for the drop impact of Test 5. Comparison with experimental results [120].

Figure 5.11 shows a comparison between the experimental and numerical results for the drop shape at several instants after impact. The 3D representation is obtained from the 2D axisymmetric using the post-processing software ParaView. The agreement between the simulation results for the drop shape and contact angle and the experimental ones is good. The numerical results also reproduce well the capillary waves that undulate the drop surface. Also note the air bubble attached to the substrate in the center of the droplet, which can be seen in both the numerical and experimental results. At $t = 2.6$ ms, the diameter obtained with CLFM is appreciably larger than that shown in the experiments. Note that the rebound motion predicted by the numerical model at $t = 6.2$ ms is delayed with respect to the experimental image. This is consistent with the faster receding velocity measured in the experiments, as can be observed from Fig. 5.10. However, the subsequent acceleration of the receding stage predicted by the numerical model makes the differences in the drop diameter and shape to be reduced considerably, as can be observed at $t = 10.2$ ms from Figs. 5.10 and 5.11.

5.2 Splashing drop impact

This section presents the experimental and numerical results of a drop impact on a solid surface resulting in a splash. It consists of the impact of a water drop of initial diameter $D_0 = 2.08$ mm and properties $\mu_l = 1$ mPa s, $\rho_l = 1000$ kg m^{-3} and $\sigma = 70.9$ mN m^{-1}, at a velocity $U = 2.8$ m s^{-1} (Re = 5826, We = 230), on a smooth glass surface covered with a hydrophobic coating (Glaco Mirror Coat Zero, Soft 99 Ltd, Japan) that yields an equilibrium contact angle of about 160° for water. The coating consists of a dispersion of silica nano-particles in an alcohol-based solution.

The experiments were performed using an apparatus similar to that described in [117]. The water was injected into a needle from a syringe. The drops were released from a height of about 0.4 m to obtain the desired impact velocity. Phantom v7.3 and Photron NOVA S6 high-speed cameras were used. The images presented below were recorded at 36,036 fps and using an exposure time of 25 μs. Backlighting was provided by four LED lamps with a total light output of 14,000 lumens.

The impact is simulated using a 3D computational domain of size $8D \times 8D \times 2D$, discretized using a root mesh of $48 \times 48 \times 12$ cubic cells, which is statically refined on two cylindrical regions centered at the coordinate origin, with their axis parallel to the z axis (vertical direction) and four refinement levels. The first cylindrical region has a diameter of $7.2D_0$ and covers an area $0 \leq z \leq 0.29D_0$, and the second has a diameter of $1.44D_0$ and covers an area $0.29D_0 \leq z \leq 1.15D_0$. The resulting mesh has a maximum equivalent resolution of 50 cpr. The drop center is initially placed at $(4D_0, 4D_0, 0.55D_0)$. Figure 5.12 shows a detail of the mesh cells around the contact line where the force is applied and the force vector distribution.

The discretization and numerical schemes are the same as that used in Section 5.1.2, except for the VOF method and the discretization scheme of the convective term in the momentum equation, which have been changed in some simulations. The impact surface is considered as a solid wall and the rest of the boundaries are open boundaries, and therefore the same boundary conditions used in Section 5.1.2 apply.

Initially, the simulations were carried out using the proposed CLFM and the same algebraic VOF method used in the previous tests. However, this yielded results in which the front of the spreading lamella broke in a physically unrealistic way in the first instants after impact, probably due to the numerical diffusion of the interface introduced by the algebraic VOF scheme, especially in regions of high velocity gradients. As noted above, this prompted to consider using more accurate interface tracking methods. The alternative chosen here is to use CLFM in conjunction with the VOF method resulting from the isoAdvector-CLCIR combination proposed in Section 3.2.3, once verified the good results obtained in terms of efficiency and accuracy shown in Section 4.6.

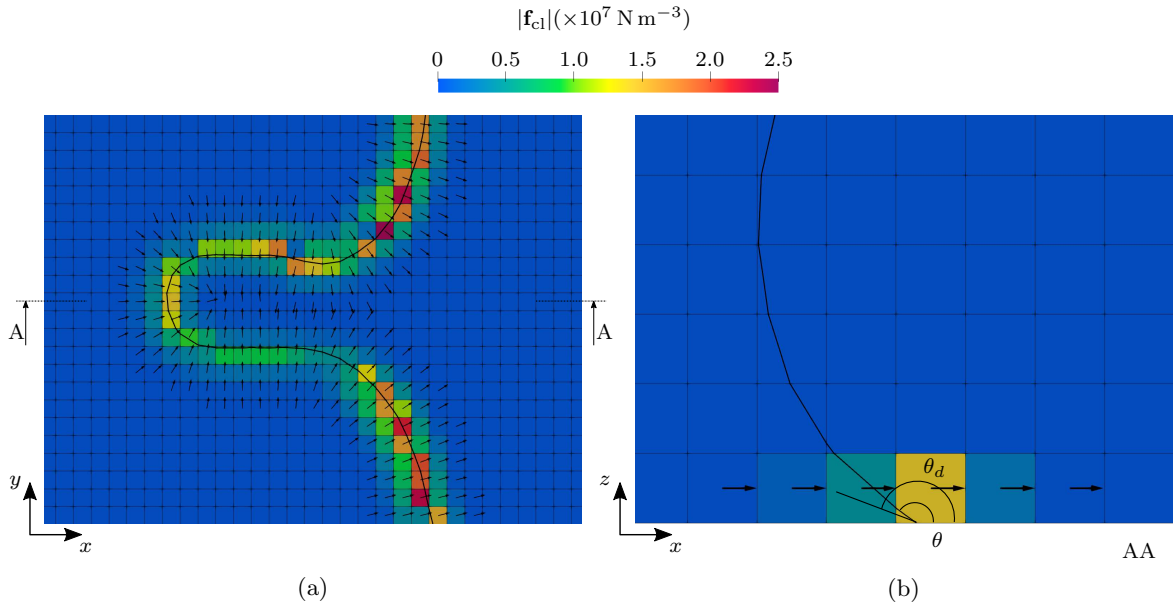


Figure 5.12: Detail of the volumetric contact line force distribution in the vicinity of the interface (0.5-isosurface, black line) at $t = 1$ ms, in the simulation of Test 6. (a) Top view of a finger with plotted contact line force vector orientation and magnitude. (b) Side view (cut plane AA) of the finger showing the calculated interface angle, $\theta = 145°$, and the dynamic contact angle obtained from Kistler's model, $\theta_d = 162°$.

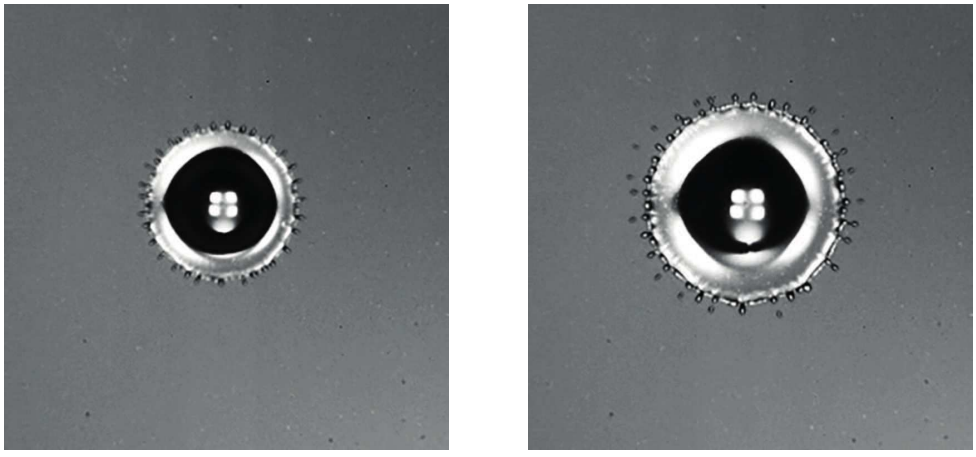
In all the simulations, the volume fraction tolerance was set to 1×10^{-6} and the maximum number of bounding steps to 5. In CLFM, since the advancing and receding contact angles are unknown, only the equilibrium contact angle $\theta_e = 160°$ is used.

When a second-order scheme is used to discretize the convective term in the momentum equation, a fingering in the spreading lamella has been found to occur soon after the drop impact. This has been observed to happen with different second-order schemes. Numerical dispersion errors are expected to be responsible for instabilities that can lead to undulations in the lamella that may eventually cause its local thinning and the formation of holes, not observed in the experiments, located between the front of the spreading lamella and the center of the droplet. Especially when the contact angle is high, and due to the high velocity near the wall, the holes tend to grow due to the effect of the surface tension and the force applied on the contact line, increasing their diameter and eventually breaking the liquid front. Another effect that might be attributed to the instabilities is the overgrowth of the length of the fingers, which tend to be thinner than that observed in experiments. To avoid these numerical difficulties, after the onset of finger formation and the detachment of the first secondary droplets, the simulation is stopped at $t \approx 0.4$ ms and the second-order upwind scheme used in the discretization of the convective term is replaced by a much more diffusive first-order upwind scheme. Obviously, it would be interesting to have a more robust and accurate discretization scheme of the convective term, which would make it possible to better reproduce the complex phenomena that appear during droplet

impact under the conditions considered. The complexity of the numerical simulation of these phenomena is probably the reason why there are relatively few publications with numerical results obtained for conditions leading to impact outcomes of similar degree of complexity.

