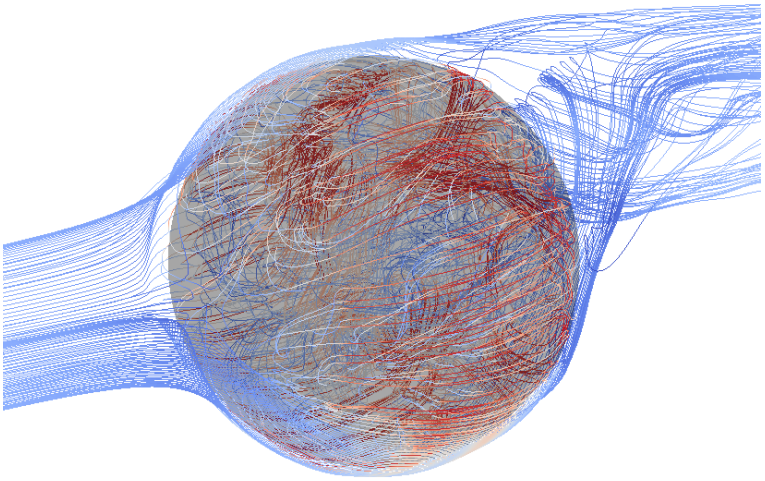
UPC CTTC

Numerical simulation of multiphase immiscible flow on unstructured meshes



Centre Tecnològic de Transferència de Calor
Departament de Màquines i Motors Tèrmics
Universitat Politècnica de Catalunya

Lluís Jofre Cruanyes

Doctoral Thesis

Numerical simulation of multiphase immiscible flow on unstructured meshes

Lluís Jofre Cruanyes

TESI DOCTORAL

presentada al

Departament de Màquines i Motors Tèrmics
E.T.S.E.I.A.T.
Universitat Politècnica de Catalunya

per a l'obtenció del grau de

Doctor per la Universitat Politècnica de Catalunya

Terrassa, Juliol 2014

Numerical simulation of multiphase immiscible flow on unstructured meshes

Lluís Jofre Cruanyes

Directors de la Tesi

Dr. Assensi Oliva Llena
Dr. Oriol Lehmkuhl Barba
Dr. Ricard Borrell Pol
Dr. Jesús Castro González

Tribunal Qualificador

Dr. Arthur E. P. Veldman
University of Groningen

Dr. Andrey V. Gorobets
Keldysh Institute of Applied Mathematics of RAS

Dr. Carlos D. Pérez-Segarra
Universitat Politècnica de Catalunya

This thesis is dedicated to

my parents, Teresa and Lluís,

my brother, Marc,

and my girl, Cristina.

The giant Ægir and the goddess Rán, who represent the sea in Norse mythology, had nine daughters whose names are poetic terms for different characteristics of ocean waves:

... The name of Ægir's wife is Rán, and they have nine daughters, as has before been written: Himinglæva, the wave reflecting the sky; Dúfa, the pitching wave; Blóðughadda, the blood-red color wave; Hefring, the surging wave; Uðr, the frothing wave; Hrönn, the grasping wave; Bylgja, the big wave; Dröfn, the foam wave; Kólga, the chilling wave ...

Skáldskaparmál section of Snorri Sturluson's Prose Edda, Iceland, 13th century.

Acknowledgements

I want to use these lines to thank all the people that has contributed, in one or other way, to the development of this thesis. The completation of this work has taken me more than five years, a time period in which many people have influenced my life and, consequently, this dissertation. Therefore, in the following lines I will try to highlight the ones that have contributed specially to the growth of this thesis, without forgetting to thank all the other ones and hoping that no one is omitted. In particular, I want to show my gratitude to ...

Prof. Assensi Oliva, head of the *Heat and Mass Transfer Technological Center* (CTTC), for depositing his trust on me as a PhD candidate, for the help along these years and for his enthusiasm for everything related to fluid mechanics. I also want to thank him for letting me focalize my work on the fields that I like the most and for giving me the freedom to decide the steps to take in most occasions.

Oriol Lehmkuhl for his magnificent supervising of all the theoretical, numerical and programming aspects that comprise this thesis, e.g., introducing me to the *TermoFluids* code, making me a better programmer by mimicking his implementations, teaching me how to deal with the discretization of convection-diffusion equations on three-dimensional unstructured meshes, making crucial decisions to facilitate the obtention of high quality numerical results valuable of being published in the scientific literature and more. In addition, I want to thank him for trusting me on the teaching of the *TermoFluids* code to the newcomers, as well on the supervision of some important improvements to the code.

Ricard Borrell for coming up with the solution to improve the parallelization of the Volume-of-Fluid method. I also want to show my gratitude for his meticulous supervision of the code used to implement the parallelization strategy, of the numerical tests on different supercomputers and of the writing of the paper, which has transformed me to a better programmer and writer.

Jesús Castro for helping me with the development and implementation of the Volume-of-Fluid method on three-dimensional unstructured meshes. In detail, his knowledge and experience provided me with the state-of-the-art necessary to start the development of the Volume-of-Fluid method proposed in this work, the choosing of the appropiate numerical tests and the writing of my first conference paper.

Néstor Balcázar for helping me create the basis of a multiphase flow solver in the *TermoFluids* code, in which now is possible to select between different interface-capturing methods and discretization schemes for the momentum equations. Moreover, for his tenacity in ensuring that all the modifications incorporated to the code provide results that are in accordance with benchmark results.

Xavi Trias, Jordi Ventosa and Ivette Rodríguez for introducing me to the idea of preserving the continuous properties of the differential equations when discretizing them, for developing a matrix solver based on a least-squares procedure, and for providing the knowledge and tools necessary to solve and analyze the turbulent flow over a circular cylinder, respectively. These three points have helped me publish my first paper regarding the conservation properties of the Navier-Stokes equations.

Ramiro Alba, Daniel Fernández, Octavi Pavon, Jorge Chiva and Guillem Colomer for providing solutions for an uncountable number of problems regarding the Linux system, the usage of the JFF, MareNostrum and Curie parallel computing systems, the correct implementation of codes and scripts using the C++ and Python languages, the utilization of the vim editor and the writing of documents in LaTeX language.

All the rest of the people of the CTTC for their collaboration and encouragement in the writing of this thesis. In special the main professors and researchers of the center: Carlos D. Pérez-Segarra, Joaquim Rigola and Carles Oliet, and the people that with the years have evolved from mates to friends: Joan Farnós, Joan Calafell, Joan López, Alex Sadurní, Roser Capdevila, Aleix Báez, Jordi Muela, Guillermo Oyarzún, Deniz Kızıldağ, Santiago Torras, Nicolás Ablanque and Pedro Galione. A special mention for Xiaofei Hou — remembered in my circle of friends and family for his endless happiness and curiosity — due to the great friendship that connects us although coming from two different worlds.

Profs. Arthur Veldman and Roel Verstappen for welcoming me during several months of 2012 in the *Computational Mechanics and Numerical Mathematics* research group of the *Johann Bernoulli Institute for Mathematics and Computer Science* in Groningen (The Netherlands), which, aside of being valuable for the theoretical part of this thesis, made me enjoy a life-inspiring experience and allowed me to meet young researchers from around the world: Henri van der Heiden, Peter van der Plas, Hande Kırbaş, Muhammad Younas, Jia (George) Liao, Ivan Vujacic and Javier González.

The Grant *Formacíon de Profesorado Universitario* (FPU) by the spanish *Ministerio de Educación, Cultura y Deporte* and the company *Termo Fluids S.L.* (TF) for their financial and technological support, respectively, which have been essentials in order to accomplish this work.

My friends and family for making me return back to earth every weekend and vacation period. In special my mother Teresa for showing me the importance of enjoying life and my father Lluís, the person who I guess that pushed me into this adventure many years ago when, as a kid, I used to help him organize alphabetically the exams that he had previously corrected. Also my brother Marc who, without knowing it, encouraged me to complete this work by finishing his thesis more than a year ago. Finally, Cristina for giving sense to my life, first as a close friend and now as my girl.

Abstract

The present thesis aims at developing a basis for the numerical simulation of multiphase flows of immiscible fluids. This approach, although limited by the computational power of the present computers, is potentially very important, since most of the physical phenomena of these flows often happen on space and time scales where experimental techniques are impossible to be utilized in practice. In particular, this research is focused on developing numerical discretizations suitable for three-dimensional (3-D) unstructured meshes, being the discretization on Cartesian grids as one particular case. This decision has been adopted in order to develop numerical algorithms adaptable to domains presenting boundaries with complex geometries. In addition, it is important to mention that within the *Heat and Mass Transfer Technological Center* (CTTC) research group, this thesis is the first attempt of discretizing these flows on 3-D unstructured meshes. Hence, rather than focusing on the study of the physics associated to these flows, most of the work is focused on the numerical discretization of the equations that govern them.

This work comprises seven chapters, the first one is an introduction to the type of flows considered, as well to the methodology used to study them. The next five chapters are the core of this dissertation, and encompass from the implementation of an interface-capturing method to the numerical resolution of the Navier-Stokes equations. In particular, the contents of this five chapters have been submitted or published in international journals and conferences, hence, they are written to be self-contained and only minor changes have been introduced with respect to the original papers. Consequently, some theoretical and numerical contents, as the advection of interfaces or the discretization of the Navier-Stokes equations, are repeated along them. The last chapter contains the concluding remarks, as well as ideas on how the present work could be continued. At the end, there are four appendices including material that may be useful in order to follow some parts of this work, but that has been placed apart so that the normal reading of the thesis is not disturbed.

In detail, the first chapter delimits the considered multiphase flows to the case in which the components are immiscible fluids — two or more fluids incapable of being mixed to form a homogeneous substance. In particular, the focus is placed on those cases where two or more continuous streams of different fluids are separated by interfaces, and hence, correspondingly named separated flows. Additionally, once the type of flow is determined, the chapter introduces the physical characteristics and the models available to predict its behavior, as well as the mathematical formulation that sustains the numerical techniques developed within this thesis.

The second chapter introduces and analyzes a new geometrical Volume-of-Fluid (VOF) method for capturing interfaces on 3-D Cartesian and unstructured meshes. The method reconstructs interfaces as first- and second-order piecewise planar ap-

proximations (PLIC), and advects volumes in a single unsplit Lagrangian-Eulerian (LE) geometrical algorithm based on constructing flux polyhedrons by tracing back the Lagrangian trajectories of the cell-vertex velocities. In this way, the situations of overlapping between flux polyhedrons are minimized.

Complementing the previous chapter, the third one proposes a parallelization strategy for the VOF method. The main obstacle is that the computing costs are concentrated in the interface between fluids. Consequently, if the interface is not homogeneously distributed, standard domain decomposition (DD) strategies lead to imbalanced workload distributions. Hence, the new strategy is based on a load balancing process complementary to the underlying domain decomposition. Its parallel efficiency has been analyzed using up to 1024 CPU-cores, and the results obtained show a gain with respect to the standard DD strategy up to $\sim 12\times$, depending on the size of the interface and the initial distribution.

In order to gain experience in the discretization of the Navier-Stokes equations on 3-D unstructured meshes, the fourth chapter describes and studies the case of single-phase flow to later extend it to the case of multiphase immiscible flow. In short, there are two main mesh discretizations for the calculation of these equations, the collocated and staggered schemes. Collocated schemes locate velocities at the same grid points as pressures, while staggered discretizations locate variables at different points within the mesh. One of the most important characteristics of the discretization schemes, aside from accuracy, is their capacity to discretely conserve kinetic energy, specially when solving turbulent flow. Hence, this chapter analyzes the accuracy and conservation properties of two particular collocated and staggered mesh schemes.

The extension of the numerical schemes suitable for the single-phase Navier-Stokes equations to the case of multiphase immiscible flow is developed in the fifth chapter. Particularly, while the numerical techniques for the simulation of turbulent flow have evolved to discretely preserve mass, momentum and, specially, kinetic energy, the mesh schemes for the discretization of multiphase immiscible flow, instead of focusing on the conservation properties, have evolved to improve their stability and robustness. Therefore, this chapter presents and analyzes two particular collocated and staggered mesh discretizations, able to simulate multiphase immiscible flow, which favor the discrete conservation of mass, momentum and kinetic energy.

Finally, the sixth chapter numerically simulates the Richtmyer-Meshkov (RM) instability of two incompressible immiscible liquids. This chapter, rather than being a detailed study of the physical phenomena of RM instabilities, is a general assessment of the numerical methods developed along this thesis. In particular, the instability has been simulated by means of a VOF method and a staggered mesh scheme. The corresponding numerical results have shown the capacity of the discrete system to obtain accurate results for the RM instability.

Contents

Abstract **v**

1 Introduction **1**
1.1 Multiphase flow of immiscible fluids 1
1.2 Models of flow prediction 3
1.3 Mathematical formulation 4
1.4 Objectives of the thesis 6
1.5 Outline of the thesis 8
References 9

2 Capturing interfaces on 3-D unstructured meshes: Volume-of-Fluid method **13**
2.1 Introduction 14
2.2 Volume-of-Fluid method 17
2.3 Interface reconstruction 18
2.3.1 Youngs method 19
2.3.2 Least-squares VOF interface reconstruction algorithm 20
2.4 Interface advection 20
2.4.1 Unsplit Lagrangian-Eulerian advection 20
2.4.2 Minimizing over/underlapping 22
2.4.3 Construction of flux polyhedrons 24
2.4.4 Truncation of flux polyhedrons 28
2.4.5 Correction of undershoots, overshoots and wisps 29
2.5 Numerical tests 31
2.5.1 Reconstruction tests 31
2.5.2 Advection tests 35
2.6 Conclusions 43
References 44

3 Parallelization of the Volume-of-Fluid method **51**
3.1 Introduction 52
3.2 Volume-of-Fluid method 54
3.2.1 Interface reconstruction 55
3.2.2 Interface advection 56
3.3 Parallelization strategy 58
3.3.1 Standard domain decomposition 58
3.3.2 New parallelization strategy 60
3.4 Numerical tests 72

3.5 Conclusions . . . 83
References . . . 84

4 Discretization of the Navier-Stokes equations on unstructured meshes 89
4.1 Introduction . . . 90
4.2 Discrete Navier-Stokes equations . . . 91
4.2.1 Collocated mesh scheme . . . 92
4.2.2 Staggered mesh scheme . . . 94
4.3 Conservation properties . . . 97
4.3.1 Mass conservation . . . 97
4.3.2 Momentum conservation . . . 98
4.3.3 Kinetic energy conservation . . . 101
4.4 Conservation and accuracy tests . . . 104
4.4.1 Rankine vortex . . . 104
4.4.2 Numerical tests of accuracy: exact sinusoidal function . . . 108
4.4.3 Turbulent flow over a circular cylinder at Re = 3900 . . . 110
4.5 Conclusions . . . 116
References . . . 118

5 Conservative discretization of multiphase immiscible flow 121
5.1 Introduction . . . 122
5.2 Motion of the interface between fluids . . . 123
5.3 Discrete Navier-Stokes equations . . . 125
5.3.1 Collocated mesh scheme . . . 126
5.3.2 Staggered mesh scheme . . . 128
5.4 Conservation properties . . . 130
5.4.1 Mass conservation . . . 131
5.4.2 Momentum conservation . . . 132
5.4.3 Kinetic energy conservation . . . 135
5.5 Conservation and accuracy tests . . . 138
5.5.1 Three-dimensional vortex . . . 138
5.5.2 Exact sinusoidal function . . . 143
5.5.3 Drag force on a spherical bubble in a turbulent pipe flow . . . 145
5.6 Conclusions . . . 153
References . . . 157

6 Numerical simulation of the Richtmyer-Meshkov instability 163
6.1 Introduction . . . 164
6.1.1 Richtmyer-Meshkov instability . . . 164
6.1.2 Method of interface-capturing . . . 164
6.1.3 Discretization of the Navier-Stokes equations . . . 165

6.2 Governing equations . . . 166
6.3 Numerical model . . . 167
6.3.1 Volume-of-Fluid method . . . 167
6.3.2 Unstructured staggered mesh scheme . . . 167
6.4 Numerical results . . . 169
6.4.1 Statement of the problem . . . 169
6.4.2 Development of the instability . . . 170
6.4.3 Amplitude measurements . . . 171
6.4.4 Velocity measurements . . . 174
6.4.5 Vorticity distributions . . . 174
6.5 Conclusions . . . 176
References . . . 177

7 Conclusions and further research **179**
7.1 Conclusions . . . 179
7.2 Further research . . . 183
References . . . 184

A Discretization of the convection-diffusion equation **187**
A.1 Convection-diffusion equation . . . 187
A.2 Finite-volume unstructured discretization . . . 188
A.3 Evaluation of the convection term . . . 188
A.4 Evaluation of the diffusion term . . . 189
References . . . 189

B Vector calculus identities **191**
B.1 Operator notation . . . 191
B.1.1 Nabla . . . 191
B.1.2 Gradient . . . 191
B.1.3 Divergence . . . 192
B.1.4 Curl . . . 192
B.1.5 Laplacian . . . 192
B.2 Operator identities . . . 192
B.2.1 Distributive properties . . . 192
B.2.2 Product rules . . . 193
B.2.3 Second derivatives . . . 193
B.3 Vector identities . . . 193
B.4 Integration identities . . . 194
B.4.1 Volume-surface integrals . . . 194
B.4.2 Surface-curve integrals . . . 194

C Parallel computing resources **195**
C.1 JFF supercomputer, Terrassa 195
C.2 MareNostrum supercomputer, Barcelona 195
C.3 Curie supercomputer, Paris 197

D Main publications in the context of this thesis **199**
D.1 Journal Papers 199
D.2 Conference Proceedings 200

1

Introduction

The main topic of this research is the numerical simulation of multiphase flow of immiscible fluids on unstructured meshes. In this introductory chapter, the physical characteristics and the models available to predict its behavior are presented, as well as the mathematical formulation that sustains the numerical techniques developed in this research. From these, the main objectives and outline of the thesis are derived.

1.1 Multiphase flow of immiscible fluids

In the context of this thesis, the term multiphase flow is used to refer to any fluid flow consisting of more than one phase or component. In particular, the focus is placed on those circumstances in which the components are immiscible fluids — two or more fluids incapable of being mixed to form a homogeneous substance. Technically, two or more immiscible fluids should be considered multi-fluid flow, but often are referred to as multiphase flow due to their similarity in behavior; see the work by Brennen [1] for a detailed explanation. Consequently, the flows considered have some level of phase or component separation at a scale well above the molecular level. This constraint still leaves an enormous spectrum of different multiphase flows. For instance, one could classify them according to the state of the different phases or components and, therefore, refer to gas/solid flows, gas/liquid flows, liquid/liquid flows and so on. Moreover, multiphase flows are also generally categorized depending on the components distribution: disperse or separated. The disperse flow consists of finite particles, drops or bubbles (the disperse phase) distributed in a connected volume of the continuous phase, while separated flow refers to the situation where two or more continuous streams of different fluids are separated by interfaces. This research work is mainly centered on the latter situation.

The separated flow type, usually named interfacial due to the interface that separates the phases, is found in a large variety of physical and biological phenomena, ranging from the prediction of atmospheric conditions to the study of blood flow,

and in engineering applications, as for example, cavitation in pumps and turbines, sprays or injection processes. In particular, a typical example of interfacial flow is the collision between liquid drops or the coalescence of gas bubbles; see Fig. 1.1. Understanding the flow in these situations not only involves the study of velocity and pressure fields in the air and water phases, but also of the interface between them. This latter part is much more difficult than the former because the interface is subject to a number of relevant physical phenomena, at scales much smaller than the typical sizes of drops or bubbles. For instance, a tough complication is the change in interface topology that occurs when the drops collide or the bubbles coalesce. This considerably complicates the physics and sharpens the requirements that the models of flow prediction must satisfy in order to resolve the motion in a satisfactory way.

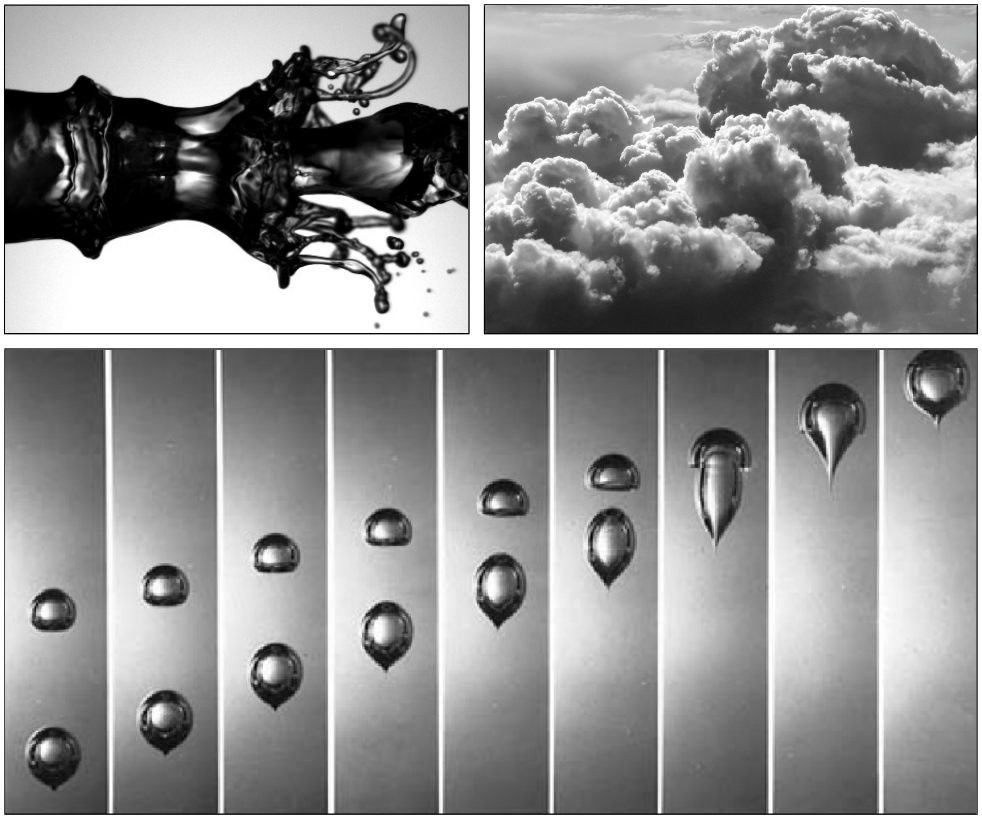


Figure 1.1: Examples of interfacial flow. Top left: Atomization of a liquid jet [2]. Top right: Atmospheric clouds. Bottom: Coalescence of two rising bubbles [3].

1.2 Models of flow prediction

A recurrent theme in fluid mechanics and, hence, also in the study of multiphase flow, is the need to model and predict the detailed behavior of flows and the phenomena that they manifest. Basically, there are three main approaches to explore them: (1) experimentally, through laboratory-scaled models equipped with appropriate instrumentation; (2) theoretically, using mathematical equations and approximations for the flow; and (3) computationally, setting up a discrete system of equations and numerically solving it on high performance computers. Clearly, there are some applications in which full-scale laboratory models are possible, but in many cases the laboratory model needs to be scaled in grand proportion or its cost becomes unaffordable, thus, in such situations the use of reliable theoretical models are an important tool for the analysis of the flow.

Most of these theoretical models are based on mathematical equations developed some centuries ago. However, the incapacity of finding their analytical solutions forced the scientists and engineers to simplify them: using hypothesis valid for specific fluid and flow characteristics and/or approximating them to parametric models. The simplification of the mathematical models has resulted useful for a large variety of low complexity cases, but as science and engineering have advanced the demanded problems have become more and more complex, till the point that these simplified models have failed in extrapolating reliable results. Fortunately, the appearance of high performance computing systems in the last decades has renovated, by means of using numerical techniques, the interest on the resolution of the full equations. For instance, complex turbulent, multiphase, compressible or combustion flows, unthinkable of being resolved just some decades ago, are now tackled by solving their discrete equivalent systems on supercomputers. This has attracted the interest from the technological companies, which are starting to rely on this approach —scaled to low cost computer clusters— to design and improve their products. Even so, the computer power required to solve most of the industrial applications is far beyond the present capability. However, in the case of affordable problems the use of this computational approach provides high quality data to the scientific community, facilitating the development of new accurate simplified models, which later may be used to characterize more complex cases.

In particular, multiphase flows may be mathematically modeled in different manners, depending if they are disperse or separated. In disperse flows, the trajectory model and two-fluid model are usually used. Trajectory models assess the motion of the disperse phase by following either the motion of the actual particles or the motion of larger representative particles. The details of the flow around each of the particles are incorporated into assumed drag, lift and moment forces acting on and altering the trajectory of those particles. Alternatively, two-fluid models treat the disperse phase as a second continuous phase mixed and interacting with the continuous phase. As a

consequence, conservation equations of mass, momentum and energy are required for each fluid, in which interaction terms that model the exchange of mass, momentum and energy between them are included. Thus, this approach involves difficult averaging processes in order to characterize the properties of the disperse phase. On the contrary, separated flows present many fewer issues, since one single set of equations is used to solve the flow in the different fluids, coupling them through appropriate kinematic and dynamic conditions at the interface. Particularly, this dissertation is mainly focused on separated flows, therefore, the complete formulation of the mathematical equations corresponding to this latter model are presented in the next section.

1.3 Mathematical formulation

The research undertaken in this thesis considers the separated multiphase flow as a set of subdomains, Ω_k, filled with individual phases, which together compose a single domain, Ω; see Fig. 1.2. These subdomains are separated by an interface, Γ, that determines a discontinuity of density, viscosity and pressure, as well of some other physical variables. In addition, the location of this discontinuity in three-dimensions is considered to be a smooth surface, which links the different phases by transferring momentum between them and, in the case of neglectable phase change, evolves according to the velocity field as

$$\frac{d\mathbf{x}_\Gamma}{dt} = \mathbf{u}(\mathbf{x}_\Gamma, t), \tag{1.1}$$

where $\mathbf{x}_\Gamma$ refers to the points on the phase interface.

The mathematical derivation exposed in this section is based on three general principles: the continuum fluid hypothesis, the hypothesis of sharp interfaces and the restriction of the effects from intermolecular forces. First, the approximation of a fluid as a continuum is a fundamental principle of fluid dynamics, and is valid in most practical cases — above 1nm for liquids in ambient conditions. Second, the transition from one phase to another occurs on very small scales, comparable to the scales of nanometers. Thus, the assumption that interfaces have an infinitesimal thickness is sufficiently correct. Third, the intermolecular forces, that play an important role in interface physics, are modelled by retaining just the effect from the surface tension force. Additionally to these three assumptions, this dissertation is restricted to incompressible flow of Newtonian fluids, since the fact of considering compressibility effects and non-Newtonian properties incorporate overwhelming complexities to the mathematical model.

Under these considerations and assumptions, the separated multiphase flow can be described for each phase k using the incompressible continuity and Navier-Stokes

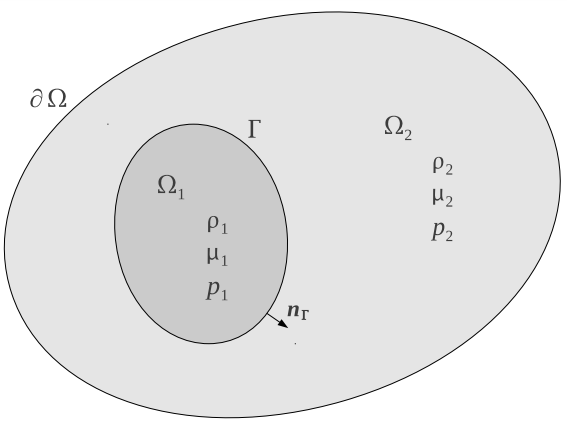


Figure 1.2: Schematic drawing representing a separated multiphase flow. The domain, Ω, is divided in two subdomains separated by an interface, Γ.

equations, written as

$$\nabla \cdot \mathbf{u} = 0, \tag{1.2}$$

$$\frac{\partial(\rho_k \mathbf{u})}{\partial t} + \nabla \cdot (\rho_k \mathbf{u}\mathbf{u}) = -\nabla p + \nabla \cdot (\mu_k [\nabla \mathbf{u} + \nabla^T \mathbf{u}]) + \mathbf{S}, \tag{1.3}$$

where ρ_k and μ_k are the constant density and dynamic viscosity of each phase k, t refers to time and $\mathbf{u}$, p and $\mathbf{S}$ represent velocity, pressure and a general source term, e.g., gravitational acceleration, $\rho_k \mathbf{g}$.

A single set of these equations, Eqs. 1.2 and 1.3, can be used to describe the flow in the whole domain, Ω, by introducing the jump of the different quantities across the interfaces [4]. This jump can be defined, regarding Fig. 1.2, as

$$[x]_\Gamma = x_2 - x_1, \tag{1.4}$$

where x_k denotes the limiting values of a variable x when an interface is approached from phase k. For example, this definition can be used to mathematically express the change of density and dynamic viscosity at the interface depicted in Fig. 1.2, resulting in

$$[\rho]_\Gamma = \rho_2 - \rho_1 \quad \text{and} \quad [\mu]_\Gamma = \mu_2 - \mu_1. \tag{1.5}$$

Moreover, in the absence of phase change, the velocity field is assumed to be continuous across the interface. This can be written in jump notation as

$$[\mathbf{u}]_\Gamma = 0. \tag{1.6}$$

In contrast, the existence of surface tension forces leads to a discontinuity in the normal stresses at the phase interface. This translates into a pressure jump that can be expressed as

$$[p]_\Gamma = \sigma\kappa + 2[\mu]_\Gamma \mathbf{n}_\Gamma{}^T \cdot \nabla\mathbf{u} \cdot \mathbf{n}_\Gamma, \tag{1.7}$$

where σ is the surface tension coefficient, κ is the curvature of the phase interface and $\mathbf{n}_\Gamma$ is the phase interface normal. Finally, making use of these jump conditions, the continuity and Navier-Stokes equations applicable to each individual phase, Eqs. 1.2 and 1.3, can be extended to a whole domain formulation as

$$\nabla \cdot \mathbf{u} = 0, \tag{1.8}$$

$$\frac{\partial(\rho\mathbf{u})}{\partial t} + \nabla \cdot (\rho\mathbf{u}\mathbf{u}) = -\nabla p + \nabla \cdot (\mu[\nabla\mathbf{u} + \nabla^T\mathbf{u}]) + \rho\mathbf{g} + \sigma\kappa\mathbf{n}_\Gamma\delta(\mathbf{x} - \mathbf{x}_\Gamma), \tag{1.9}$$

where ρ and μ vary across the interface, $\mathbf{x}$ represents a general point and δ is the Dirac delta function concentrated on the interface.

1.4 Objectives of the thesis

The numerical simulation, or Computational Fluid Dynamics (CFD), is a powerful tool to understand the physics of multiphase flows of immiscible fluids, as well to design or improve engineering equipments encompassing such type of flows. This approach relies on the discretization of the mathematical equations describing the flow —in this case presented in Sec. 1.3—, and takes advantage of the computational power of the modern parallel computing resources to solve, on a discrete basis, the resulting system of equations. At first sight, the limitations of the numerical techniques and computational power make it impossible to consider the solution of all the regimes of multiphase flow, forcing us to stay at rather low Reynolds (Re) and Weber (We) numbers. However, these methods are potentially very important. For one, the continuous improvement of the available computational power continuously extends the range of affordable problems. Second, and more destacable, the phenomena under consideration often happen on space and time scales where experimental techniques are difficult to be utilized in practice.