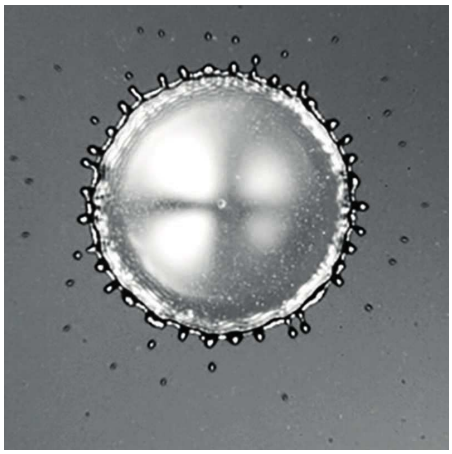
Figure 5.13 shows the comparison between the experimental and numerical results obtained at four representative instants. Note that the images of the experiments are taken from below, while the numerical results are shown from above in order to observe more clearly the evolution of the fingers. At instant $t = 0.3$ ms, fingers are observed in both the experimental and numerical results, although in the latter they are clearly thinner than in the experimental pictures. There are also detached droplets in the numerical results that are not visible in the experiments. At instant $t = 0.5$ ms, the shape of the fingers obtained numerically is not much closer to that observed in the experiments, and there are secondary droplets that were detached from the fingertips at instants in between those of Figs. 5.13(a) and (b), in agreement with experiments. At $t = 1$ ms, the numerical results show that the model is able to reproduce the finger coalescence phenomenon observed experimentally. Note that both the numerical and experimental results show that the fingerstips and the rim of the lamella increase in size with time. The undulations of the free surface propagating from the center towards the periphery of the drop observed experimentally are also reproduced in the numerical simulations. However, the number and especially the size of the secondary droplets are smaller in the numerical results, which is the main discrepancy between the numerical and experimental results. At $t = 1.3$ ms, the finger shape and distribution along the lamella front obtained numerically are very similar to those observed in the experiments, with a difference of about 2 in the number of fingers. Note that fingers of different sizes and shapes coexist in both the experimental and numerical results. On the other hand, note also that in the experimental images there are a few secondary droplets (exactly three) that have detached from the fingertips between the instants corresponding to Figs. 5.13(c) and (d), whereas they do not exist in the numerical results, which can surely be attributed to the first-order accuracy of the discretization of the convective term in the momentum equation.

In addition to the above qualitative comparison, which has shown reasonable agreement between numerical and experimental results despite the complexity of the flow, a quantitative comparison is presented below. The experimental images are treated with image processing software imageJ [64] in order to measure the outer, $\overline{D}_{\text{ext}}$, and inner, $\overline{D}_{\text{inn}}$, diameters defined by the tip and base of the fingers. The value of each of these diameters is obtained as an average of several measurements. Similarly, these diameters are obtained from the numerical results by measuring them on the 0.5-isosurface representation obtained with the help of ParaView software. Figure 5.14 shows good agreement between numerical predictions and experimental measurements for the evolution of the dimensionless inner and outer diameters. The only more ap-

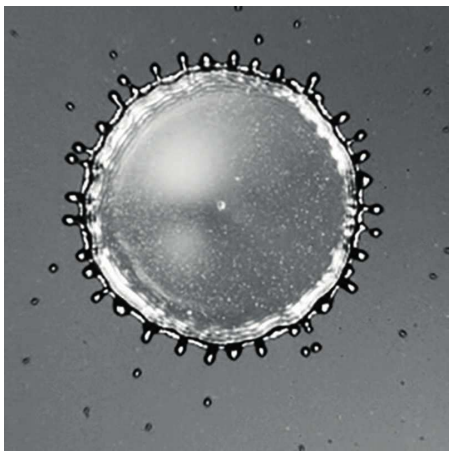


(a) $t = 0.3$ ms

(b) $t = 0.5$ ms



(c) $t = 1$ ms



(d) $t = 1.3$ ms

Figure 5.13: Comparison between the numerical results obtained with the proposed CLFM and isoAdvector-CLCIR and experimental results in the splashing drop impact. In the numerical results, the 0.5-isosurface is represented as viewed from above while the experimental images are taken from below.

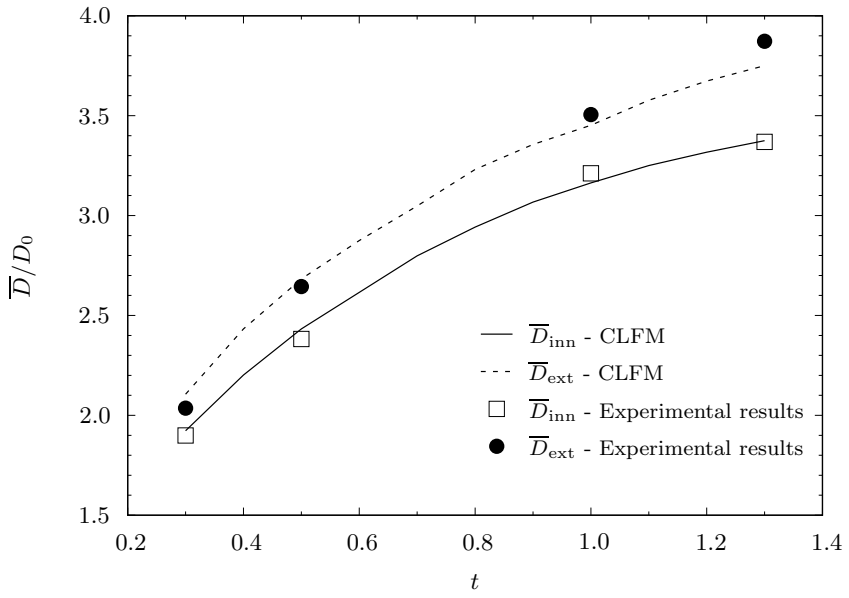


Figure 5.14: Evolution of the spread in the splashing drop impact. Comparison between the numerical results obtained with the proposed CLFM and isoAdvector-CLCIR, and the experimental results.

preciable discrepancy occurs for the last two instants of Figs. 5.13(c) and (d), for which the numerically predicted outer diameter is below the experimental data, while the agreement for the inner diameter is better for the whole time range considered. It then turns out that the predicted finger size is slightly smaller than that obtained experimentally.

The approach used in this work to promote finger onset is different from that adopted by Bussmann et al. [20], which, as mentioned in Section 1.5, relies on imposing a perturbation to the radial component of the velocity near the solid surface that triggers fingering at the advancing front. The application of such a perturbation to the flow is intended to reproduce numerically the effects caused by the instabilities that give rise to finger formation, as described in Section 1.2.