Until now, most of the computational techniques for the numerical simulation of flows with interfaces have been based on Cartesian discretizations of the spatial domain, which sometimes have been restricted to only two-dimensional (2-D) grids. The applicability of this Cartesian approach to simulate flows, although intuitive for discretization purposes, is restricted to very simple domains, which in most cases are mainly academic configurations. For instance, in recent years the primary atomization [5–7] and the motion of bubbles and drops [8–10] have been numerically simulated by means of Cartesian discretizations; see Fig. 1.3 for examples. This

contrasts with the state-of-the-art in turbulence modelling, which is already tackling three-dimensional (3-D) cases on the basis of unstructured mesh discretizations, as for example, the direct numerical simulations (DNS) of the turbulent flow around airfoils [11–13]. This difference in discretization schemes between turbulent and multiphase flows is logic, since till now many researchers have been occupied with the numerical solution of laminar and turbulent incompressible flow — which makes sense due to its importance in engineering problems —, and consequently, not much effort has been paid to the solution of multiphase flow. Therefore, due to its youthfulness, the numerical simulation of flows with interfaces has been addressed mainly on Cartesian grids. However, due to the recent maturation gained on turbulence modelling, as well on the multiphase simulation on 3-D Cartesian grids, various research groups have started to focus their attention on the development of discretizations suitable for multiphase flow on 2-D and 3-D unstructured meshes. For example, in the past year the research group *Computational Thermo-Fluids Laboratory* [14], led by Olivier Desjardins at *Cornell University* [15], presented one of the first DNS of primary atomization in complex geometries [16], demonstrating that the numerical simulation of multiphase flows on unstructured meshes is feasible.

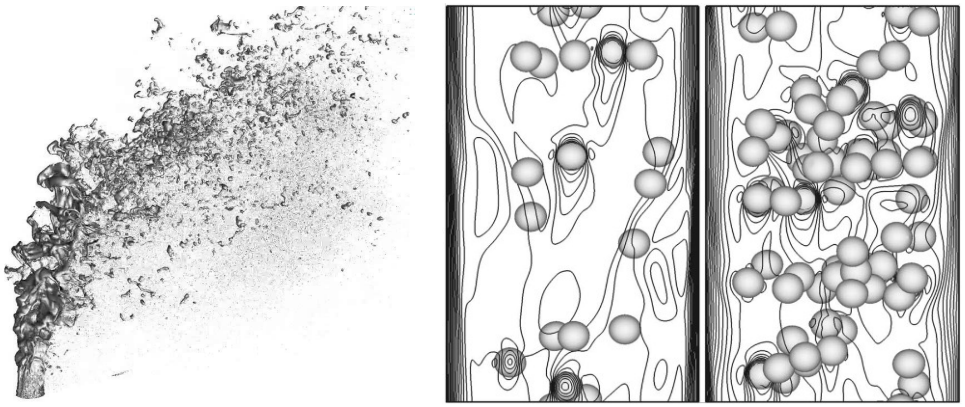


Figure 1.3: Examples of numerical simulations. Left: Atomization of a turbulent liquid jet [14]. Right: Bubble distribution in a vertical channel for two different cases of void fraction [17].

Therefore, considering the actual state-of-the-art in the field of numerical simulation of multiphase flow, the main objectives of this thesis are:

- Develop a suitable formulation for the numerical simulation of multiphase flow of immiscible fluids on complex geometries.
- Implement the numerical formulation on the basis of a parallel high perfor-

mance computing (HPC) platform suitable for 3-D unstructured meshes.

- Conduct computational simulations of the phenomena comprising the physics of multiphase flows and analyze the resulting numerical data.

To do so, the equation of interface motion, Eq. 1.1, and the equations of continuity, Eq. 1.8, and Navier-Stokes, Eq. 1.9, have been discretized and introduced into the *TermoFluids* (TF) CFD platform [18]. This platform consists of a HPC CFD code suitable for 3-D unstructured meshes, as well for Cartesian grids, developed and maintained by *Termo Fluids S.L.* [19], which is a spin-off from the *Heat and Mass Transfer Technological Center* (CTTC) [20] of the *Technical University of Catalonia* (UPC) [21]. The initial purpose of this software was to solve turbulent flows on complex geometries by using unstructured grids — see the PhD theses of Oriol Lehmkuhl [22] and Ricard Borrell [23] for details —, but now is being extended in order to simulate other types of fluid dynamics problems, such as: compressible and low-mach number flows, solid-liquid phase change, combustion, moving grid, adaptive mesh refinement (AMR) and, in particular, multiphase flow. In detail, this research has implemented in the TF code: (1) a 3-D Volume-of-Fluid (VOF) method [24], suitable for unstructured meshes, in order to capture the motion of the interface; and (2) a symmetry-preserving discretization [25] of the Navier-Stokes equations, on 3-D unstructured meshes, able to calculate the solution of the velocity and pressure fields in the presence of multiphase flow with interfaces.

1.5 Outline of the thesis

As has been mentioned, this thesis aims at creating a numerical basis for the simulation of multiphase immiscible flows within the TF CFD code. To accomplish this goal, first, the physical phenomena that characterize this type of flows, together with the mathematical formulation that describes them, are explained in Chapter 1, **Introduction**.

The next two chapters are devoted to the development of a VOF method able to capture interfaces on 3-D unstructured meshes. In detail, Chapter 2, **Capturing interfaces on 3-D unstructured meshes: Volume-of-Fluid method**, accurately presents the geometrical algorithm of the VOF method implemented, as well as the numerical results of the accuracy tests performed. Coordinately, the design of the parallelization strategy utilized to improve the computational performance of the interface capturer is described in Chapter 3, **Parallelization of the Volume-of-Fluid method**.

In the following chapters, the discretization of the Navier-Stokes equations on 3-D unstructured meshes is meticulously studied with the aim to analyze its discrete conservation properties. First, Chapter 4, **Discretization of the Navier-Stokes equations on unstructured meshes**, analyzes the equations in the case where no interfaces

are found in the domain. Next, a similar study is performed in Chapter 5, **Conservative discretization of multiphase immiscible flow**, but considering the case where different fluids separated by interfaces are present.

Finalizing this thesis, the numerical simulation of the Richtmyer-Meshkov (RM) instability of two incompressible immiscible liquids is performed in Chapter 6, **Numerical simulation of the Richtmyer-Meshkov instability**, permitting a general assessment of the different methods developed. As final point, conclusions and future research are highlighted in Chapter 7, **Conclusions and further research**.

Moreover, four appendices are included to complement some parts of this thesis. In particular, an introduction to the discretization of partial differential equations (PDE) on unstructured meshes is presented in Appendix A, **Discretization of the convection-diffusion equation**. Some vector calculus identities, important for the discretization and study of PDEs, are listed in Appendix B, **Vector calculus identities**. The parallel computing systems utilized to perform the numerical simulations of this thesis are shown in Appendix C, **Parallel computing resources**. Finally, the publications resulting from this research are listed in Appendix D, **Main publications in the context of this thesis**.

References

[1] C. E. Brennen. *Fundamentals of Multiphase Flows*. Cambridge University Press, 2005.

[2] E. Villermaux. Mixing and Spray Formation in Coaxial Jets. *Journal of Propulsion and Power*, 14:807–817, 1998.

[3] M. Samimy, K. S. Breuer, L. G. Leal, and P. H. Steen. *A Gallery of Fluid Motion*. Cambridge University Press, 2003.

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

[5] D. Fuster, A. Bagué, T. Boeck, L. Le Moyne, A. Leboissetier, S. Popinet, P. Ray, R. Scardovelli, and S. Zaleski. Simulation of Primary Atomization with an Octree Adaptive Mesh Refinement and VOF Method. *International Journal of Multiphase Flow*, 35:550–565, 2009.

[6] M. Herrmann. Detailed Numerical Simulations of the Primary Atomization of a Turbulent Liquid Jet in Crossflow. *Journal of Engineering for Gas Turbines and Power*, 132:061506–10, 2010.

[7] J. Shinjo and A. Umemura. Simulation of Liquid Jet Primary Breakup: Dynamics of Ligament and Droplet Formation. *International Journal of Multiphase Flow*, 36:513–532, 2010.

[8] W. Dijkhuizen, I. Roghair, M. S. Annaland, and J. A. M. Kuipers. DNS of Gas Bubbles Behaviour Using an Improved 3D Front Tracking Model - Drag Force on Isolated Bubbles and Comparison with Experiments. *Chemical Engineering Science*, 64:1415–1426, 2010.

[9] J. H. Seo, K. L. Sanjiva, and G. Tryggvason. Investigation and Modeling of Bubble-Bubble Interaction Effect in Homogeneous Bubbly Flows. *Physics of Fluids*, 22:063302, 2010.

[10] Y. Pengtao and Y. Renardy. Spontaneous Penetration of a Non-Wetting Drop into an Exposed Pore. *Physics of Fluids*, 25:052104, 2013.

[11] H. Shan, J. Li, and L. Chaoqun. Direct Numerical Simulation of Flow Separation Around a NACA 0012 Airfoil. *Computers & Fluids*, 34:1096–1114, 2005.

[12] L. E. Jones, R. D. Sandberg, and N. D. Sandham. Direct Numerical Simulations of Forced and Unforced Separation Bubbles on an Airfoil at Incidence. *Journal of Fluid Mechanics*, 602:175–207, 2008.

[13] I. Rodríguez, O. Lehmkuhl, R. Borrell, and A. Oliva. Direct Numerical Simulation of a NACA 0012 in Full Stall. *International Journal of Heat and Fluid Flow*, 43:194–203, 2013.

[14] Computational Thermo-Fluids Laboratory. Cornell University. Webpage: http://ctflab.mae.cornell.edu/index.html.

[15] Cornell University. Webpage: http://www.cornell.edu/.

[16] O. Desjardins, J. O. McCaslin, M. Owkes, and P. Brady. Direct Numerical and Large-Eddy Simulation of Primary Atomization in Complex Geometries. *Atomization and Sprays*, 23:1001–1048, 2013.

[17] J. Lu and G. Tryggvason. Numerical Study of Turbulent Bubbly Downflows in a Vertical Channel. *Physics of Fluids*, 18:103302, 2006.

[18] O. Lehmkuhl, C. D. Pérez-Segarra, R. Borrell, M. Soria, and A. Oliva. TERMOFLUIDS: A New Parallel Unstructured CFD Code for the Simulation of Turbulent Industrial Problems on Low Cost PC Cluster. In *Proceedings of the Parallel CFD Conference*, pages 1–8, 2007.

[19] Termo Fluids. Webpage: http://www.termofluids.com.

[20] Heat and Mass Transfer Technological Center. Technical University of Catalonia. Webpage: http://www.cttc.upc.edu.

[21] Technical University of Catalonia. Webpage: http://www.upc.edu.

[22] O. Lehmkuhl. Numerical Resolution of Turbulent Flows on Complex Geometries. PhD Thesis, Technical University of Catalonia, 2012.

[23] R. Borrell. Parallel Algorithms for Computational Fluid Dynamics on Unstructured Meshes. PhD Thesis, Technical University of Catalonia, 2012.

[24] C. W. Hirt and B. D. Nichols. Volume of fluid (VOF) Method for the Dynamics of Free Boundaries. *Journal of Computational Physics*, 39:201–225, 1981.

[25] R. W. C. P. Verstappen and A. E. P. Veldman. Symmetry-Preserving Discretization of Turbulent Flow. *Journal of Computational Physics*, 187:343–368, 2003.

2

Capturing interfaces on 3-D unstructured meshes: Volume-of-Fluid method

Main contents of this chapter have been published in:

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.

Abstract. A new geometrical Volume-of-Fluid (VOF) method for capturing interfaces on three-dimensional (3-D) Cartesian and unstructured meshes is introduced. The method reconstructs interfaces as first- and second-order piecewise planar approximations (PLIC), and advects volumes in a single unsplit Lagrangian-Eulerian (LE) geometrical algorithm based on constructing flux polyhedrons by tracing back the Lagrangian trajectories of the cell-vertex velocities. In this way, the situations of overlapping between flux polyhedrons are minimized, consequently, the accuracy in the solution of the advection equation is improved by minimizing the creation of overshoots (volume fractions over one), undershoots (volume fractions below zero) and wisps (fluid in void regions or vice versa). However, if not treated carefully, the use of cell-vertex velocities may result in the construction of flux polyhedrons that contain nonplanar faces and that do not conserve volume. Therefore, this work explains in detail a set of geometric algorithms necessary to overcome these two drawbacks. In addition, the new VOF method is analyzed numerically on 3-D Cartesian and unstructured meshes, first, by reconstructing the interface of spherical geometries and, second, by evaluating the final advection result of a sphere placed in a rotation, shear and deformation field.

2.1 Introduction

The numerical simulation of immiscible multi-fluid flows is currently an object of intense research, since it is present in fields as varied as engineering, fundamental physics or geophysics. These kind of flows are called interfacial due to the thin region, named interface, that separates them. Typical examples are the simulation of sprays, jets and injection processes or the study of hydrodynamic phenomena such as movement of bubbles, breakup of drops and wave motion. So far, many different strategies exist to calculate interface motion, most of them recapitulated in the work by Scardovelli and Zaleski [1]. In general, all of them may be classified in two main groups: interface-tracking and interface-capturing. On the one hand, the interface-tracking approaches chase the interface as it moves: (1) defining the interface as a boundary between two subdomains of a moving grid [2–5] or (2) following the Lagrangian trajectories of massless particles [6–10]. On the other hand, the interface-capturing approaches describe the motion of the interface by embedding the different fluids into a fixed grid with the help of scalar values. In particular, from this last group, the two main options of choice are the Volume-of-Fluid (VOF) [11–14] and Level-Set (LS) [15–18] methods, as well as algorithms based on combinations of both, like the Coupled Level-Set/Volume-of-Fluid method [19–21].

Each of the methods classified in the above paragraph excels in the simulation of a particular interfacial problem. For instance, in Arbitrary Lagrangian-Eulerian (ALE) methods [3–5], the mesh is updated continuously to fit the variation of the interface, which is a good approach for fluid-particle flows [22, 23], simplifying the analysis near the interface. Another example are the Front-Tracking (FT) methods [8–10], these employ separate sets of discrete points to represent each of the individual interfaces, resulting accurate for the simulation of dense bubbly flow [24–26], since large numbers of points can be used on the interface and merging of bubbles can be explicitly controlled. However, considering general multi-fluid flows, where large interface topology changes may be found, both ALE and FT approaches result in fairly complex implementations. On the contrary, interface-capturing methods, like VOF and LS, have the natural ability to handle topological changes, which for many applications, such as atomization [18,27] or breakup and coalescence of drops [28,29], is an important advantage. In particular, both VOF and LS methods, in order to locate interfaces, embed the different fluids into a static mesh by means of fluid volume fraction values; i.e., a value between 0 and 1 in cells containing the interface, and values 0 or 1 in cells completely empty or filled of a particular fluid, respectively. The main difference is that VOF methods do it in a discontinuous manner while LS methods describe the interface as the zero level-set of an auxiliary function. Therefore, the principal advantage of VOF methods over LS ones is their inherent conservation of volume, however, LS methods are more accurate in calculating geometric quantities such as curvature and normal vectors. In consequence, from all these options, this

paper is focused on the VOF method, since it represents one of the most accurate options to capture interfaces and their complex deformation, including breakups and coalescence, while complying with the volume preservation constraint.

In the VOF method, the temporal evolution of the volume fraction function is governed by an advection equation. A first option is to solve it by means of standard numerical convection schemes, resulting appropiate for grids ranging from two-dimensional (2-D) Cartesian to three-dimensional (3-D) unstructured. However, because the volume fraction is a discontinuous function, this approach easily diffuses the interface when is advected, which contrary, should remain sharp. This shortcoming can be minimized, but not completely overcome, by combining high-resolution and compressive schemes [30–33]. A second option is to advect volume fractions based on a reconstructed interface determined by the volume fraction field (volume tracking). Hence, interface reconstruction is an important part of any volume tracking VOF method. In fact, given its importance, the methods for interface reconstruction have evolved from the original simple line interface calculation (SLIC) and piecewise linear interface calculation (PLIC) for 2-D Cartesian grids —accurately detailed in the work by Rider and Kothe [12]— to a large variety of equivalent methods suitable for 3-D Cartesian and unstructured grids. In particular, considering the two main interface reconstruction methods, the PLIC is the favored one in many implementations, since the use of lines/planes (2-D or 3-D) to reconstruct interfaces provides improved results. Contrary, the SLIC method is a better option to reconstruct interfaces in the case of cells containing more than two fluids. Moreover, other methods involving parabolic [13], spline [34,35] and least-square [36] reconstructions have also been presented, however, the complexities of their implementations do not generally compensate their improvements in accuracy. Focusing on PLIC methods, modern implementations are based on the first-order accurate interface reconstruction by Youngs [37], which positions each reconstructed interface line/plane, defined by a slope/normal and intercept, within the cell. The slope/normal of the line/plane is given by the gradient of the volume fractions, and the intercept follows from invoking volume conservation. Starting from this implementation, the following ones have evolved to provide second-order accuracy on Cartesian meshes [38], second-order accuracy with efficient algorithms on 2-D [39] and 3-D [40] Cartesian meshes, and first-order accuracy on spherical coordinates [41]. Similarly, in the case of unstructured meshes, the methods have evolved from 2-D first- [42] and second-order [43,44] implementations to 3-D first- [45] and second-order [46–48] ones.

On the other hand, based on the reconstructed interface, several schemes have been developed to advect the volume fractions. In general, geometrical advection schemes may be categorized in directional (operator) split and unsplit algorithms. Split algorithms integrate the advection equation along each coordinate direction, while unsplit algorithms integrate the equation in a single computational step. There-

fore, split advection is much simpler to implement, but requires a reconstruction and advection step for each dimension. However, in the case of unstructured meshes, operator split schemes cannot be used, leaving unsplit advection algorithms as the only option. Moreover, as referred in the work by Shahbazi et al. [49], advection schemes may also be classified depending if the advective fluxes are defined explicitly as advection flux volumes across mesh cell faces (Eulerian) or obtained as remapped volumes from polyhedron intersection operations (Lagrangian-Eulerian). In general, the first implementations on Cartesian grids were based on the Eulerian approach; see the work by Rider and Kothe [12] for descriptions of 2-D Cartesian split and unsplit Eulerian advection algorithms. However, the implementation of Eulerian schemes on unstructured grids is difficult, since it is very complicated to compute the fluid volume fluxes on such geometries. Otherwise, the Lagrangian-Eulerian (LE) advection method, which consists of three typical stages: a Lagrangian projection, an interface reconstruction and a remapping, is suitable for both Cartesian and unstructured meshes. Hence, recently published volume tracking VOF implementations, generally thought for unstructured meshes, rely on unsplit LE advection schemes. For example, typical works on unsplit LE advection algorithms for 2-D Cartesian and unstructured meshes are the ones by López et al. [34], and Shahbazi et al. [49] and Ashgriz et al. [50], respectively. Further, in the case of 3-D meshes the most relevant recent works are the ones by Liovic et al. [14] and Hernández et al. [35] for Cartesian discretizations, and the ones by Ivey and Moin [51] and Marić et al. [45] for unstructured grids.

Given the proven capability of VOF methods to handle large topological changes and conserve volume, specially on 2-D Cartesian grids, the aim of this paper is to develop a geometrical VOF method suitable for the case of 3-D unstructured discretizations, since it is a situation that has not been much explored yet. In fact (to our knowledge), just our previous work [52] and the recent papers by Ivey and Moin [51] and Marić et al. [45] deal with the implementation (reconstruction and advection) of volume tracking VOF methods on 3-D unstructured meshes. The topology of unstructured meshes dictates the interface reconstruction and advection algorithms, as well as the underlying geometrical operations. In particular, this work reconstructs interfaces by means of the 3-D unstructured versions of the first-order Parker and Youngs [53] and second-order LVIRA [38] PLIC methods. As for the time integration of the advection equation, this paper proposes to implement a 3-D unstructured unsplit LE geometrical algorithm based on the Lagrangian trajectories of the cell-vertex velocities. In detail, the advection strategy is similar to the one presented by Marić et al. [45], since both approaches choose to calculate the advection volume fluxes by tracing back the Lagrangian trajectories of the cell-vertex velocities in order to minimize over/underlapping situations. However, some differences between both papers can be found. For instance, in the work by Marić et al. the

method to ensure that the volumes of the swept-face polyhedra equal the ones defined by the advection equation relies on an iterative approach, while in this work the solution is given by an efficient analytical method similar to the one proposed by Hernández et al. [35]. In addition, this work presents results of advection tests on 3-D Cartesian and unstructured meshes, while Marić et al. just provide results on Cartesian grids. All these similarities and differences between recent papers, along with the details of the advection method, will be extensively explained in the following sections.

The main purpose of this paper is to detail accurately the implementation of a geometrical VOF method suitable for 3-D unstructured grids. Hence, Sec. 2.2 exposes the mathematical formulation of the VOF method on unstructured meshes. Next, Secs. 2.3 and 2.4 explain in detail the geometrical implementations of the interface reconstruction and advection methods. Following, Sec. 2.5 presents numerical results of reconstruction (sphere and hollowed sphere) and advection (rotational, shearing and deformation flow) accuracy tests. Finally, conclusions are drawn in Sec. 2.6.

2.2 Volume-of-Fluid method

In VOF methods, the fluids interface is captured by embedding it into a static grid with the help of scalar values. In particular, a volume fraction scalar field, C_k, is defined for each fluid k, determining the fraction of volume that occupies within each computational cell. Basically, $C_k = 0$ for cells that do not contain fluid k, $C_k = 1$ for cells that only contain the k'th fluid and $0 < C_k < 1$ if part but not all of a cell's volume is occupied by the k'th fluid. These cells in which different fluids coexist are referred as interface or mixed cells. Indeed, C_k can be defined as the normalized integral of the volume fraction Heaviside step function, $C_k(\mathbf{x}, t)$, defined as

$$C_k(\mathbf{x}, t) = \begin{cases} 1 & \text{if there is fluid } k \\ 0 & \text{otherwise,} \end{cases} \tag{2.1}$$

where $\mathbf{x}$ is a position in space and t refers to time instant. Therefore, for each cell c, with volume V_c, its k'th fluid volume fraction at time t is evaluated as

$$C_k[c, t] = \frac{\int C_k(\mathbf{x}, t) dV_c}{V_c}. \tag{2.2}$$

Assuming that the fluids are immiscible and that their movement is defined by a unique velocity field, i.e., $u^k = \mathbf{u}$ for each fluid k, the interface motion can then be captured by solving the respective conservation equation

$$\frac{\partial C_k}{\partial t} + \nabla \cdot (C_k \mathbf{u}) = 0. \tag{2.3}$$

As previously commented in the introduction, Eq. 2.3 might be discretized as a standard advection equation. For instance, Darwish and Moukalled [32] propose to discretize the transient term by means of a second-order Crank-Nicholson scheme and to use a special convection scheme, named STACS, that switches between SUPERBEE [54] (compressive) and STOIC [55] (high-resolution) schemes. This solution may be computationally fast and easy to implement on 3-D unstructured meshes, however, it tends to produce diffuse interfaces as they are integrated in time. Another more common approach is the volume tracking method, which advects volume fractions based on a reconstructed interface determined by the volume fraction field. In a summarized form, this method solves the advection equation by using the interface reconstruction to geometrically calculate the volumetric fluxes of fluid k across mesh cell faces. Indeed, this approach preserves the interface sharpness, but requires complex and computationally expensive geometrical routines to construct and truncate flux volumes. In particular, applying the divergence theorem and using the first-order Euler explicit time scheme, the volume tracking VOF method presented in this paper discretizes Eq. 2.3 for each cell as

$$C_k^{n+1} - C_k^n + \frac{1}{V_c} \sum_{f \in F(c)} V_{k,f}^n = 0, \tag{2.4}$$

where the superscript n refers to the discrete time level and $V_{k,f}$ is the volumetric flow of fluid k across face f; see Fig. 2.1. In order to calculate $V_{k,f}$, two consecutive steps are required: interface reconstruction and advection. First, the interface is reconstructed by means of PLIC methods. In particular, two different methods are implemented on the basis of 3-D unstructured meshes: (1) least-squares gradient (LSG) approach of the Parker and Youngs method [53] and (2) least-squares Volume-of-Fluid interface reconstruction algorithm (LVIRA) [38]. Second, once the interface has been reconstructed, the advection step geometrically constructs volumetric flows (polyhedrons) at mesh cell faces and, later, cuts them by the reconstructed interface in order to compute the amount of fluid k across the faces, $V_{k,f}$. As a novelty, this work presents a 3-D unstructured unsplit LE geometrical algorithm, that tries to minimize the over/underlapping situations by constructing flux polyhedrons from cell-vertex velocities. Both steps are fully described in the following sections.

2.3 Interface reconstruction

The interface separating different fluids is a thin region that can be approximated to a surface. Therefore, the interface reconstruction step tries to find a geometric shape that best approximates this surface. In this work, the PLIC approach is chosen to represent interfaces, thus, in the case of 3-D grids the interfaces are represented for

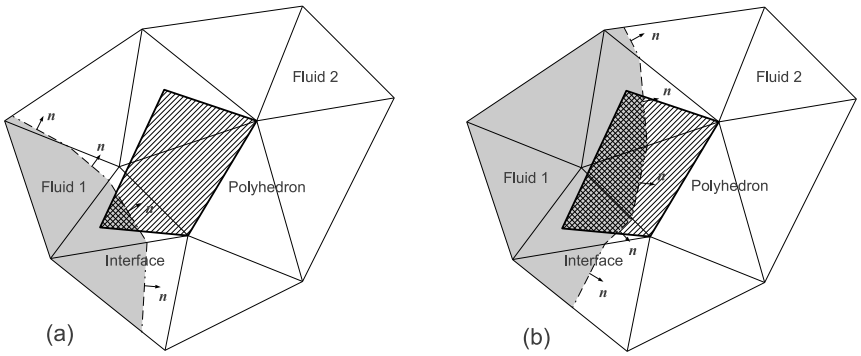


Figure 2.1: Representation on a 2-D unstructured grid of two situations, a and b, sharing the same total volumetric flow (hatched area), but presenting different values of $V_{k,f}$ (double hatched area).

each cell by a plane

$$\mathbf{n} \cdot \mathbf{x} - d = 0, \tag{2.5}$$

where $\mathbf{x}$, $\mathbf{n}$ and d are the coordinates of a point on the plane, the unit normal of the plane and the signed distance from the origin to the plane, respectively.

The principal reconstruction constraint is local volume conservation, i.e., the reconstructed interface must truncate the cell verifying

$$C_k = \frac{V_k}{V_c}, \tag{2.6}$$

where C_k and V_k are the volume fraction and volume of fluid k for cell c, respectively, and V_c is the volume of the entire cell. Hence, given a normal $\mathbf{n}$, it is necessary to find the constant d in Eq. 2.5 such that the intersection of the corresponding half-space and cell satisfies Eq. 2.6.

Since a unique interface configuration does not exist, the interface geometry must be inferred based on local data and the assumptions of a particular algorithm. Particularly, PLIC methods differ in how the normal $\mathbf{n}$ is computed, but for a given normal, d is uniquely defined from Eq. 2.6. A commonly used method to efficiently find d is the Brent's root-finding algorithm [56], since it is a combination of the bisection, secant and inverse quadratic interpolation methods.

2.3.1 Youngs method

In the Youngs interface reconstruction method [53], the plane normal is computed by approximating it to the normalized gradient of the volume fraction scalar field, C_k, as

$$\mathbf{n} = \frac{-\nabla C_k}{|\nabla C_k|}. \tag{2.7}$$

In particular, considering the case of 3-D unstructured meshes consisting of generalized polyhedra, it is convenient to use a vertex-connectivity least-squares gradient procedure [57].

2.3.2 Least-squares VOF interface reconstruction algorithm

In the least-squares VOF interface reconstruction algorithm (LVIRA) [38], the interface normal $\mathbf{n}$ is computed by minimizing the error functional

$$E(\mathbf{n}) = \left(\sum_{nb \in C(c)} ({C_{k,nb}}^{ref} - C_{k,nb}(\mathbf{n}))^2 \right)^{1/2}, \tag{2.8}$$

where subscript nb refers to the cells around cell c that share a common vertex with it (neighbor cells), $C^{ref}_{k,nb}$ is the neighbor cell reference volume fraction and $C_{k,nb}(\mathbf{n})$ is the neighbor cell actual (reconstructed) volume fraction taken by extending the interface of central cell c, under the constraint that the corresponding plane exactly reproduces the volume fraction in the cell under consideration; see Fig. 2.2.

The normal $\mathbf{n}$, in three dimensions, can be described by polar coordinates, hence, the LVIRA implementation requires an algorithm for the minimization of a nonlinear function of two variables. In the present work, the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm [56] is used.

The LVIRA method requires more computational resources than the Youngs one, since for each cell an error has to be minimized, but it is second-order accurate (reconstructs planar interfaces exactly) while Youngs is just first-order.

2.4 Interface advection

2.4.1 Unsplit Lagrangian-Eulerian advection

As previously said in the introduction, this work focuses on developing a geometrical advection scheme suitable for 3-D unstructured meshes, hence, in order to efficiently deal with unstructured grids, as well as Cartesian, the choice of an unsplit LE advection method is preferred. In addition, making use of an unsplit algorithm, although it implies more complicated geometric operations: (1) results in a faster implementation than split advection, since it just requires one interface reconstruction and time integration per time step; and (2) avoids the generation of direction split

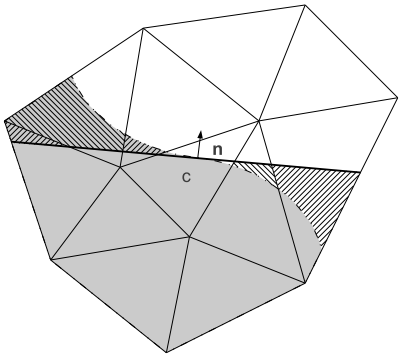


Figure 2.2: Representation on a 2-D unstructured mesh of the LVIRA error functional. Given a central cell c, its plane interface reconstruction (boldface line) is extended to the neighbor cells and for each one ${C_{k,nb}}^{ref} - C_{k,nb}(\mathbf{n})$ is calculated (hatched areas) and added to the LVIRA error $E(\mathbf{n})$.

errors [12, 42]. Therefore, following the LE strategy, the fluid k volume fractions are advected forward in time, as described in Eq. 2.4, by geometrically calculating the k'th volumetric fluxes, $V_{k,f}$, across mesh cell faces. In particular, the interface geometry evaluated in the previous step is used to discriminate, in the zones where two or more fluids coexist, which part of the volumetric flux corresponds to each of them. The different steps required to evaluate these volumetric fluxes at any face f, and consequently calculate Eq. 2.4, are presented below:

1. *Calculate the total volumetric flux value*. The value of the total advection volume is evaluated as
$$V_f = |\mathbf{u}_f \cdot \mathbf{n}_f| A_f \Delta t, \tag{2.9}$$
where Δt is the time step and $\mathbf{u}_f$, $\mathbf{n}_f$ and A_f correspond, respectively, to the velocity, the unit-outward normal and the area of face f. Notice that $\mathbf{u}_f$ is explicitly given if an analytic velocity field is used, or differently, $\mathbf{u}_f \cdot \mathbf{n}_f$ is the face velocity flux provided by the solution of the momentum equations in the case of a full problem.

2. *Construct the total volumetric flux polyhedron*. A polyhedron with volume V_f must be constructed over face f. In particular, a vertex-matched approach is used by setting the direction of the extrusion edges equal to the velocity vectors at the face vertices. This approach minimizes flux over/underlapping and ensures volume preservation. Further details are described in Subs. 2.4.3.