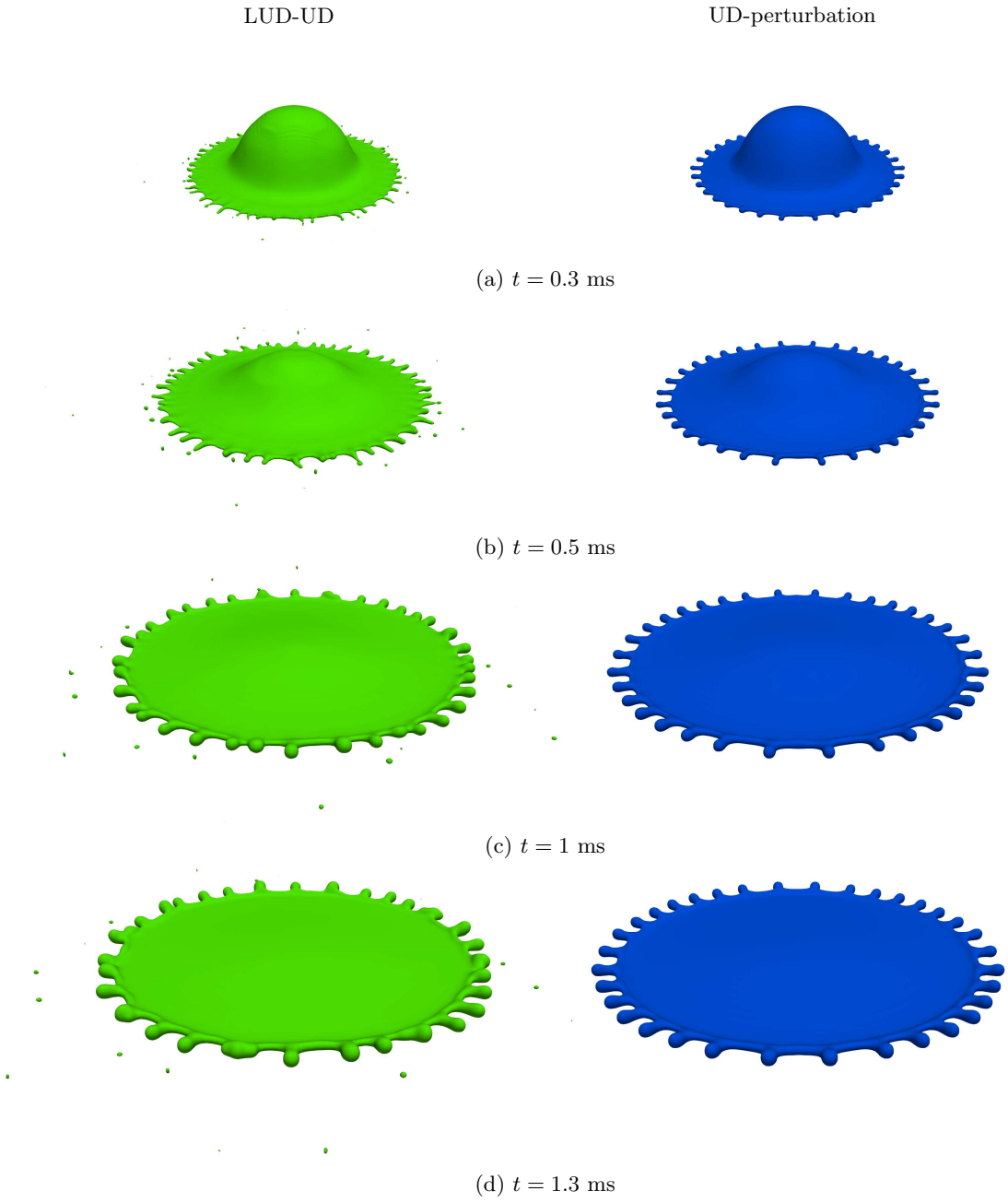
The results obtained in this work are compared with those obtained with the approach used by Bussmann et al. [20] to promote fingering. For this purpose, the radial component of the velocity near the solid surface, $\mathbf{u}_{r,up}$, is perturbed to act on the spreading front of the lamella at $t^* = 0.1$. Thus, the perturbed radial velocity is obtained through

$$\mathbf{u}_{r,p} = \mathbf{u}_{r,up}\left[1 + A_p\, e^{-B_p(z/D_0)^2}\cos(N\,\vartheta)\right], \tag{5.5}$$

where A_p is the amplitude of the perturbation at the impact surface ($z = 0$), which is assumed to be 0.25; B_p is the rate of decay of the perturbation away from the solid surface, which is selected to be 4000; N is the number of fingers; and ϑ is the azimuthal coordinate. The velocity perturbation is applied at the first three layers above the solid surface. To apply the perturbation on the velocity, the simulation is stopped and then restarted.

Since the upwind scheme does not cause perturbations that promote fingering, but is capable of propagating them once they have been introduced, it is used throughout the simulation along with CLFM and isoAdvector-CLCIR. However, it should be noted that such a scheme will tend to inhibit secondary droplet detachment. Based on the experimental data, the number of fingers imposed is $N = 34$ and the rest of the conditions are kept the same as in the previous simulation. Figure 5.15 shows the results obtained with the implementation of this approach (blue) compared to those obtained with the one used in this work (green). Note that, at $t = 0.3$ ms, the distribution of fingers along the advancing front of the lamella obtained when the perturbation in velocity and a first-order upwind (UD) scheme are applied is evidently very regular compared to that obtained when it is the less diffusive and more unstable second-order linear upwind differencing (LUD) scheme that causes the onset of fingering. At $t = 0.5$ ms, the numerical results obtained when initially using LUD (up to $t = 0.3$ ms) show secondary droplets detached from the fingers, while the simulation with first-order upwind differencing and use of a velocity perturbation shows no secondary droplets. Despite this, the shape and size of the fingers at $t = 1$ ms and $t = 1.3$ ms are very similar in both results, although the regularity is much higher in the second case.

Figure 5.16 shows a top view of the isosurface contour in the horizontal midplane cutting the fingers. The plane is located a small distance above the surface ($\sim 3\,\mu$m). The numerical results obtained with the LUD-UD and UD-perturbation schemes for the convective term in the momentum equation are compared with the mean diameters $\overline{D}_{\text{inn}}$ and $\overline{D}_{\text{ext}}$, measured on the experimental images. The agreement with the experimental results is good. The differences on the size of the fingers is very clear at $t = 0.3$ ms, where those obtained with the LUD-UD approach are slightly larger than the measured experimentally and than that obtained with UD-perturbation. The secondary droplets detached from these fingers can also be observed at $t = 0.5$ ms. At this instant, the size of the fingers in UD-perturbation is closer to the experimental data. At instants $t = 1$ ms and $t = 1.3$ ms, the agreement between experimental diameters and those given by LUD-UD is very good, especially for $\overline{D}_{\text{inn}}$.



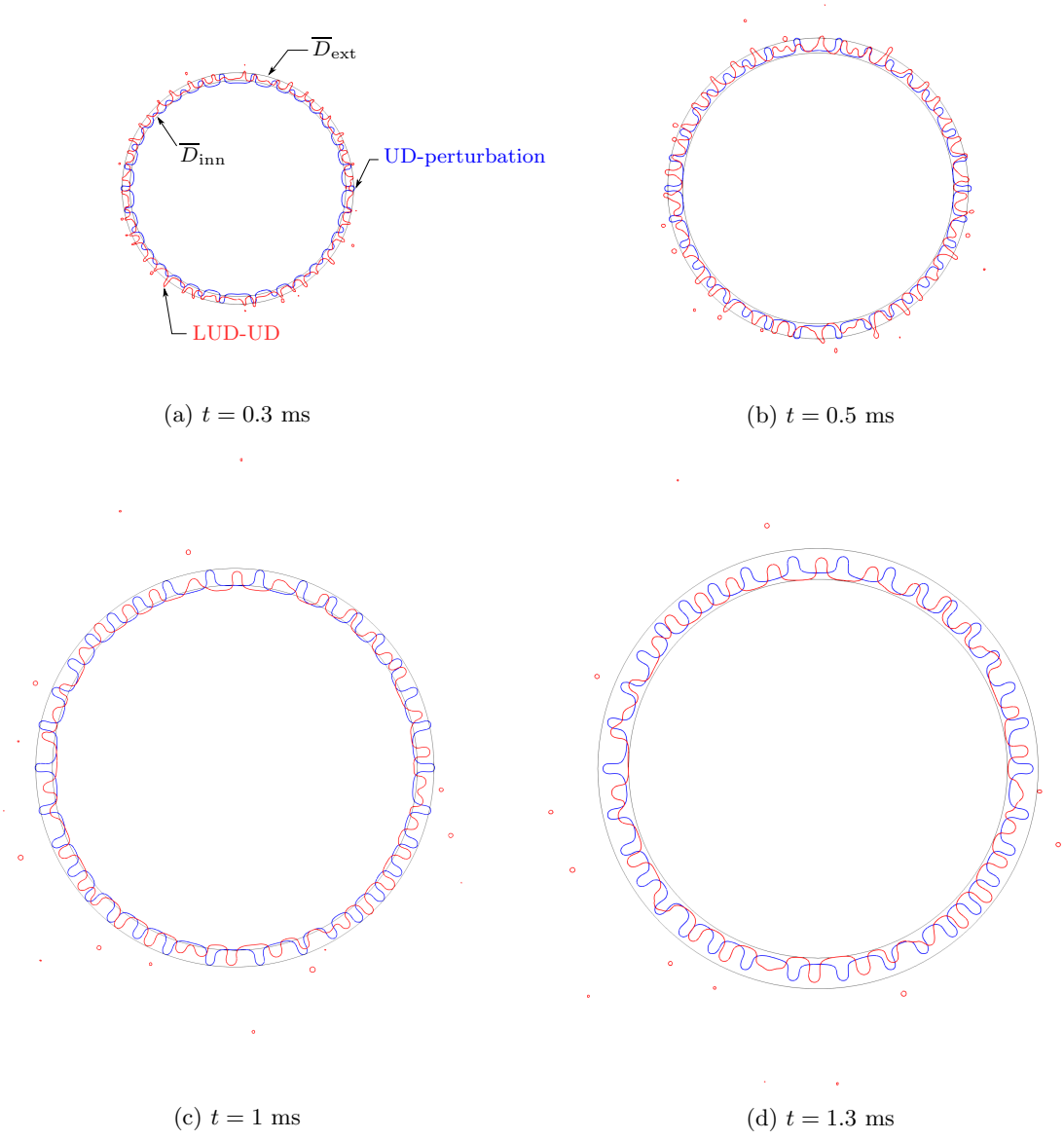
(a) $t = 0.3$ ms

(b) $t = 0.5$ ms

(c) $t = 1$ ms

(d) $t = 1.3$ ms

Figure 5.15: Comparison between the numerical results for the splashing drop impact obtained using a LUD scheme up to $t = 0.3$ ms (green, left column) and a UD scheme with a perturbation in the radial velocity (blue, right column).