3. *Truncate the part of the volumetric flux polyhedron corresponding to each fluid*. If the

polyhedron only contains one fluid, truncation is not necessary, since $V_{k,f}$ is equal to V_f or 0, depending on the fluid being considered. Otherwise, geometric operations are required in order to truncate the part of the polyhedron corresponding to each fluid. In particular, the stencil of neighboring cells considered for the calculations are the cells that share at least one vertex with face f. For each of those, three actions are performed to evaluate the local part of $V_{k,f}$: (1) evaluate the intersection of the flux polyhedron and the cell; (2) if it is an interface cell, truncate the resulting polyhedron by the reconstructed interface; (3) add to $V_{k,f}$ the volume of the polyhedron resulting from the two previous actions. The basic geometric operation of the first two actions is the truncation of a polyhedron by a plane, details on its implementation can be found in [58].

4. *Calculate the new k'th fluid volume fraction*. Once the values of $V_{k,f}$ have been calculated for the different faces of a cell, the k'th volume fractions at time $n+1$, C_k^{n+1}, can be calculated by evaluating Eq. 2.4.

In the following subsections, steps 2 and 3 are explained in detail. Especially step 2 since it is the one related to the production of polyhedron over/underlapping, which, as discussed by Rider and Kothe [12], tends to produce volume fraction over/undershoots that decrease accuracy.

2.4.2 Minimizing over/underlapping

Situations of polyhedron overlapping are produced when two or more polyhedrons generated from different faces share a common part of their volume. Alternatively, underlapping is intrinsically generated from overlapping situations, i.e., polyhedrons do not embrace the correct amount of domain since some part of their volume is wasted in overlapping with other polyhedrons. Different polyhedron constructions may produce overlapping. For instance, an easy way to construct flux polyhedrons is by using face velocities. In this way, if velocities are not parallel, it may happen that polyhedrons created from contiguous edge faces overlap, as shown by polyhedrons A and B in Fig. 2.3. In order to overcome this problem, different methods exist in the literature that propose to construct special flux polyhedrons. For example, the 2-D edge-matched flux polygon advection (EMFPA-2D) method proposed and implemented on Cartesian meshes by López et al. [34], based on constructing edge-matched flux polygons at cell faces, which avoids over/underlapping between polygons. Extension of this method to 3-D meshes has been first accomplished for Cartesian grids by Hernández et al. [35], although making use of face-matched flux polyhedrons (FMFPA-3D) instead of EMFPA, and later has been generalized for unstructured grids and exactly using edge-matched flux polyhedrons (EMFPA-3D) by Ivey and Moin [51] — FMFPA-3D may present some over/underlapping between flux polyhedra constructed at cell faces with only one common vertex, as depicted in Fig. 2.3 by

polyhedrons *C*, *D* and *E*, while EMFPA-3D prevents this over/underlapping between polyhedrons.

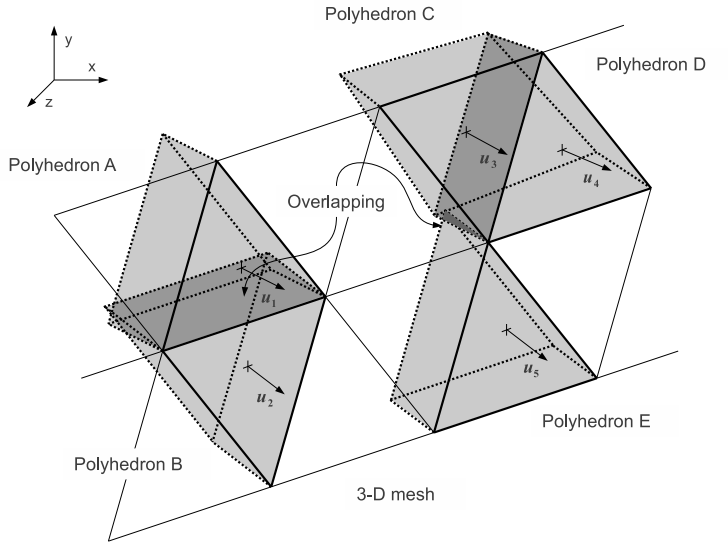


Figure 2.3: Overlapping between flux regions. Polyhedrons *A* and *B* overlap since they are constructed from face velocities. Polyhedrons *C*, *D* and *E* are constructed from interpolated face velocities, hence, overlapping may just occur between polyhedrons that share only one common vertex.

Another approach to minimize over/underlapping between flux polyhedrons is to construct them by tracing back the Lagrangian trajectories of the cell-vertex velocities. These vertex velocities may be explicitly given in the case of using an analytical velocity field or, in a general case, may be distance-interpolated from the velocities of the cells sharing the vertex. In this way, the over/underlappings between flux polyhedrons represented in Fig. 2.3 are avoided, since, as shown in Fig. 2.4, instead of creating a different face for each polyhedron that shares an edge or a vertex with another, a unique face is used for all them. Indeed, this idea is initially considered in the work by Liovic et al. [14], however, they finally decide to assign cell-face velocities to vertices in order to avoid the construction of flux polyhedrons having nonplanar faces. Even so, Mencinger and Žun [44] rely on this solution to implement their 2-D advection method suited for adaptive moving grids. Recently, this approach has been presented on 3-D Cartesian and unstructured meshes in a previous work [52] and, almost parallely in time with this paper, by Marić et al. [45], although they have just tested it on Cartesian grids. Hence, in order to reinforce this idea and analyze it on

3-D unstructured meshes, we also propose to construct flux polyhedrons by using cell-vertex velocities to minimize over/underlapping problems.

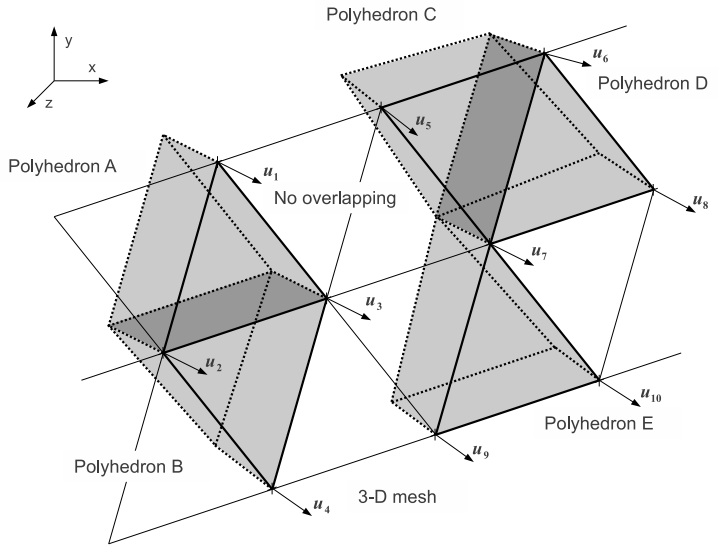


Figure 2.4: Overlapping between flux regions is minimized if polyhedrons are constructed by using cell-vertex velocities, since polyhedrons that share an edge or a vertex are created by using the same velocity components and, consequently, they share the same face instead of having different ones.

The construction of flux polyhedrons by using cell-vertex velocities considerably minimizes the number of over/underlappings, as shown in Fig. 2.4, but implies to deal with nonplanar surfaces. In fact, Liovic et al. [14] state that, even if orthogonal meshes are considered, using cell-vertex velocities results in having flux polyhedrons composed of five nonplanar faces. For example, if we analyze the situation depicted in Fig. 2.5, where polyhedron A is constructed by using vertices **a**, **b** and **c**, and their corresponding traced back Lagrangian trajectories to generate points $\mathbf{d} = \mathbf{a} - \Delta t \mathbf{u}_a$, $\mathbf{e} = \mathbf{b} - \Delta t \mathbf{u}_b$ and $\mathbf{f} = \mathbf{c} - \Delta t \mathbf{u}_c$, it can be observed that if a plane is created by points **a**, **b** and **d**, point **e** may not live on it. This is a major complication that must be taken into account when constructing flux polyhedrons in Sec. 2.4.3.

2.4.3 Construction of flux polyhedrons

This paper proposes to construct flux polyhedrons by using cell-vertex velocities in order to minimize overlapping, as shown in Fig. 2.4, but as explained in Fig. 2.5, this

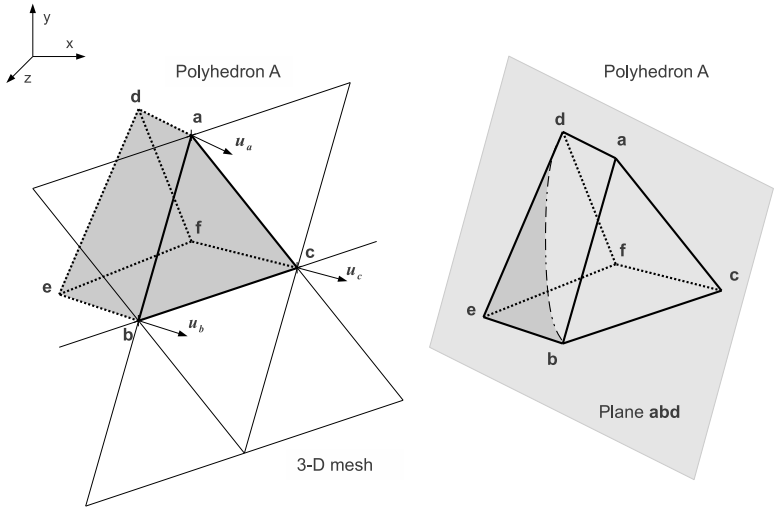


Figure 2.5: The construction of flux polyhedrons by using cell-vertex velocities (points **a**, **b** and **c**, and their corresponding traced back Lagrangian trajectories to generate points $\mathbf{d} = \mathbf{a} - \Delta t \mathbf{u}_a$, $\mathbf{e} = \mathbf{b} - \Delta t \mathbf{u}_b$ and $\mathbf{f} = \mathbf{c} - \Delta t \mathbf{u}_c$) produces faces that are nonplanar surfaces. If a plane is created by using points **a**, **b** and **d**, it can be seen that point **e** may not live on it.

solution creates polyhedrons with nonplanar surfaces. The straightforward manner to handle this complication is to approximate them by a set of tetrahedrons. In this way, it is ensured that all the polyhedrons comprised in the calculation are composed of convex tetrahedrons and, consequently, it is avoided the necessity to implement complex geometrical tools able to deal with nonconvex polyhedrons. This procedure is depicted in Fig. 2.6a. First, the centroid, **pc**, of the polyhedron defined by points **a**, **b**, **c**, **d**, **e** and **f** is calculated. Second, for each nonplanar face its centroid is calculated from its known vertices. For example, points **a**, **b**, **d** and **e** represent a nonplanar face, therefore, its centroid **fc** is calculated. Third, each nonplanar face is approximated to four tetrahedrons defined by the centroids of the polyhedron, **pc**, and face, **fc**, and two consecutive vertices of the face. For instance, in the face considered in Fig. 2.6a, four different tetrahedrons (gray scale) are used to define the volume comprised between the centroid of the polyhedron and points **a**, **b**, **d** and **e**. It is important to notice that this procedure is in accordance with the idea of nonoverlapping polyhedrons, since the centroid of two nonplanar faces belonging to two different polyhedrons that share an edge are placed in the same position in space, thus, the volumes contained by these two faces do not overlap.

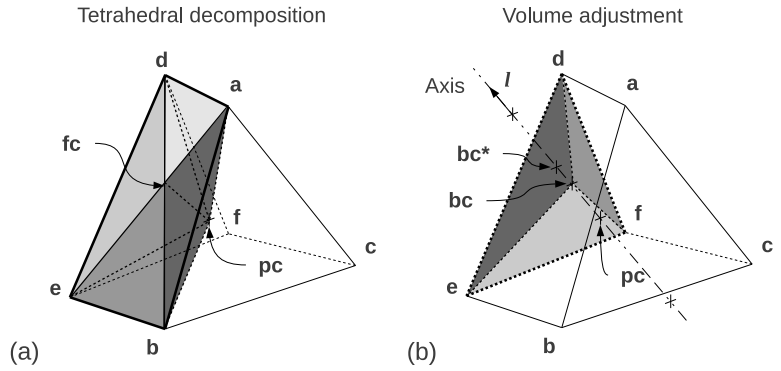


Figure 2.6: (a) Nonplanar faces are approximated by four different tetrahedrons (gray scale), implying the use of face vertices (**a**, **b**, **d**, **e**), face centroid (**fc**) and polyhedron centroid (**pc**). (b) The volume of the flux polyhedron is adjusted by placing the centroid of the back face in a specific position which makes it be equal to V_f.

In addition to the problem of having nonplanar faces, another complication is that the volume of the resulting polyhedron may not equal $V_f = |\mathbf{u}_f \cdot \mathbf{n}_f| A_f \Delta t$, thus, the conservation of volume required in Eq. 2.4 is not ensured —this condition is straightforwardly fulfilled if flux polyhedrons are constructed by using cell-face velocities, but it is not directly accomplished when using cell-vertex velocities. In the scientific literature, there are different approaches to solve this issue: (1) use a scalar coefficient to proportionally correct the geometric flux volumes [14, 34]; (2) calculate analytically [35] or iteratively [45] the position of the polyhedron's back face such that the volume of the resulting polyhedron equals V_f; (3) parametrically modify the lenght of the polyhedron's traced back Lagrangian trajectories to produce a polyhedron with volume V_f [44]. We choose the analytic option of the second approach for its good relation between performance and complexity, although we propose some differences. In particular, both Hernández et al. [35] and Marić et al. [45] adjust the polyhedron's back face so that its volume is equal to V_f by moving the end points as a whole (points **d**, **e** and **f** in Fig. 2.6). Differently, we propose to analytically calculate the specific position of the centroid of the polyhedron's back face such that the resulting polyhedron fulfills the volume condition; i.e., regarding Fig. 2.6, we fix points **d**, **e** and **f**, but we change the position of the centroid of the polyhedron's back face. In this way, the proposed polyhedron preserves volume while the traced back Lagrangian trajectories are not modified.

The volume adjustment method is better explained if considering Fig. 2.6b. In this case, the polyhedron's back face is defined by points $\mathbf{d} = \mathbf{a} - \Delta t \mathbf{u}_a$, $\mathbf{e} = \mathbf{b} - \Delta t \mathbf{u}_b$ and

$\mathbf{f} = \mathbf{c} - \Delta t \mathbf{u}_c$. First, the centroid of the polyhedron's back face placed in the standard position, **bc***, is calculated by using the position of points **d**, **e** and **f** as

$$\mathbf{bc^*} = \frac{1}{3}(\mathbf{d} + \mathbf{e} + \mathbf{f}). \tag{2.10}$$

Second, this point **bc*** together with the polyhedron's centroid, **pc**, are used to determine the unit vector, **l**, of the line where we choose to place the final centroid of the polyhedron's back face, **bc**. In detail, this unit vector is defined as

$$\mathbf{l} = \frac{\mathbf{bc^*} - \mathbf{pc}}{|\mathbf{bc^*} - \mathbf{pc}|}. \tag{2.11}$$

In this way, the generic position of the centroid of the polyhedron's back face can be expressed as

$$\mathbf{bc} = \mathbf{pc} + \lambda \mathbf{l}, \tag{2.12}$$

where λ is a scalar value that needs to be calculated.

Third, the front, side and back faces of the polyhedron are decomposed in tetrahedrons, as previously explained, and the volume of each one is calculated by using the method presented by Tuzikov et al. [59]. For instance, if a tetrahedron is defined by points **a**, **b**, **c** and the coordinates origin ($T_1 = T(\mathbf{a}, \mathbf{b}, \mathbf{c})$), its volume is

$$V_{T_1} = \frac{1}{6}[\mathbf{a} \cdot (\mathbf{b} \times \mathbf{c})]. \tag{2.13}$$

Similarly, in a general case like the polyhedron defined in Fig. 2.6a by points **a**, **fc**, **b** and **pc**, the volume may be evaluated as

$$V_{T_2} = \frac{1}{6}[\mathbf{a} \cdot (\mathbf{fc} \times \mathbf{b}) + \mathbf{a} \cdot (\mathbf{b} \times \mathbf{pc}) + \mathbf{a} \cdot (\mathbf{pc} \times \mathbf{fc}) + \mathbf{b} \cdot (\mathbf{fc} \times \mathbf{pc})]. \tag{2.14}$$

Deepening, this expression may be largely simplified if it is noticed that the terms comprised of points defining interior faces of the polyhedron (e.g., $\mathbf{a} \cdot (\mathbf{b} \times \mathbf{pc})$, $\mathbf{a} \cdot (\mathbf{pc} \times \mathbf{fc})$ and $\mathbf{b} \cdot (\mathbf{fc} \times \mathbf{pc})$) will cancel out between them when it is summed the volume of all the different tetrahedrons that comprise it. Thus, in the case that V_{T_2} is to be evaluated as part of the total volume of the polyhedron, Eq. 2.14 may be simplified to

$$V_{T_2} = \frac{1}{6}[\mathbf{a} \cdot (\mathbf{fc} \times \mathbf{b})]. \tag{2.15}$$

Hence, the volume of the different tetrahedrons (four for each face) that compose the side faces are calculated from the method of Tuzikov et al. [59] by using the coordinates of the points that define them, and the summation of all them is saved as S. Same scheme is used to evaluate the volume of the three tetrahedrons defining

the front face and the sum is stored as F. Also, although the position of **bc** is still unknown, the same method can be applied to express the volume defined by the back face, B, as

$$B = \frac{1}{6}[\mathbf{bc} \cdot (\mathbf{e} \times \mathbf{d}) + \mathbf{bc} \cdot (\mathbf{d} \times \mathbf{f}) + \mathbf{bc} \cdot (\mathbf{f} \times \mathbf{e})]. \tag{2.16}$$

In this way, the volume of the entire polyhedron, which must equal V_f, may be defined as

$$V_f = F + S + B. \tag{2.17}$$

Then, if the volume of the polyhedron's back face is transformed to group the terms multiplying **bc**, by writing

$$B = \frac{1}{6}[\mathbf{bc} \cdot (\mathbf{e} \times \mathbf{d} + \mathbf{d} \times \mathbf{f} + \mathbf{f} \times \mathbf{e})] = \frac{1}{6}[\mathbf{bc} \cdot \mathbf{B}'], \tag{2.18}$$

the volume of the polyhedron may be rewritten as

$$V_f = F + S + \frac{1}{6}[\mathbf{bc} \cdot \mathbf{B}']. \tag{2.19}$$

Fifth, if the generic position of **bc**, Eq. 2.12, is introduced into Eq. 2.19, the value of λ can be calculated analytically as

$$\lambda = \frac{6(V_f - F - S) - \mathbf{pc} \cdot \mathbf{B}'}{\mathbf{l} \cdot \mathbf{B}'}. \tag{2.20}$$

Finally, once the value of λ is known, Eq. 2.12 determines the position of the centroid of the polyhedron's back face, **bc**, such that the polyhedron's volume equals V_f.

2.4.4 Truncation of flux polyhedrons

Once the flux polyhedron is constructed, it needs to be truncated by the reconstructed interface in order to determine the part of it corresponding to fluid k, $V_{k,f}$. Since this operation is complex and time consuming, especially for 3-D grids, it is important to simplify it by identifying the three situations depicted in Fig. 2.7. For instance, if the advection equation, Eq. 2.4, is solved for fluid 1, three different polyhedrons may be defined: (1) polyhedron A is situated completely outside fluid 1, hence, its flux volume is zero; (2) polyhedron C is totally immersed in fluid 1, consequently, its flux volume is directly V_f; (3) polyhedron B is cut by an interface plane, as a result, this one needs to be constructed and truncated in order to evaluate the portion of its volume that belongs to fluid 1. In the third case, the local parts corresponding to fluid 1 of the neighboring cells that share at least one vertex with the face need to be

calculated. For each of these cells, three actions are performed to evaluate the local part of $V_{1,f}$: (1) the flux polyhedron is truncated by the faces of the cell; (2) if it is an interface cell, the resulting polyhedron is truncated by the reconstructed interface; (3) the volume of the polyhedron resulting from the two previous actions is added to $V_{1,f}$. Notice that the main operation in the first two actions is the truncation of a polyhedron by a plane, what is a very complex geometrical operation in the 3-D case, but, since we propose to decompose flux polyhedrons by sets of convex tetrahedrons, the geometrical solutions proposed by López and Hernández [58] may be directly utilized.

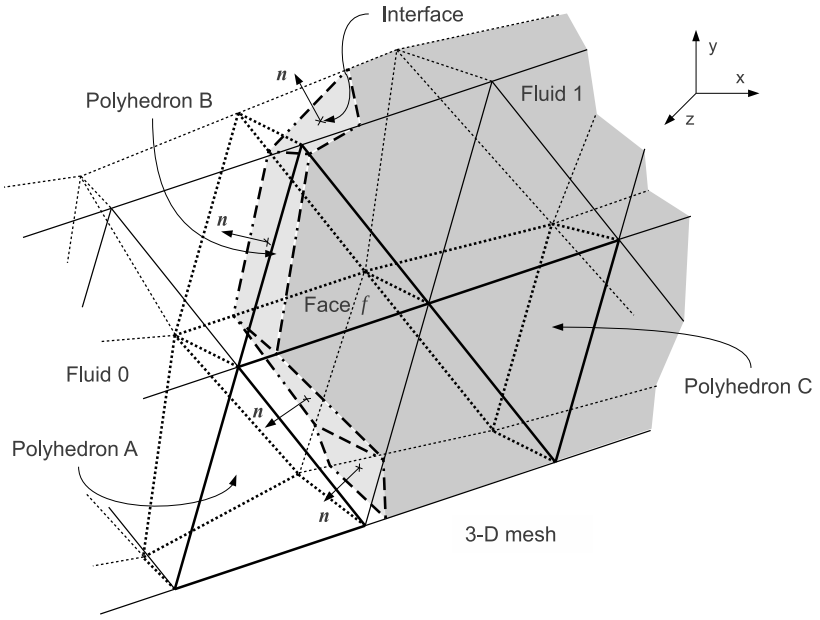


Figure 2.7: Schematic drawing representing a 3-D unstructured mesh, the flux polyhedrons constructed and the interface between fluids 0 and 1. Three different situations are sketched: (1) polyhedron *A* is situated outside fluid 1, (2) polyhedron *B* is cut by the interface and (3) polyhedron *C* is totally immersed in fluid 1.

2.4.5 Correction of undershoots, overshoots and wisps

On occassions, some errors may be introduced to the solution of the advection equation, Eq. 2.4, generating, in the cells close to the interface, undershoots ($C_k < 0$),

overshoots ($C_k > 1$) or wisps (fluid in void regions or vice versa) that later make more difficult to reconstruct correctly the interface. These errors are caused basically by discretization errors and velocity fields with nonzero divergence. In order to improve the boundedness of the interface, when any of the previous errors occurs is useful to use a local redistribution algorithm similar to the one proposed by Harvie and Fletcher [60], but modified for 3-D unstructured meshes as described in Alg. 1.

Algorithm 1 Redistribution of non-real k volume fractions

for $0 \leq c < nonRealCells$ **do**
  if $C_{k,c} < 0$ **then**
    while $C_{k,c} < 0$ **do**
      Find surrounding cell with highest $C_k \Rightarrow C_{k,max}$
      $V_{aux} = C_{k,max} \cdot V_{max} + C_{k,c} \cdot V_c$
      $C_{k,max} = max(V_{aux}/V_{max}, 0)$
      $C_{k,c} = min(V_{aux}/V_c, 0)$
    end while
  else $\{C_{k,c} > 1\}$
    while $C_{k,c} > 1$ **do**
      Find surrounding cell with lowest $C_k \Rightarrow C_{k,min}$
      $V_{aux1} = C_{k,min} \cdot V_{min} + (C_{k,c} - 1) \cdot V_c$
      $V_{aux2} = (C_{k,min} - 1) \cdot V_{min} + C_{k,c} \cdot V_c$
      $C_{k,max} = min(V_{aux1}/V_{min}, 1)$
      $C_{k,c} = max(V_{aux2}/V_c, 1)$
    end while
  end if
end for

First, cells that present undershoots are corrected by transferring fluid k from the surrounding cells with highest k volume fractions. For instance, if a cell presents a k volume fraction value under 0, part of the volume of fluid k of the surrounding cell with highest C_k is used to fill the cell under consideration. Hence, at the end the result is that the cell presents a volume fraction value of 0 while the surrounding cell still contains a volume of fluid k between real values.

Second, cells with overshoots are adjusted by transferring the extra volume to the surrounding cells with lowest C_k in a similar way as the undershoots. For example, if a cell presents a C_k over 1, the extra volume is transferred to the surrounding cell with lowest C_k, carefully checking that the surrounding cell is able to gain the volume without becoming an overshoot. In this way, the overshoot cell is fixed to a volume fraction value of 1 without creating new overshoots or undershoots.

Finally, cells containing fluid/void wisps are corrected to values 0 or 1 depending

on their surrounding cells: the C_k of a fluid cell surrounded just by cells is changed to 0, alternatively, void cells surrounded by fluid cells are changed to 1.

2.5 Numerical tests

The accuracy of the reconstruction and advection algorithms presented in this paper is studied in detail by implementing them in the *TermoFluids* parallel unstructured CFD platform [61] and performing numerical tests. These are performed on 3-D Cartesian and unstructured meshes containing different number of cells. In particular, the Cartesian grids used are named according to the number of cells in which each direction (x, y and z) is discretized, while the unstructured meshes are named by utilizing the name of the Cartesian grid that contains a similar number of total cells. The results on Cartesian grids are compared to the ones found in the scientific literature. In contrast, in the case of 3-D unstructured meshes, this paper is one of the first works in where numerical results of advection tests are presented.

Another important issue is the quality of the meshes used. In this work, the quality of a mesh cell is defined as the ratio between the radius of an inscribed sphere to a circumscribed one (aspect ratio). The values are scaled, so that an aspect ratio of 1 is perfectly regular, and an aspect ratio of 0 indicates that the element has zero volume. In this way, the Cartesian grids used in this paper present aspect ratios of 1, while the unstructured ones present average aspect ratios between 0.7 (coarse mesh) and 0.9 (dense mesh).

2.5.1 Reconstruction tests

The sphere and hollowed sphere tests are used to examine the accuracy of the reconstruction methods presented in Sec. 2.3. These test problems are stationary, i.e., no advection is performed and hence there is no error due to discretization in time. In particular, the sphere test reconstructs a sphere of radius 0.325 and the hollowed sphere test a sphere of radius 0.4 (convex surface) with a spherical core of radius 0.2 (concave surface) hollowed out of it, on a cube of length 1.

The interface reconstruction error is measured as the difference between the exact interface and the reconstructed one. An L_1 error norm is used, which as found in the work by Liovic et al. [14], in the continuous limit is the integral

$$E_{L_1} = \int |\chi(\mathbf{x}) - \tilde{\chi}(\mathbf{x})| dV, \tag{2.21}$$

where $\chi(\mathbf{x})$ is the exact interface topology and $\tilde{\chi}(\mathbf{x})$ is its approximation obtained using an interface reconstruction method.

The error norm L_1 results are shown in Tabs. 2.1 and 2.2. In particular, the Youngs algorithm shows first-order accuracy and better results on coarse grids (10^3 - 20^3 cells), while LVIRA is second-order accurate and performs better when the grid is refined (40^3 - 80^3 cells), but requires more computational time since it performs a 2-D minimization. Furthermore, the results on Cartesian grids are similar to the ones obtained on equivalent unstructured meshes, except for the very coarse meshes where Cartesian grids present better results.

It is important to note that the calculation of the Youngs interface reconstruction is faster on unstructured meshes than on Cartesian ones due to the less number of faces per cell — tetrahedrons are composed of four faces while hexahedrons of six. On the contrary, the LVIRA interface reconstruction method performs faster on Cartesian grids than on unstructured meshes since the number of surrounding cells per cell is minor in the first case — a hexahedral cell is surrounded by twenty-six cells that share at least one vertex with it, while a tetrahedral cell is surrounded, depending on the mesh configuration, by sixty to seventy cells.

The reconstruction planes of the sphere and hollowed sphere interface reconstruction tests are shown, on successively refined meshes, from Fig. 2.8 to Fig. 2.15. The reconstructed spheres of the hollowed sphere test have been cutted by its half to show the inner interface.

Sphere	**Cartesian**				**Unstructured**			
mesh	Youngs		LVIRA		Youngs		LVIRA	
10^3	$1.87E{-3}$		$2.79E{-3}$		$4.86E{-3}$		$1.06E{-2}$	
20^3	$5.85E{-4}$	*1.68*	$6.68E{-4}$	*2.06*	$8.66E{-4}$	*2.49*	$1.30E{-3}$	*3.03*
40^3	$2.51E{-4}$	*1.22*	$1.64E{-4}$	*2.02*	$2.79E{-4}$	*1.64*	$3.47E{-4}$	*1.91*
80^3	$1.21E{-4}$	*1.06*	$4.55E{-5}$	*1.85*	$9.88E{-5}$	*1.50*	$8.07E{-5}$	*2.10*

Table 2.1: E_{L_1} errors for the sphere interface reconstruction tests. The computed orders of accuracy between meshes are in italics on the right side.

Hollowed	**Cartesian**				**Unstructured**			
mesh	Youngs		LVIRA		Youngs		LVIRA	
10^3	$3.60E{-3}$		$5.35E{-3}$		$1.62E{-2}$		$2.41E{-2}$	
20^3	$1.13E{-3}$	*1.67*	$1.20E{-3}$	*2.16*	$1.64E{-3}$	*3.30*	$2.46E{-3}$	*3.29*
40^3	$4.80E{-4}$	*1.23*	$3.06E{-4}$	*1.97*	$5.34E{-4}$	*1.62*	$6.43E{-4}$	*1.94*
80^3	$2.29E{-4}$	*1.07*	$8.52E{-5}$	*1.85*	$1.89E{-4}$	*1.50*	$1.49E{-4}$	*2.11*

Table 2.2: E_{L_1} errors for the hollowed sphere interface reconstruction tests. The computed orders of accuracy between meshes are in italics on the right side.

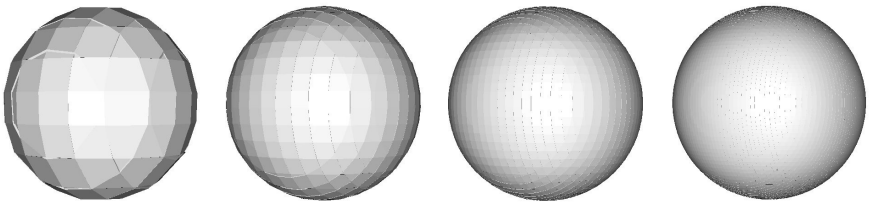


Figure 2.8: Reconstructions of the sphere using the Youngs algorithm on successively refined Cartesian grids.

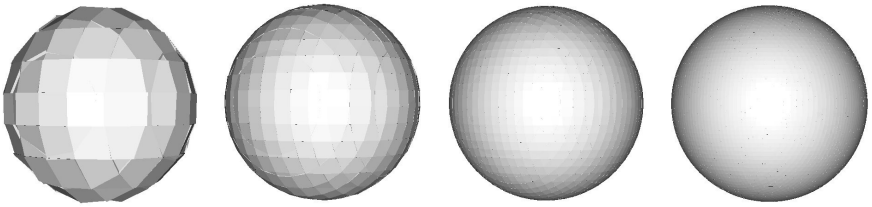


Figure 2.9: Reconstructions of the sphere using the LVIRA algorithm on successively refined Cartesian grids.

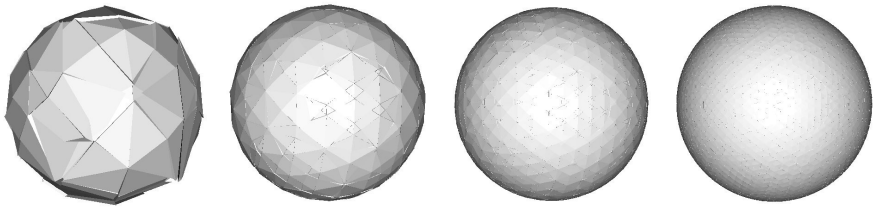


Figure 2.10: Reconstructions of the sphere using the Youngs algorithm on successively refined unstructured meshes.