(a) $t = 0.3$ ms

(b) $t = 0.5$ ms

(c) $t = 1$ ms

(d) $t = 1.3$ ms

Figure 5.16: Top view of the isosurface contour at the mid-horizontal plane cutting the fingers, obtained with LUD-UD and UD-perturbation. The black lines represent the experimental diameters $\overline{D}_{\text{inn}}$ and $\overline{D}_{\text{ext}}$.

Chapter 6

Conclusions and future work

6.1 Conclusions

A numerical study of the flow resulting from the impact of drops on solid surfaces has been carried out, with special emphasis on the analysis of the contact line dynamics. For this purpose, different aspects of the problem have been addressed: the discretization schemes used for solving the Navier-Stokes equations, in particular for the convective term in the momentum equation; the alternatives of using an algebraic or geometric type VOF method of interface tracking; the algorithms used for the reconstruction and advection of the interface within the geometric type VOF method; different possible models to reproduce the contact line dynamics; and other computational aspects, in particular the chosen mesh. These aspects have been analyzed making use of the general purpose code OpenFOAM. Some methods used were already available in the code (the MULES algebraic VOF method and the advection algorithm of the isoAdvector geometric VOF method); others, already existing, had to be implemented (CLCIR reconstruction algorithm and isoap surface extraction method); and, as regards the contact line model, which is the main contribution of the present work, a new model (contact line force model, CLFM) has been proposed and implemented, and compared with other models that have also had to be implemented in OpenFOAM.

The above has provided an analysis of the suitability of the proposed contact line model and OpenFOAM for the simulation of the complex phenomena that may arise in drop impacts for impact conditions leading to different flow patterns, from deposition to splashing.

Different VOF-type interface tracking methods have been analyzed. On the one hand, the capabilities of the algebraic VOF method used by default in interFoam have been assessed, having found limitations to simulate drop impacts that give rise to outcomes different from a simple deposition. On the other hand, some alternative advection and reconstruction algorithms have been assessed for use in a geometric VOF method to improve the performance of the algebraic method. A number of kinematic and dynamic tests have shown that a combination of the advection algorithm

used in isoAdvector and the CLCIR reconstruction algorithm provides an adequate balance between accuracy and computational efficiency, not only for analyzing the type of flow under study, but probably even more complex flows demanding higher computational resources. Through the tests, an analysis of the parameters influencing the performance of the algorithms was also carried out.

A contact line force model has been proposed to reproduce the dynamics of the contact line on solid walls. Its main contribution lies in the fully three-dimensional character, which allows its use in the numerical simulation of drop impacts on solid surfaces other than simple deposition (e.g., involving fingering and splashing), and which has enabled good reproduction of experiments to be achieved. The development of a 3D model entails additional difficulties to those of 2D models, such as the calculation of the contact line velocity and the introduction of an appropriate force distributed along the contact line, aspects that become more relevant at complex interfaces, such as those arising in drop impacts resulting in splashing. Another improvement has been to assume that the force on the contact line depends on the local deviation of the calculated contact angle from that predicted by a dynamic contact angle model for the calculated contact line velocity, which takes into account wettability hysteresis. This force is applied at wall cells in the vicinity of those containing the interface, which is defined by a 0.5-isosurface obtained from the VOF distribution. When an algebraic VOF method is used in the numerical simulations, and given the diffuse nature of the interface, an interface reconstruction step is used to find the interfacial wall cells. In addition, the interface contact angle is calculated from the 0.5-isosurface orientation, which yields a more accurate value than that obtained from the volume fraction gradient. As already mentioned, the deviation of the angle calculated in this way from that prescribed by a given dynamic contact angle model is used to determine the contact line force. The force model is used while also imposing a boundary condition on the wall for the fluid volume fraction, based on the contact angle prescription. The latter is essentially equivalent to taking into account the prescribed interface orientation at the wall when calculating the interface curvature used to determine the surface tension force in the CSF method.

A detailed analysis of the sensitivity and validity of the proposed contact line force model has been carried out by simulating several axisymmetric drop impacts on solid surfaces in a wide range of Reynolds and Weber numbers. The results compare favorably with those obtained with other contact angle models. Also, the introduction of the contact line force substantially improves the ability of the numerical model to reproduce the interfacial dynamics observed in several experimental tests from other authors.

The proposed contact line force model has been applied along with the VOF method consisting of the selected combination of reconstruction and advection algorithms to simulate the impact of a drop on a hydrophobic surface with a high contact angle. The

numerical simulation has been proven to reproduce with reasonable agreement the fingering and splashing phenomena observed in the experimental results. The numerical results were shown to be highly sensitive to the scheme used in the discretization of the convective term of the momentum equation. As in previous work by other authors, it has been found that a first-order upwind scheme cannot trigger finger growth, so its use requires introducing a velocity perturbation to achieve it. Even so, with the mesh sizes used, it has not been possible to achieve secondary droplet detachment from the fingertips, although the shape, size and number of fingers are reproduced quite well at all times. On the other hand, second-order schemes such as LUD do predict finger formation in adequate number without the need for any perturbation, but the fingers are appreciably thinner than those observed experimentally, and thus the secondary droplets detached from them are smaller in size than those generated in the experiments. The combination of first- and second-order schemes at different stages of the drop impact and lamella spreading processes allows the shape and size of the fingers to end up very close to those observed experimentally.

6.2 Future work

The following are some topics that would be of interest for further research:

- It would be desirable to address the limitations of the proposed contact line force model related to the dependence of the model on the experimental values of the contact angle, values about which, moreover, there is often uncertainty. In addition, the evolution of the receding stage needs in-depth analysis to find better estimates of the dynamic contact angle that can be used in the proposed CLFM.
- The influence of the discretization scheme of the convective term in the momentum equation on the fingering pattern and secondary droplet detachment, and the development of more accurate schemes, requires further investigation.
- The effect of surface irregularities produced by the superhydrophobic coating should be analyzed.
- A full implementation in OpenFOAM of the gVOF advection methods and their parallelization with the MPI application would allow a comparison of advection methods under similar execution conditions. It would also be desirable to improve the implementation of the reconstruction methods to avoid data duplication and thus increase their computational efficiency. Since the isoAdvector-CLCIR combination has been shown to give very accurate results at low computational cost, further modifications of these methods could improve the overall performance of such a combination, particularly for high values of CFL, for which the advection method loses accuracy.

- A hybrid implementation of parallelization of the reconstruction methods using both MPI and OpenMP applications, with each MPI process using multiple threads, would be desirable for future code enhancements. It is expected that this could significantly reduce computational time, thus allowing much more complex problems with finer mesh resolutions to be solved, with a small increase in computational resource demand.
- The instabilities that arise in the impact of a drop of high viscosity and its relationship with the hysteresis of the static contact angle require further research.

Bibliography

[1] S. Afkhami, J. Buongiorno, A. Guion, S. Popinet, Y. Saade, R. Scardovelli, and S. Zaleski. Transition in a numerical model of contact line dynamics and forced dewetting. *Journal of Computational Physics*, 374:1061 – 1093, 2018.

[2] S. Afkhami, S. Zaleski, and M. Bussmann. A mesh-dependent model for applying dynamic contact angles to VOF simulations. *Journal of Computational Physics*, 228(15):5370 – 5389, 2009.

[3] H. T. Ahn and M. Shashkov. Multi-material interface reconstruction on generalized polyhedral meshes. *Journal of Computational Physics*, 226(2):2096–2132, 2007.

[4] H. T. Ahn and M. Shashkov. Geometric algorithms for 3D interface reconstruction. In M. L. Brewer and D. Marcum, editors, *Proceedings of the 16th International Meshing Roundtable*, pages 405–422. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[5] A. Albadawi, D. Donoghue, A. Robinson, D. Murray, and Y. Delauré. 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, 2013.

[6] R. F. Allen. The role of surface tension in splashing. *Journal of Colloid and Interface Science*, 51(2):350–351, 1975.

[7] C. Antonini, F. J. Carmona, E. Pierce, M. Marengo, and A. Amirfazli. General methodology for evaluating the adhesion force of drops and bubbles on solid surfaces. *Langmuir*, 25(11):6143–6154, 2009.

[8] E. Aulisa, S. Manservisi, R. Scardovelli, and S. Zaleski. Interface reconstruction with least-squares fit and split advection in three-dimensional Cartesian geometry. *Journal of Computational Physics*, 225(2):2301–2319, 2007.