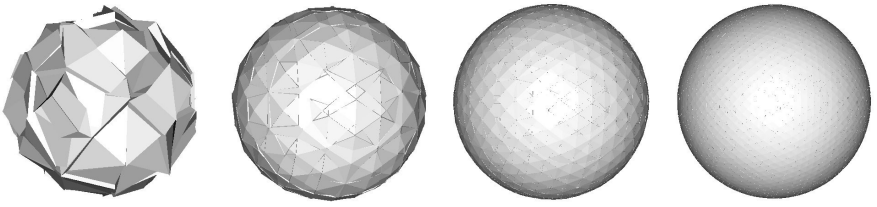


Figure 2.11: Reconstructions of the sphere using the LVIRA algorithm on successively refined unstructured meshes.

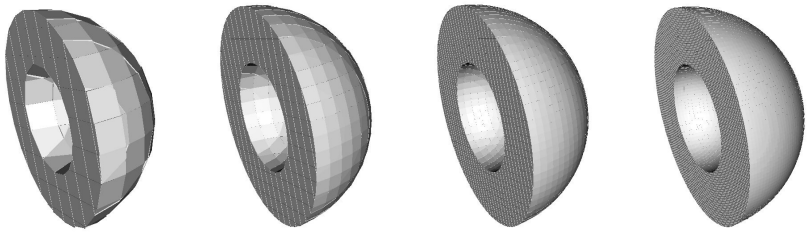


Figure 2.12: Reconstructions of the hollowed sphere using the Youngs algorithm on successively refined Cartesian grids.

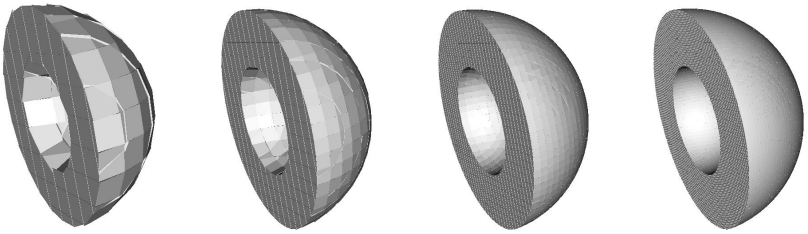


Figure 2.13: Reconstructions of the hollowed sphere using the LVIRA algorithm on successively refined Cartesian grids.

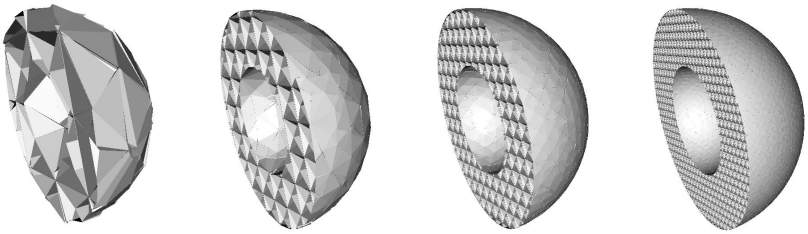


Figure 2.14: Reconstructions of the hollowed sphere using the Youngs algorithm on successively refined unstructured meshes.

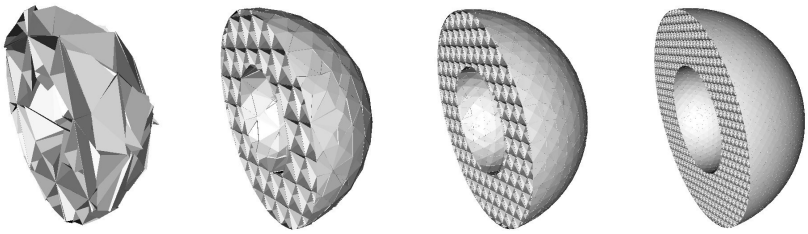


Figure 2.15: Reconstructions of the hollowed sphere using the LVIRA algorithm on successively refined unstructured meshes).

2.5.2 Advection tests

The rotation, shear and deformation tests are used to analyze the accuracy of the advection algorithm presented in Sec. 2.4. The tests are solved on 3-D Cartesian and unstructured meshes using the Youngs and LVIRA reconstruction methods.

The advection error is measured as the difference between the initial and final (after advection) volume fraction functions. Similar to Liovic et al. [14], an L_1 error norm is used, which in the discrete form is the summation

$$E_{L_1} = \sum_{c \in \Omega} V_c |\tilde{C}_{k,c} - C_{k,c}|, \tag{2.22}$$

where $C_{k,c}$ and $\tilde{C}_{k,c}$ are the volume fraction functions for fluid k before and after advection, respectively, and V_c refers to the volume of cell c. In addition, the relative conservation of volume, E_{m_1}, between the initial and final total volume occupied by fluid k is calculated for each case, which as proposed by Aulisa et al. [62], may be expressed as

$$E_{m_1} = \frac{|\sum_{c \in \Omega} \tilde{C}_{k,c} - \sum_{c \in \Omega} C_{k,c}|}{\sum_{c \in \Omega} C_{k,c}}. \tag{2.23}$$

Rotation flow

The rotation flow test is a simple problem that induces no change in the interface topology and is largely used to test VOF implementations, e.g., Rider and Kothe [12], Liovic et al. [14] and Aulisa et al. [62]. Starts with a sphere of radius 0.15 centered at $(0.5, 0.75, 0.5)$ in a cube of length 1 and, then, is advected for a complete turn in a rotation flow field:

$$\begin{aligned} u &= y - y_0, \\ v &= -(x - x_0), \\ w &= 0.0, \end{aligned} \tag{2.24}$$

where x_0, y_0 and z_0 are the coordinates of the center of the cube. In particular, a CFL value of 1.0 for the calculation of the time step is used (maximum velocity 1.0) in this test.

The error norm L_1 results for the 3-D rotation advection tests are shown in Tab. 2.3. In general, the results obtained on Cartesian meshes are similar to the ones presented by Hernández et al. [35], while results for successively refined unstructured meshes are of same order of magnitude as the Cartesian ones. In addition, notice that the results on Cartesian grids present first-order accuracy when using the Youngs reconstruction algorithm and second-order if using LVIRA. However, the accuracy

on unstructured meshes is not as easy to analyze, since the proportionality in cell size is not maintained when meshes are densified. Even so, the results on unstructured meshes tend to be first- and second-order for Youngs and LVIRA algorithms, respectively.

Shear flow

The shear flow test is a more complex problem which combines a single vortex in the xy-plane with a laminar pipe flow in the z-direction. Many authors have used it to test their VOF algorithms, e.g., Rider and Kothe [12], Du et al. [10], Liovic et al. [14] and Aulisa et al. [62]. The test starts with a sphere of radius 0.15 placed at position $(0.5, 0.75, 0.25)$ in a $1.0\times1.0\times2.0$ rectangular prism domain and, then, is advected by the velocity flow field:

$$\begin{aligned} u &= sin(2\pi y)sin^2(\pi x)cos\left(\frac{\pi t}{T}\right), \\ v &= -sin(2\pi x)sin^2(\pi y)cos\left(\frac{\pi t}{T}\right), \\ w &= U_{max}\left(1-\frac{r}{R}\right)^2 cos\left(\frac{\pi t}{T}\right), \end{aligned} \tag{2.25}$$

where $r = \sqrt{(x-x_0)^2+(y-y_0)^2}$, $x_0 = 0.5$, $y_0 = 0.5$, $R = 0.5$ and $U_{max} = 1.0$. In this test, a CFL value of 0.5 is used for the calculation of the time step (maximum velocity 1.0) and the test period is set to $T = 3$.

The norm L_1 errors for the 3-D shear advection tests are written in Tab. 2.4. The table shows that the results obtained on Cartesian grids are of same order of magnitude as the ones presented by Liovic et al. [14] (from 10^{-3} on coarse grids to 10^{-4} on dense ones), while results on unstructured meshes present similar errors than the Cartesian ones. Furthermore, as previously pointed out in the rotation tests, the Youngs reconstruction method is first-order while the LVIRA one tends to be second-order, both on Cartesian as on unstructured meshes.

The interface reconstruction planes of the shear tests for the different meshes and reconstruction methods are plotted at half-period, $T/2$, from Fig. 2.16 to Fig. 2.19. Notice that, when meshes are refined, the reconstructed interfaces become smoother, both on Cartesian grids as on unstructured ones. Moreover, the conservation of volume remains delimited between 10^{-9} and 10^{-11}, what proves that Eq. 2.4 is being accurately resolved by using the volume-adjusted flux polyhedrons proposed in Sec. 2.4.3.

Deformation flow

The final problem used to analyze the accuracy of the advection algorithm is the deformation flow test, which was first proposed by LeVeque [63]. The test consists in a sphere of radius 0.15 centered at $(0.35, 0.35, 0.35)$ within a unit cube domain that is deformed in a solenoidal velocity flow field:

$$\begin{aligned} u &= 2sin^2(\pi x)sin(2\pi y)sin(2\pi z)cos\left(\frac{\pi t}{T}\right), \\ v &= -sin(2\pi x)sin^2(\pi y)sin(2\pi z)cos\left(\frac{\pi t}{T}\right), \\ w &= -sin(2\pi x)sin(2\pi y)sin^2(\pi z)cos\left(\frac{\pi t}{T}\right). \end{aligned} \tag{2.26}$$

In this problem, a CFL value of 0.5 for the calculation of the time step is used (maximum velocity 2.0) and the test period is set to $T = 3$.

The L_1 errors for the deformation tests are shown in Tab. 2.5. The first observation is that the results on Cartesian meshes are similar to the ones found in the scientific literature, e.g., Du et al. [10], Liovic et al. [14], Hernández et al. [35]. In particular, all these literature results range from 10^{-3} (32^3 grids) to 10^{-4} (128^3 grids). On the contrary, errors on unstructured meshes tend to be larger than Cartesian ones on coarse meshes, but converge to Cartesian results when meshes are densified. The main reason for this error difference between both mesh types, is the better capacity of Cartesian grids over unstructured ones to reconstruct interfaces if given a same amount of cells. Furthermore, the results show that the LVIRA reconstruction method obtains better results than the Youngs one, both on Cartesian as on unstructured meshes.

The interface reconstruction planes of the deformation tests for the different meshes and reconstruction methods are depicted at maximum deformation, $T/2$, from Fig. 2.20 to Fig. 2.23. The figures clearly show that the deformation applied to the spheres is so large that the meshes used are not fine enough to correctly reconstruct the interface. Even so, when using the LVIRA reconstruction method, the interface at the extremely deformed zones is reconstructed in a smoother manner. In this way, the advection step cuts the flux polyhedrons by more accurate planes, resulting in better overall outcomes. Furthermore, notice that when using unstructured coarse meshes (32^3), the interface reconstruction planes at $T/2$ are not accurate compared to the Cartesian ones, see Fig. 2.22 and 2.23, but as meshes are densified (64^3 and 128^3) results rapidly converge to the Cartesian ones, as analyzed in Sec. 2.5.1. Finally, it is important to mention that, even when using coarse grids (32^3), the conservation of volume is good (around 10^{-9}), what means that the break up of the interface is due to its inaccurate reconstruction, not by loss or gain of volume.

Rotation	**Cartesian**				**Unstructured**			
mesh	Youngs		LVIRA		Youngs		LVIRA	
32^3	$4.23E{-4}$		$5.47E{-4}$		$4.25E{-4}$		$6.30E{-4}$	
64^3	$1.62E{-4}$	*1.39*	$1.29E{-4}$	*2.08*	$1.79E{-4}$	*1.25*	$2.31E{-4}$	*1.45*
128^3	$7.93E{-5}$	*1.03*	$3.46E{-5}$	*1.90*	$7.50E{-5}$	*1.25*	$6.79E{-5}$	*1.76*

Table 2.3: E_{L_1} errors for the rotation advection tests. The computed orders of accuracy between meshes are in italics on the right side.

Shear	**Cartesian**				**Unstructured**			
mesh	Youngs		LVIRA		Youngs		LVIRA	
32x32x64	$4.06E{-3}$		$4.08E{-3}$		$6.15E{-3}$		$5.97E{-3}$	
64x64x128	$1.29E{-3}$	*1.66*	$1.46E{-3}$	*1.48*	$2.03E{-3}$	*1.60*	$1.64E{-3}$	*1.87*
128x128x256	$5.45E{-4}$	*1.24*	$3.53E{-4}$	*2.05*	$8.53E{-4}$	*1.25*	$5.37E{-4}$	*1.61*

Table 2.4: E_{L_1} errors for the shear advection tests. The computed orders of accuracy between meshes are in italics on the right side.

Deformation	**Cartesian**				**Unstructured**			
mesh	Youngs		LVIRA		Youngs		LVIRA	
32^3	$7.47E{-3}$		$6.92E{-3}$		$1.02E{-2}$		$1.02E{-2}$	
64^3	$2.77E{-3}$	*1.43*	$2.43E{-3}$	*1.51*	$4.45E{-3}$	*1.20*	$3.54E{-3}$	*1.53*
128^3	$8.14E{-4}$	*1.77*	$6.37E{-4}$	*1.93*	$9.43E{-4}$	*2.24*	$7.20E{-4}$	*2.30*

Table 2.5: E_{L_1} errors for the deformation advection tests. The computed orders of accuracy between meshes are in italics on the right side.

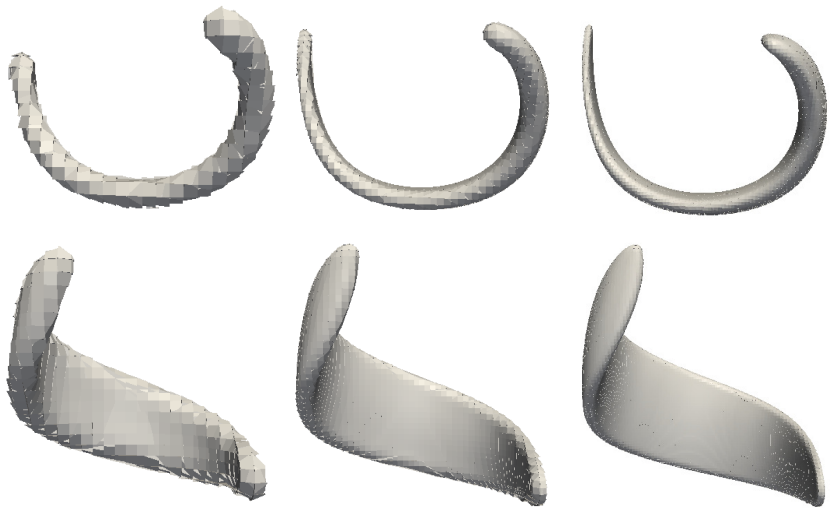


Figure 2.16: Interface reconstruction planes at half period for the 3-D shear flow test using the Youngs algorithm on successively refined Cartesian grids. The top row shows the xy-plane view, while the bottom one the xz-plane view.

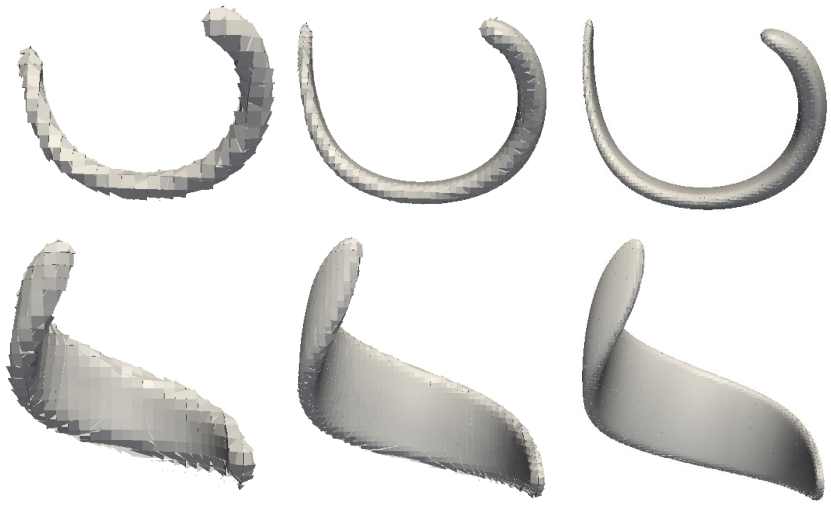


Figure 2.17: Interface reconstruction planes at half period for the 3-D shear flow test using the LVIRA algorithm on successively refined Cartesian grids. The top row shows the xy-plane view, while the bottom one the xz-plane view.