[9] S. D. Aziz and S. Chandra. Impact, recoil and splashing of molten metal droplets. *International Journal of Heat and Mass Transfer*, 43(16):2841–2857, 2000.

[10] J. L. Barrera and K. Maute. Ambiguous phase assignment of discretized 3d geometries in topology optimization. *Computer Methods in Applied Mechanics and Engineering*, 369:113201, 2020.

[11] T. Blake and Y. Shikhmurzaev. Dynamic wetting by liquids of different viscosity. *Journal of Colloid and Interface Science*, 253(1):196–202, 2002.

[12] T. D. Blake. The physics of moving wetting lines. *Journal of Colloid and Interface Science*, 299(1):1 – 13, 2006.

[13] L. Bocquet and E. Charlaix. Nanofluidics, from bulk to interfaces. *Chem. Soc. Rev.*, 39:1073–1095, 2010.

[14] A. M. P. Boelens and J. J. de Pablo. Simulations of splashing high and low viscosity droplets. *Physics of Fluids*, 30(7):072106, 2018.

[15] A. M. P. Boelens and J. J. de Pablo. Generalised Navier boundary condition for a volume of fluid approach using a finite-volume method. *Physics of Fluids*, 31(2):021203, 2019.

[16] D. Bonn, J. Eggers, J. Indekeu, J. Meunier, and E. Rolley. Wetting and spreading. *Reviews of Modern Physics*, 81:739–805, 2009.

[17] J. P. Boris and D. L. Book. Flux-corrected transport. i. SHASTA, a fluid transport algorithm that works. *Journal of Computational Physics*, 11(1):38–69, 1973.

[18] J. Brackbill, D. Kothe, and C. Zemach. A continuum method for modeling surface tension. *Journal of Computational Physics*, 100(2):335 – 354, 1992.

[19] R. Brent. *Algorithms for minimization without derivatives*. Prentice-Hall, Englewood Cliffs, New Jersey, 1973.

[20] M. Bussmann, S. Chandra, and J. Mostaghimi. Modeling the splash of a droplet impacting a solid surface. *Physics of Fluids*, 12(12):3121–3132, 2000.

[21] S. Chandra and C. T. Avedisian. On the collision of a droplet with a solid surface. *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences*, 432(1884):13–41, 1991.

[22] S. Chandrasekhar. *Hydrodynamic and Hydromagnetic Stability*. Dover Books on Physics Series. Dover Publications, 1981.

[23] X. Chen and X. Zhang. A predicted-Newton's method for solving the interface positioning equation in the MoF method on general polyhedrons. *Journal of Computational Physics*, 384:60–76, 2019.

[24] E. V. Chernyaev. Marching cubes 33: Construction of topologically correct isosurfaces. Technical report, CERN Report, CN/95-17, 1995.

[25] C. Chevalier and F. Pellegrini. PT-Scotch: A tool for efficient parallel graph ordering. *Parallel Computing*, 34(6):318–331, 2008. Parallel Matrix Algorithms and Applications.

[26] A. J. Chorin. Curvature and solidification. *Journal of Computational Physics*, 57(3):472–490, 1985.

[27] P. Cifani, W. Michalek, G. Priems, J. Kuerten, C. van der Geld, and B. Geurts. A comparison between the surface compression method and an interface reconstruction method for the VOF approach. *Computers & Fluids*, 136:421–435, 2016.

[28] R. Clift, J. Grace, and M. E. Weber. *Bubbles, Drops, and Particles*. Academic Press, 1978.

[29] R. Courant, K. Friedrichs, and H. Lewy. On the partial difference equations of mathematical physics. *IBM Journal of Research and Development*, 11:215–234, 1967.

[30] R. G. Cox. The dynamics of the spreading of liquids on a solid surface. part 1. viscous flow. *Journal of Fluid Mechanics*, 168:169–194, 1986.

[31] S. J. Cummins, M. M. Francois, and D. B. Kothe. Estimating curvature from volume fractions. *Computers & Structures*, 83(6):425–434, 2005. Frontier of Multi-Phase Flow Analysis and Fluid-Structure.

[32] D. Dai and A. Y. Tong. An analytical interface reconstruction algorithm in the PLIC-VOF method for 2D polygonal unstructured meshes. *International Journal for Numerical Methods in Fluids*, 88(6):265–276, 2018.

[33] D. Dai and A. Y. Tong. Analytical interface reconstruction algorithms in the PLIC-VOF method for 3D polyhedral unstructured meshes. *International Journal for Numerical Methods in Fluids*, 91(5):213–227, 2019.

[34] P. G. de Gennes. Wetting: statics and dynamics. *Reviews of Modern Physics*, 57:827–863, 1985.

[35] P. G. de Gennes. On fluid/wall slippage. *Langmuir*, 18(9):3413–3414, 2002.

[36] T. C. de Goede, N. Laan, K. G. de Bruin, and D. Bonn. Effect of wetting on drop splashing of Newtonian fluids and blood. *Langmuir : the ACS journal of surfaces and colloids*, 34(18):5163–5168, 2018.

[37] D. Deganello, T. Croft, A. Williams, A. Lubansky, D. Gethin, and T. Claypole. Numerical simulation of dynamic contact angle using a force based formulation. *Journal of Non-Newtonian Fluid Mechanics*, 166(16):900–907, 2011. Papers Presented at the University of Wales Institute of Non-Newtonian Fluid Mechanics Meeting on Rheometry, 29-31 March 2010, Lake Vyrnwy, Wales.

[38] S. S. Deshpande, L. Anumolu, and M. F. Trujillo. Evaluating the performance of the two-phase flow solver interFoam. *Computational Science & Discovery*, 5(1):014016, 2012.

[39] M. Dianat, M. Skarysz, and A. Garmory. A Coupled Level Set and Volume of Fluid method for automotive exterior water management applications. *International Journal of Multiphase Flow*, 91:19–38, 2017.

[40] A. Doi and A. Koide. An efficient method of triangulating equi-valued surfaces by using tetrahedral cells. *IEICE Transactions on Information and Systems*, 74:214–224, 1991.

[41] M. M. Driscoll, C. S. Stevens, and S. R. Nagel. Thin film formation during splashing of viscous liquids. *Physical Review E*, 82:036302, 2010.

[42] J. Dupont and D. Legendre. Numerical simulation of static and sliding drop with contact angle hysteresis. *Journal of Computational Physics*, 229(7):2453 – 2478, 2010.

[43] E. B. Dussan. On the spreading of liquids on solid surfaces: Static and dynamic contact lines. *Annual Review of Fluid Mechanics*, 11(1):371–400, 1979.

[44] E. B. Dussan, E. Ramé, and S. Garoff. On identifying the appropriate boundary conditions at a moving contact line: an experimental investigation. *Journal of Fluid Mechanics*, 230:97–116, 1991.

[45] D. Enright, R. Fedkiw, J. Ferziger, and I. Mitchell. A hybrid particle level set method for improved interface capturing. *Journal of Computational Physics*, 183(1):83–116, 2002.

[46] A. Esteban, J. López, J. Hernández, P. Gómez, and J. Roenby. On the accuracy and efficiency of advection and reconstruction algorithms of geometric volume of fluids methods, 2021. In submitting process.

[47] A. Esteban, J. López, J. Hernández, P. Gómez, C. Zanzi, and M. Bussmann. A contact line force model for the numerical simulation of drop impacts on solid surfaces using volume of fluid methods, 2021. In submitting process.

[48] J. H. Ferziger and M. Perić. *Computational Methods for Fluid Dynamics*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2002.

[49] M. M. François. Recent numerical and algorithmic advances within the volume tracking framework for modeling interfacial flows. *Procedia IUTAM*, 15:270–277, 2015. IUTAM Symposium on Multiphase Flows with Phase Change: Challenges and Opportunities.

[50] L. Gamet, M. Scala, J. Roenby, H. Scheufler, and J. L. Pierson. Validation of volume-of-fluid OpenFOAM® isoadvector solvers using single bubble benchmarks. *Computers & Fluids*, 213:104722, 2020.

[51] J. Gerbeau and T. Lelièvre. Generalized Navier boundary condition and geometric conservation law for surface tension. *Computer Methods in Applied Mechanics and Engineering*, 198(5):644 – 656, 2009.

[52] A. Giacomello, S. Meloni, M. Chinappi, and C. M. Casciola. Cassie–baxter and wenzel states on a nanostructured surface: Phase diagram, metastabilities, and transition mechanism by atomistic free energy calculations. *Langmuir*, 28(29):10764–10772, 2012.

[53] J. Göhl, A. Mark, S. Sasic, and F. Edelvik. An immersed boundary based dynamic contact angle framework for handling complex surfaces of mixed wettabilities. *International Journal of Multiphase Flow*, 109:164 – 177, 2018.

[54] D. Gueyffier, J. Li, A. Nadim, R. Scardovelli, and S. Zaleski. Volume-of-Fluid interface tracking with smoothed surface stress methods for three-dimensional flows. *Journal of Computational Physics*, 152(2):423–456, 1999.

[55] D. Gueyffier and S. Zaleski. Formation de digitations lors de l'impact d'une goutte sur un film liquide. *Comptes Rendus De L Academie Des Sciences Serie Ii Fascicule B-mecanique Physique Astronomie*, 326:839–844, 1998.

[56] Y. Guo, Y. Lian, and M. Sussman. Investigation of drop impact on dry and wet surfaces with consideration of surrounding air. *Physics of Fluids*, 28(7):073303, 2016.

[57] M. Haghshenas, J. A. Wilson, and R. Kumar. Algebraic Coupled Level Set-Volume of Fluid method for surface tension dominant two-phase flows. *International Journal of Multiphase Flow*, 90:13–28, 2017.

[58] D. J. Harvie and D. F. Fletcher. A new volume of fluid advection algorithm: The stream scheme. *Journal of Computational Physics*, 162(1):1–32, 2000.

[59] D. J. E. Harvie and D. F. Fletcher. A new volume of fluid advection algorithm: The defined donating region scheme. *International Journal for Numerical Methods in Fluids*, 35(2):151–172, 2001.

[60] J. Hernández, J. López, P. Gómez, C. Zanzi, and F. Faura. 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(8):897–921, 2008.

[61] C. Hirt and B. Nichols. Volume of fluid (VOF) method for the dynamics of free boundaries. *Journal of Computational Physics*, 39(1):201–225, 1981.

[62] R. L. Hoffman. A study of the advancing interface. i. interface shape in liquid-gas systems. *Journal of Colloid and Interface Science*, 50(2):228–241, 1975.

[63] S. Hysing, S. Turek, D. Kuzmin, N. Parolini, E. Burman, S. Ganesan, and L. Tobiska. Quantitative benchmark computations of two-dimensional bubble dynamics. *International Journal for Numerical Methods in Fluids*, 60(11):1259–1288, 2009.

[64] imageJ. https://imagej.nih.gov/ij/, v1.53m.

[65] M. Isoz. *Dynamics of rivulets and other multiphase flows*. Ph.D. thesis, University of Chemistry and Technology, Prague, 2018.

[66] R. Issa, A. Gosman, and A. Watkins. The computation of compressible and incompressible recirculating flows by a non-iterative implicit scheme. *Journal of Computational Physics*, 62(1):66–82, 1986.

[67] C. B. Ivey and P. Moin. Conservative and bounded volume-of-fluid advection on unstructured grids. *Journal of Computational Physics*, 350:387–419, 2017.

[68] H. Jasak. *Error Analysis and Estimation for the Finite Volume Method with Applications to Fluid Flows*. Ph.D. thesis, Imperial College of Science, Technology and Medicine, 1996.

[69] Z. Jian, C. Josserand, S. Popinet, P. Ray, and S. Zaleski. Two mechanisms of droplet splashing on a solid substrate. *Journal of Fluid Mechanics*, 835:1065–1086, 2018.

[70] T. Jiang, O. Soo-Gun, and J. C. Slattery. Correlation for dynamic contact angle. *Journal of Colloid and Interface Science*, 69(1):74 – 77, 1979.

[71] L. Jofre, O. Lehmkuhl, J. Castro, and A. Oliva. A 3-D Volume-of-Fluid advection method based on cell-vertex velocities for unstructured meshes. *Computers & Fluids*, 94:14–29, 2014.

[72] C. Josserand and S. Thoroddsen. Drop impact on a solid surface. *Annual Review of Fluid Mechanics*, 48(1):365–391, 2016.

[73] N. Kavokine, R. R. Netz, and L. Bocquet. Fluids at the nanoscale: From continuum to subcontinuum transport. *Annual Review of Fluid Mechanics*, 53(1):377–410, 2021.

[74] H. Kim, U. Park, C. Lee, H. Kim, M. Hwan Kim, and J. Kim. Drop splashing on a rough surface: How surface morphology affects splashing threshold. *Applied Physics Letters*, 104(16):161608, 2014.

[75] S. F. Kistler. *Wettability*. Marcel Dekker, New York, 1993.

[76] J. M. Kolinski, S. M. Rubinstein, S. Mandre, M. P. Brenner, D. A. Weitz, and L. Mahadevan. Skating on a film of air: Drops impacting on a surface. *Physical Review Letters*, 108:074503, 2012.

[77] J. Koplik, J. R. Banavar, and J. F. Willemsen. Molecular dynamics of fluid flow at solid surfaces. *Physics of Fluids A: Fluid Dynamics*, 1(5):781–794, 1989.

[78] A. Latka, A. M. P. Boelens, S. R. Nagel, and J. J. de Pablo. Drop splashing is independent of substrate wetting. *Physics of Fluids*, 30(2):022105, 2018.

[79] J. B. Lee, D. Derome, R. Guyer, and J. Carmeliet. Modeling the maximum spreading of liquid droplets impacting wetting and nonwetting surfaces. *Langmuir*, 32(5):1299–1308, 2016.

[80] J. S. Lee, B. M. Weon, J. H. Je, and K. Fezzaa. How does an air film evolve into a bubble during drop impact? *Physical Review Letters*, 109:204501, 2012.

[81] D. Legendre and M. Maglio. Numerical simulation of spreading drops. *Colloids and Surfaces A: Physicochemical and Engineering Aspects*, 432:29–37, 2013.

[82] D. Legendre and M. Maglio. Comparison between numerical models for the simulation of moving contact lines. *Computers & Fluids*, 113:2 – 13, 2015. Small scale simulation of multiphase flows.

[83] D. Li, D. Zhang, Z. Zheng, and X. Tian. Numerical analysis on air entrapment during a droplet impacts on a dry flat surface. *International Journal of Heat and Mass Transfer*, 115:186–193, 2017.

[84] J. Li. Calcul d'interface affine par morceaux (Piecewise linear interface calculation). *Comptes Rendus de l'Académie des Sciences - Series II B*, 320:391–396, 1995.

[85] P. Liovic, M. Rudman, J.-L. Liow, D. Lakehal, and D. Kothe. A 3D unsplit-advection volume tracking algorithm with planarity-preserving interface reconstruction. *Computers & Fluids*, 35(10):1011–1032, 2006.

[86] Y. Liu, P. Tan, and L. Xu. Kelvin-Helmholtz instability in an ultrathin air film causes drop splashing on smooth surfaces. *Proceedings of the National Academy of Sciences*, 112(11):3280–3284, 2015.

[87] J. López, A. Esteban, J. Hernández, P. Gómez, R. Zamora, C. Zanzi, and F. Faura. A new isosurface extraction method on arbitrary grids. *Journal of Computational Physics*, page 110579, 2021.

[88] J. López and J. Hernández. Analytical and geometrical tools for 3D volume of fluid methods in general grids. *Journal of Computational Physics*, 227(12):5939–5948, 2008.

[89] J. López and J. Hernández. gVOF: An open-source package for unsplit geometric volume of fluid methods on arbitrary grids. 2021. ArXiv paper.

[90] J. López and J. Hernández. isoap: A software for isosurface extraction on arbitrary polyhedra. https://doi.org/10.17632/4rcf98s74c.1, 2021. Mendeley Data, V1.

[91] J. López, J. Hernández, P. Gómez, and F. Faura. A volume of fluid method based on multidimensional advection and spline interface reconstruction. *Journal of Computational Physics*, 195(2):718–742, 2004.

[92] J. López, J. Hernández, P. Gómez, and F. Faura. An improved PLIC-VOF method for tracking thin fluid structures in incompressible two-phase flows. *Journal of Computational Physics*, 208(1):51–74, 2005.

[93] J. López, J. Hernández, P. Gómez, and F. Faura. A new volume conservation enforcement method for PLIC reconstruction in general convex grids. *Journal of Computational Physics*, 316:338–359, 2016.

[94] J. López, J. Hernández, P. Gómez, and F. Faura. VOFTools – A software package of calculation tools for volume of fluid methods using general convex grids. *Computer Physics Communications*, 223:45–54, 2018.

[95] J. López, J. Hernández, P. Gómez, and F. Faura. Non-convex analytical and geometrical tools for volume truncation, initialization and conservation enforcement in VOF methods. *Journal of Computational Physics*, 392:666–693, 2019.

[96] J. López, J. Hernández, P. Gómez, F. Faura, and C. Zanzi. Application of non-convex analytic and geometric tools to a PLIC–VOF method. In *ASME 2016 International Mechanical Engineering Congress and Exposition IMECE2016*. 2016.

[97] J. López, C. Zanzi, P. Gómez, F. Faura, and J. Hernández. A new volume of fluid method in three dimensions – Part II: Piecewise-planar interface reconstruction with cubic-Bézier fit. *International Journal for Numerical Methods in Fluids*, 58(8):923–944, 2008.

[98] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *SIGGRAPH Computer Graphics*, 21(4):163–169, 1987.

[99] J. López, J. Hernández, P. Gómez, C. Zanzi, and R. Zamora. VOFTools 3.2: Added VOF functionality to initialize the liquid volume fraction in general convex cells. *Computer Physics Communications*, 245:106859, 2019.

[100] J. López, J. Hernández, P. Gómez, C. Zanzi, and R. Zamora. VOFTools 5: An extension to non-convex geometries of calculation tools for volume of fluid methods. *Computer Physics Communications*, 252:107277, 2020.

[101] J. López, C. Zanzi, P. Gómez, R. Zamora, F. Faura, and J. Hernández. An improved height function technique for computing interface curvature from volume fractions. *Computer Methods in Applied Mechanics and Engineering*, 198(33):2555–2564, 2009.

[102] I. Malgarinos, N. Nikolopoulos, M. Marengo, C. Antonini, and M. Gavaises. VOF simulations of the contact angle dynamics during the drop spreading: Standard models and a new wetting force model. *Advances in Colloid and Interface Science*, 212:1 – 20, 2014.

[103] S. Mandre and M. P. Brenner. The mechanism of a splash on a dry solid surface. *Journal of Fluid Mechanics*, 690:148–172, 2012.

[104] S. Manservisi and R. Scardovelli. A variational approach to the contact angle dynamics of spreading droplets. *Computers & Fluids*, 38(2):406 – 424, 2009.

[105] T. Marić, D. B. Kothe, and D. Bothe. Unstructured un-split geometrical Volume-of-Fluid methods – A review. *Journal of Computational Physics*, 420:109695, 2020.

[106] H. Marmanis and S. T. Thoroddsen. Scaling of the fingering pattern of an impacting drop. *Physics of Fluids*, 8(6):1344–1346, 1996.

[107] G. Masala, B. Golosio, and P. Oliva. An improved marching cube algorithm for 3d data segmentation. *Computer Physics Communications*, 184(3):777–782, 2013.

[108] N. Max. Weights for computing vertex normals from facet normals. *Journal of Graphics Tools*, 4(2):1–6, 1999.

[109] S. Mirjalili, S. Jain, and M. Dodd. Interface-capturing methods for two-phase flows: An overview and recent developments. *Center for Turbulence Research Annual Research Briefs*, pages 117–135, 2017.

[110] S. Mosso, B. Swartz, D. Kothe, and S. Clancy. Recent enhancements of volume tracking algorithms for irregular grids. Technical report, Los Alamos Natl.Lab., Los Alamos, N.M., 1996.

[111] F. Moukalled, L. Mangani, and M. Darwish. *The Finite Volume Method in Computational Fluid Dynamics: An Advanced Introduction with OpenFOAM® and Matlab*. Springer International Publishing, Cham, 2016.

[112] S. Muzaferija. A two-fluid navier-stokes solver to simulate water entry. *Proceeding of the 22nd symposium of naval hydrodynamics. Washington, DC*, 1998.

[113] T. S. Newman and H. Yi. A survey of the marching cubes algorithm. *Computers & Graphics*, 30(5):854–879, 2006.

[114] W. Noh and P. Woodward. SLIC (Simple line interface calculation). In A. van de Vooren and P. Zandbergen, editors, *Lecture Notes in Physics*, pages 336–340. Springer, 1976.

[115] OpenFOAM: The open-source CFD toolbox. https://www.openfoam.com/, v1812.

[116] M. Owkes and O. Desjardins. A computational framework for conservative, three-dimensional, unsplit, geometric transport with application to the volume-of-fluid (VOF) method. *Journal of Computational Physics*, 270:587–612, 2014.

[117] J. Palacios, J. Hernández, P. Gómez, C. Zanzi, and J. López. On the impact of viscous drops onto dry smooth surfaces. *Experimental Fluids*, 52:1449–1463, 2012.

[118] J. Palacios, J. Hernández, P. Gómez, C. Zanzi, and J. López. Experimental study of splashing patterns and the splashing/deposition threshold in drop impacts onto dry smooth solid surfaces. *Experimental Thermal and Fluid Science*, 44:571–582, 2013.

[119] ParaView. https://www.paraview.org/, v5.7.0.

[120] M. Pasandideh-Fard, Y. M. Qiao, S. Chandra, and J. Mostaghimi. Capillary effects during droplet impact on a solid surface. *Physics of Fluids*, 8(3):650–659, 1996.

[121] J. E. Pilliod and E. G. Puckett. Second-order accurate volume-of-fluid algorithms for tracking material interfaces. *Journal of Computational Physics*, 199(2):465–502, 2004.

[122] L. Pismen. Mesoscopic hydrodynamics of contact line motion. *Colloids and Surfaces A: Physicochemical and Engineering Aspects*, 206(1):11–30, 2002.

[123] E. G. Puckett, A. S. Almgren, J. B. Bell, D. L. Marcus, and W. J. Rider. A high-order projection method for tracking fluid interfaces in variable density incompressible flows. *Journal of Computational Physics*, 130(2):269–282, 1997.

[124] T. Qiang, X. Wang, and P. Sheng. Molecular scale contact line hydrodynamics of immiscible flows. *Physical Review E*, 68(016306):1–15, 2003.

[125] M. A. Quetzeri-Santiago, K. Yokoi, A. A. Castrejón-Pita, and J. R. Castrejón-Pita. Role of the dynamic contact angle on splashing. *Physical Review Letters*, 122:228001, 2019.

[126] E. S. Quintero, G. Riboux, and J. M. Gordillo. Splashing of droplets impacting superhydrophobic substrates. *Journal of Fluid Mechanics*, 870:175–188, 2019.

[127] K. Range and F. Feuillebois. Influence of surface roughness on liquid drop impact. *Journal of Colloid and Interface Science*, 203(1):16–30, 1998.

[128] W. Ren and W. E. Boundary conditions for the moving contact line problem. *Physics of Fluids*, 19(2):022101, 2007.

[129] Reyssat, M., Pépin, A., Marty, F., Chen, Y., and Quéré, D. Bouncing transitions on microtextured materials. *Europhysics Letters*, 74(2):306–312, 2006.

[130] C. M. Rhie and W. L. Chow. Numerical study of the turbulent flow past an airfoil with trailing edge separation. *AIAA Journal*, 21(11):1525–1532, 1983.

[131] G. Riboux and J. M. Gordillo. Experiments of drops impacting a smooth solid surface: A model of the critical impact speed for drop splashing. *Physical Review Letters*, 113:024507, 2014.

[132] W. J. Rider and D. B. Kothe. Reconstructing volume tracking. *Journal of Computational Physics*, 141(2):112–152, 1998.

[133] M. Rieber and A. Frohn. A numerical study on the mechanism of splashing. *International Journal of Heat and Fluid Flow*, 20(5):455–461, 1999.

[134] R. Rioboo, M. Marengo, and C. Tropea. Time evolution of liquid drop impact onto solid, dry surfaces. *Experiments in Fluids*, 33(1):112–124, 2002.

[135] R. Rioboo, C. Tropea, and M. Marengo. Outcomes from a drop impact on solid surfaces. *Atomization and Sprays*, 11(2):155–165, 2001.

[136] J. Roenby, H. Bredmose, and H. Jasak. A computational method for sharp interface advection. *Royal Society Open Science*, 3(11):160405, 2016.

[137] I. Roisman, L. Opfer, C. Tropea, M. Raessi, J. Mostaghimi, and S. Chandra. Drop impact onto a dry surface: Role of the dynamic contact angle. *Colloids and Surfaces A: Physicochemical and Engineering Aspects*, 322(1):183 – 191, 2008.

[138] I. V. Roisman, A. Lembach, and C. Tropea. Drop splashing induced by target roughness and porosity: The size plays no role. *Advances in Colloid and Interface Science*, 222:615–621, 2015. Reinhard Miller, Honorary Issue.

[139] I. V. Roisman, R. Rioboo, and C. Tropea. Normal impact of a liquid drop on a dry surface: model for spreading and receding. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 458(2022):1411–1430, 2002.

[140] H. Rusche. *Computational Fluid Dynamics of Dispersed Two-Phase Flows at High Phase Fractions*. Ph.D. thesis, Imperial College of Science, Technology and Medicine, 2002.

[141] Y. Sato and B. Ničeno. A new contact line treatment for a conservative level set method. *Journal of Computational Physics*, 231(10):3887–3895, 2012.

[142] R. Scardovelli and S. Zaleski. Direct numerical simulation of free–surface and interfacial flow. *Annual Review of Fluid Mechanics*, 31(1):567–603, 1999.

[143] R. Scardovelli and S. Zaleski. Analytical relations connecting linear interfaces and volume fractions in rectangular grids. *Journal of Computational Physics*, 164(1):228–237, 2000.

[144] R. Scardovelli and S. Zaleski. Interface reconstruction with least-square fit and split Eulerian-Lagrangian advection. *International Journal for Numerical Methods in Fluids*, 41(3):251–274, 2003.

[145] H. Scheufler and J. Roenby. Accurate and efficient surface reconstruction from volume fraction data on general meshes. *Journal of Computational Physics*, 383:1–23, 2019.

[146] ScienceDirect. https://www.sciencedirect.com/, 2021.

[147] C. Shen, C. Zhang, M. Gao, X. Li, Y. Liu, L. Ren, and A. S. Moita. Investigation of effects of receding contact angle and energy conversion on numerical prediction of receding of the droplet impact onto hydrophilic and superhydrophilic surfaces. *International Journal of Heat and Fluid Flow*, 74:89 – 109, 2018.

[148] Y. Shikhmurzaev. The moving contact line on a smooth solid surface. *International Journal of Multiphase Flow*, 19(4):589–610, 1993.

[149] Y. D. Shikhmurzaev. Mathematical modeling of wetting hydrodynamics. *Fluid Dynamics Research*, 13(1):45–64, 1994.

[150] H. Si. TetGen, a Delaunay-based quality tetrahedral mesh generator. *ACM Transactions on Mathematical Software*, 41(2), 2015.

[151] Š. Šikalo, M. Marengo, C. Tropea, and E. Ganić. Analysis of impact of droplets on horizontal surfaces. *Experimental Thermal and Fluid Science*, 25(7):503–510, 2002. International Thermal Science Seminar.

[152] Š. Šikalo, C. Tropea, and E. Ganić. Dynamic wetting angle of a spreading droplet. *Experimental Thermal and Fluid Science*, 29(7):795–802, 2005. Two Phase Flow.

[153] Š. Šikalo, H. Wilhelm, I. Roisman, S. Jakirlić, and C. Tropea. Dynamic contact angle of spreading droplets: Experiments and simulations. *Physics of Fluids*, 17(6):062103, 2005.

[154] M. Skarysz, A. Garmory, and M. Dianat. An iterative interface reconstruction method for PLIC in general convex grids as part of a Coupled Level Set Volume of Fluid solver. *Journal of Computational Physics*, 368:254–276, 2018.

[155] M. Skarysz, A. Garmory, and M. Dianat. An iterative interface reconstruction method for PLIC in general convex grids as part of a Coupled Level Set Volume of Fluid solver. *Journal of Computational Physics*, 368:254–276, 2018.

[156] J. H. Snoeijer and B. Andreotti. Moving contact lines: Scales, regimes, and dynamical transitions. *Annual Review of Fluid Mechanics*, 45(1):269–292, 2013.

[157] Springer Link. https://link.springer.com/, 2021.

[158] Y. Sui, H. Ding, and P. Spelt. Numerical simulations of flows with moving contact lines. *Annual Review of Fluid Mechanics*, 46(1):97–119, 2014.

[159] J. Sun, Q. Liu, Y. Liang, Z. Lin, and C. Liu. Three-dimensional VOF simulation of droplet impacting on a superhydrophobic surface. *Bio-Design and Manufacturing*, 2:10 – 23, 2019.

[160] B. Swartz. The second-order sharpening of blurred smooth border. *Mathematics of Computation*, 52(186):675, 1989.

[161] R. Tadmor. Line energy and the relation between advancing, receding, and young contact angles. *Langmuir*, 20(18):7659–7664, 2004.

[162] L. H. Tanner. The spreading of silicone oil drops on horizontal surfaces. *Journal of Physics D: Applied Physics*, 12(9):1473–1484, 1979.

[163] S. T. Thoroddsen, T. G. Etoh, K. Takehara, N. Ootsuka, and Y. Hatsuki. The air bubble entrapped under a drop impacting on a solid surface. *Journal of Fluid Mechanics*, 545:203–212, 2005.

[164] S. T. Thoroddsen and J. Sakakibara. Evolution of the fingering pattern of an impacting drop. *Physics of Fluids*, 10(6):1359–1374, 1998.

[165] G. Thürrner and C. A. Wüthrich. Computing vertex normals from polygonal facets. *Journal of Graphics Tools*, 3(1):43–46, 1998.

[166] C. Tropea and M. Marengo. The impact of drops on walls and films. *Multiphase Science and Technology*, 11(1):19–36, 1999.

[167] G. Tryggvason, B. Bunner, A. Esmaeeli, D. Juric, N. Al-Rawahi, W. Tauber, J. Han, S. Nas, and Y.-J. Jan. A front-tracking method for the computations of multiphase flow. *Journal of Computational Physics*, 169(2):708–759, 2001.

[168] G. Tryggvason, R. Scardovelli, and S. Zaleski. *Direct Numerical Simulations of Gas-Liquid Multiphase Flows*. Cambridge University Press, 2011.

[169] O. Ubbink and R. Issa. A method for capturing sharp fluid interfaces on arbitrary meshes. *Journal of Computational Physics*, 153(1):26–50, 1999.

[170] D. C. Vadillo, A. Soucemarianadin, C. Delattre, and D. C. D. Roux. Dynamic contact angle effects onto the maximum drop impact spreading on solid surfaces. *Physics of Fluids*, 21(12):122002, 2009.

[171] H. K. Versteeg and W. Malalasekera. *An Introduction to Computational Fluid Dynamics*. Pearson Education Limited, Harlow, England, 2007.

[172] O. V. Voinov. Hydrodynamics of wetting. *Fluid Dynamics*, 11(5):714–721, 1976.

[173] M.-J. Wang, F.-H. Lin, Y.-L. Hung, and S.-Y. Lin. Dynamic behaviors of droplet impact and spreading: Water on five different substrates. *Langmuir*, 25(12):6772–6780, 2009.

[174] R. Wenger. *Isosurfaces: Geometry, Topology and Algorithms*. A K Peters/CRC Press, 2013.

[175] M. Williams, D. Kothe, and E. Puckett. Convergence and accuracy of kernel-based continuum surface tension models. In *Proceedings of the 13th U.S. National Congress of Applied Mechanics*. 1998.

[176] K. G. Winkels, J. H. Weijs, A. Eddi, and J. H. Snoeijer. Initial spreading of low-viscosity drops on partially wetting surfaces. *Physical Review E*, 85:055301, 2012.

[177] M. Wörner. Numerical modeling of multiphase flows in microfluidics and micro process engineering: A review of methods and applications. *Microfluidics and Nanofluidics*, 12:841– 886, 2012.

[178] A. M. Worthington. On the forms assumed by drops of liquids falling vertically on a horizontal plate. *Proceedings of the Royal Society of London*, 25(171-178):261–272, 1876.

[179] T. Xavier, D. Zuzio, M. Averseng, and J.-L. Estivalezes. Toward direct numerical simulation of high speed droplet impact. *Meccanica*, 55(2):387– 401, 2020.

[180] F. Xiao, Y. Honma, and T. Kono. A simple algebraic interface capturing scheme using hyperbolic tangent function. *International Journal for Numerical Methods in Fluids*, 48(9):1023–1040, 2005.

[181] L. Xu. Liquid drop splashing on smooth, rough, and textured surfaces. *Physical Review Letters*, 75:056316, 2007.

[182] L. Xu, W. W. Zhang, and S. R. Nagel. Drop splashing on a dry smooth surface. *Phys Rev Lett*, 94(18):184505, 2005.

[183] A. Yarin. Drop impact dynamics: Splashing, spreading, receding, bouncing.... *Annual Review of Fluid Mechanics*, 38(1):159–192, 2006.

[184] K. Yokoi. Numerical studies of droplet splashing on a dry surface: triggering a splash with the dynamic contact angle. *Soft Matter*, 7:5120–5123, 2011.

[185] K. Yokoi. A practical numerical framework for free surface flows based on CLSVOF–method, multi-moment methods and density-scaled CSF model: Numerical simulations of droplet splashing. *Journal of Computational Physics*, 232(1):252 – 271, 2013.

[186] K. Yokoi. A density-scaled continuum surface force model within a balanced force formulation. *Journal of Computational Physics*, 278:221 – 228, 2014.

[187] D. L. Youngs. Time-dependent multi-material flow with large fluid distortion. In K. Morton and M. Baines, editors, *Numerical Methods for Fluid Dynamics*, pages 273–285. Academic Press, 1982.

[188] S. T. Zalesak. Fully multidimensional flux-corrected transport algorithms for fluids. *Journal of Computational Physics*, 31(3):335–362, 1979.

[189] Y. Zhang and J. Qian. Dual contouring for domains with topology ambiguity. *Computer Methods in Applied Mechanics and Engineering*, 217-220:34–45, 2012.

[190] Y. Zhang and J. Qian. Resolving topology ambiguity for multiple-material domains. *Computer Methods in Applied Mechanics and Engineering*, 247-248:166–178, 2012.