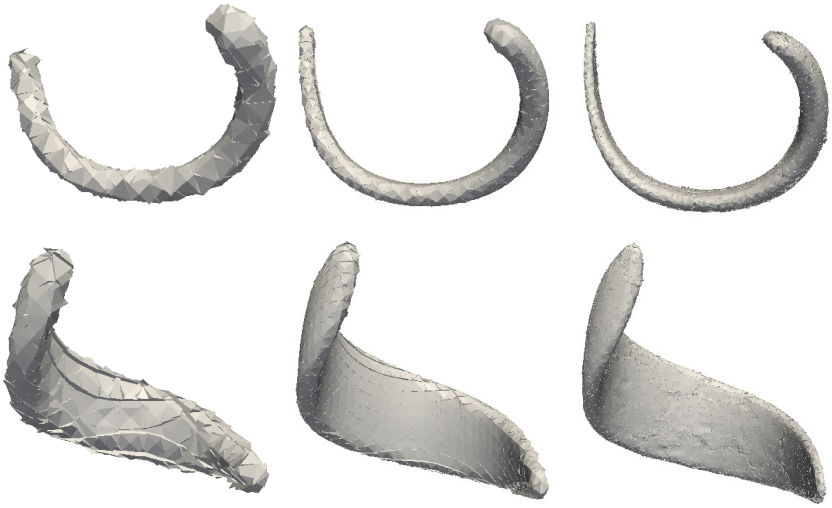


Figure 2.18: Interface reconstruction planes at half period for the 3-D shear flow test using the Youngs algorithm on successively refined unstructured meshes. The top row shows the xy-plane view, while the bottom one the xz-plane view.

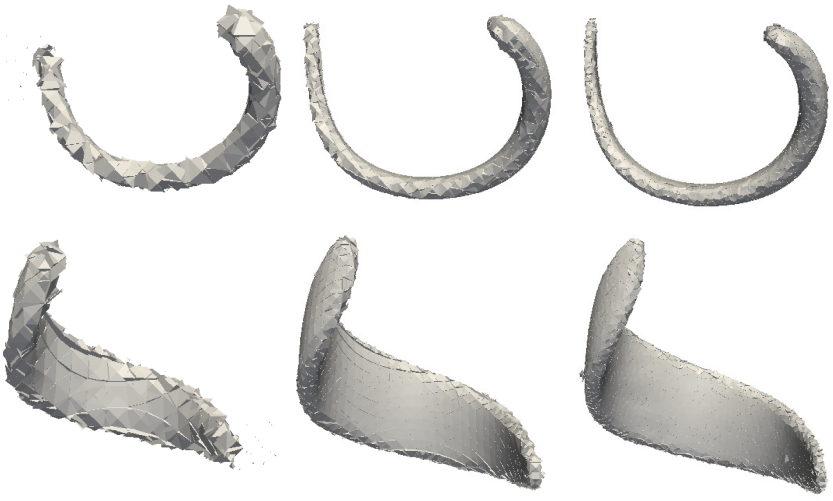


Figure 2.19: Interface reconstruction planes at half period for the 3-D shear flow test using the LVIRA algorithm on successively refined unstructured meshes. The top row shows the xy-plane view, while the bottom one the xz-plane view.

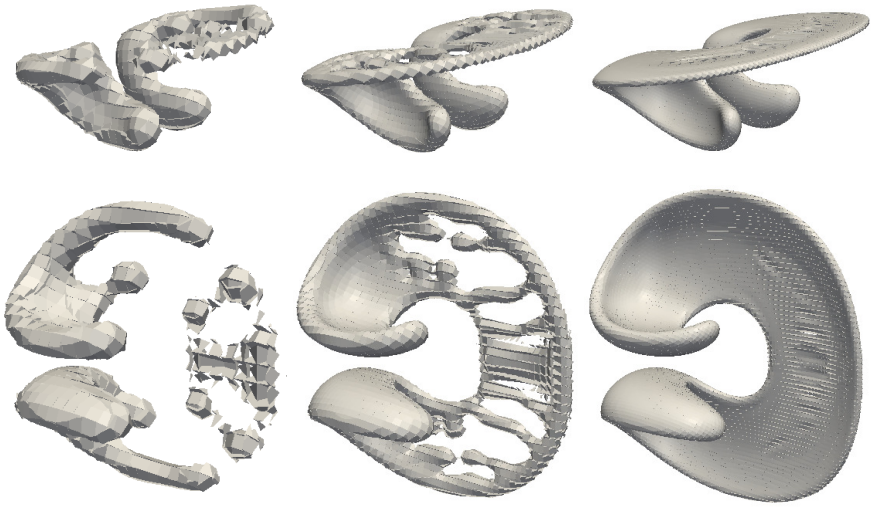


Figure 2.20: Interface reconstruction planes at half period for the 3-D deformation flow test using the Youngs algorithm on successively refined Cartesian grids. The images are two side views of the test at the instant of maximum deformation.

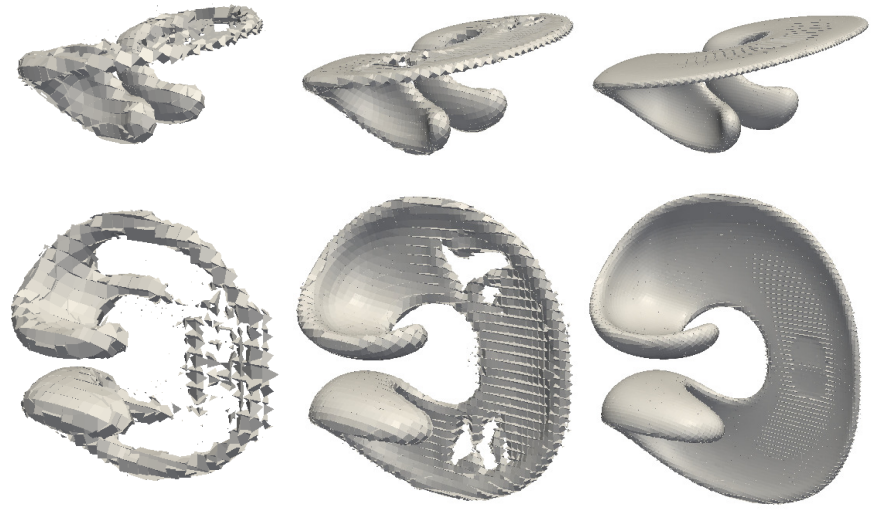


Figure 2.21: Interface reconstruction planes at half period for the 3-D deformation flow test using the LVIRA algorithm on successively refined Cartesian grids. The images are two side views of the test at the instant of maximum deformation.

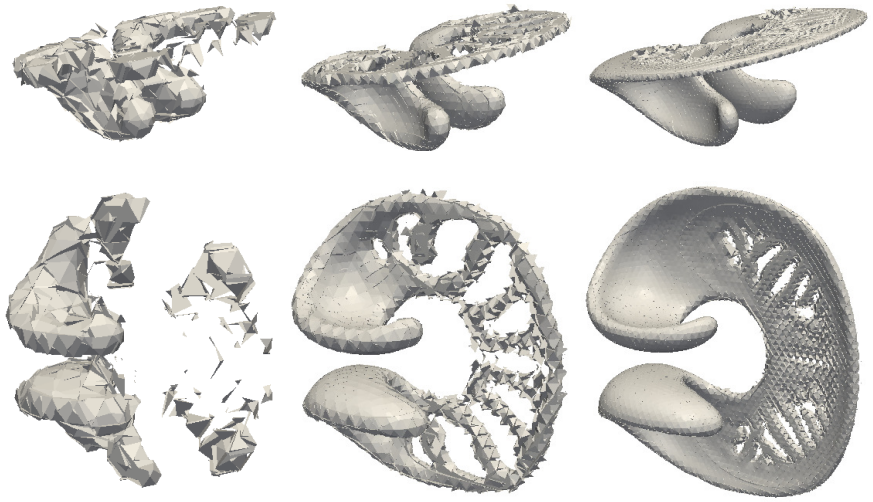


Figure 2.22: Interface reconstruction planes at half period for the 3-D deformation flow test using the Youngs algorithm on successively refined unstructured meshes. The images are two side views of the test at the instant of maximum deformation.

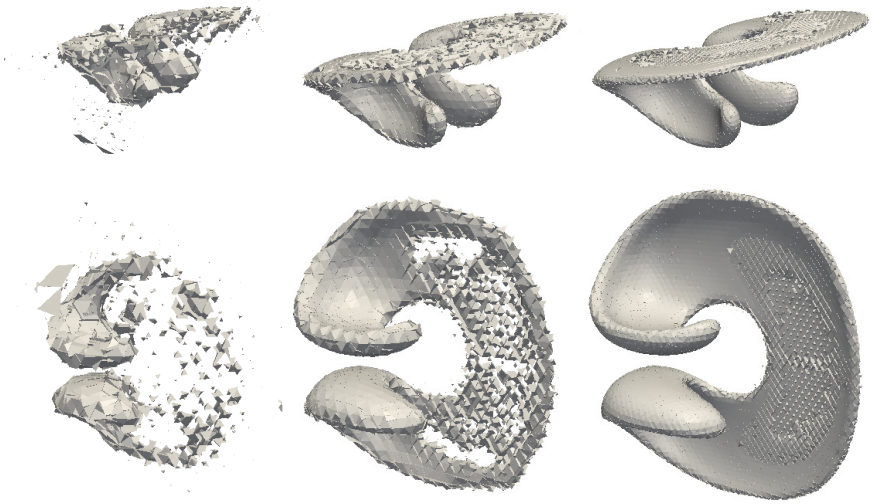


Figure 2.23: Interface reconstruction planes at half period for the 3-D deformation flow test using the LVIRA algorithm on successively refined unstructured meshes. The images are two side views of the test at the instant of maximum deformation.

2.6 Conclusions

A geometrical Volume-of-Fluid method based on a new approach for the multidimensional advection has been proposed for capturing interfaces on 3-D Cartesian and unstructured meshes. In particular, the first-order Parker and Youngs [53] and second-order LVIRA [38] interface reconstruction methods have been implemented on 3-D unstructured meshes, while the proposed advection step constructs flux polyhedrons by using the Lagrangian trajectories of the cell-vertex velocities. In detail, the advection method presents similarities with the one recently published by Marić et al. [45], since both works use cell-vertex velocities, but differs in that ours adjusts analitically the volume of the flux polyhedrons. Moreover, this work presents more complete advection results, since tests on unstructured meshes are performed. The use of cell-vertex velocities minimizes the situation of over/underlapping between flux polyhedrons, however, the volume of the polyhedrons needs to be adjusted in order to correctly solve the advection equation. For this purpose, a set of geometric algorithms have been explained in detail in Sec. 2.4. In addition, the possible numerical errors introduced to the solution when advecting volumes in time — e.g., undershoots, overshoots and wisps — are sorted out by using a local redistribution algorithm similar to the one proposed by Harvie and Fletcher [60], but extended to unstructured meshes.

The accuracy of the interface reconstruction algorithms has been studied by solving the sphere and hollowed sphere reconstruction tests, using Cartesian and unstructured meshes ranging from 10^3 to 80^3 cells (approximate for unstructured meshes). The results obtained are of same order of magnitude as the ones found in the literature, e.g., Liovic et al. [14] and Ahn and Shashkov [46]. In addition, the tests demonstrate that the Youngs algorithm is first-order accurate and exhibits better results on coarse grids (10^3 - 20^3 cells), while LVIRA is second-order accurate and performs better when the grid is refined (40^3 - 80^3 cells), but requires more computational time since it performs a 2-D minimization.

The rotation, shear and deformation flow tests have been used to analyze the accuracy of the advection algorithm developed in this work. The tests are solved on 3-D Cartesian grids and, for the first time in the scientific literature, on 3-D unstructured meshes using the Youngs and LVIRA reconstruction methods. First, the rotation and shear flow test results obtained on Cartesian grids are similar to the ones presented by Hernández et al. [35] and Liovic et al. [14], respectively, while results for successively refined unstructured meshes are of same order of magnitude as the Cartesian ones. Second, the deformation test results on Cartesian meshes follow the behavior and magnitude of the different errors presented in the literature, e.g., Du et al. [10], Liovic et al. [14] and Hernández et al. [35], while errors for unstructured meshes tend to be larger than Cartesian ones on coarse meshes, but converge to Cartesian results as meshes are densified. Hence, the fact that the results on Cartesian grids are similar to

the ones found in the literature and that the results on unstructured meshes are similar to the former ones, demonstrates that the proposed unsplit advection algorithm solves correctly the advection equation both on Cartesian as on unstructured meshes. Furthermore, independently of the type of mesh used, the three tests show that the Youngs reconstruction method is first-order while LVIRA tends to be second-order. Finally, the conservation of volume remains delimited between 10^{-9} and 10^{-11} for all tests, what proves that the advection equation is accurately solved if cell-vertex velocities are used to construct flux polyhedrons and their volumes are adjusted.

Acknowledgements

This work has been financially supported by the *Ministerio de Economía y Competitividad, Secretaría de Estado de Investigación, Desarrollo e Innovación*, Spain (ENE-2010-17801 and ENE-2011-28699), a FPU Grant by the *Ministerio de Educación, Cultura y Deporte*, Spain (AP-2008-03843) and by *Termo Fluids S.L.*

The computations presented in this work have been carried out on the *IBM MareNostrum-III* supercomputer at the *Barcelona Supercomputing Center* (BSC), Spain (FI-2012-3-0021 and FI-2013-1-0024). The authors thankfully acknowledge this Institution.

References

[1] R. Scardovelli and S. Zaleski. Direct Numerical Simulation of Free-Surface and Interfacial Flow. *Annual Review of Fluid Mechanics*, 31:567–603, 1999.

[2] C. W. Hirt, J. L. Cook, and T. D. Butler. A Lagrangian Method for Calculating the Dynamics of an Incompressible Fluid with Free Surface. *Journal of Computational Physics*, 5:103–124, 1970.

[3] B. Maury and O. Pironneau. Characteristics ALE Method for the Unsteady 3D Navier-Stokes Equations with a Free Surface. *International Journal of Computational Fluid Dynamics*, 6:175–188, 1996.

[4] C. W. Hirt, A. A. Amsden, and J. L. Cook. An Arbitrary Lagrangian-Eulerian Computing Method for All Flow Speeds. *Journal of Computational Physics*, 135:203–216, 1997.

[5] H. H. Hu, N. A. Patankar, and M. Y. Zhu. Direct Numerical Simulations of Fluid-Solid Systems Using the Arbitrary Lagrangian-Eulerian Technique. *Journal of Computational Physics*, 169:427–462, 2001.

[6] F. H. Harlow and J. E. Welch. Numerical Calculation of Time-Dependent Viscous Incompressible Flow of Fluid with Free Surface. *Physics of Fluids*, 8:2182–2189, 1965.

[7] C. S. Peskin. Numerical Analysis of Blood Flow in the Heart. *Journal of Computational Physics*, 25:220–252, 1977.

[8] S. O. Unverdi and G. Tryggvason. A Front-Tracking Method for Viscous, Incompressible, Multi-Fluid Flows. *Journal of Computational Physics*, 100:25–37, 1992.

[9] G. Tryggvason, B. Bunner, A. Esmaeeli, D. Juric, N. Al-Rawashi, W. Tauber, J. Han, S. Nas, and Y. J. Jan. A Front-Tracking Method for the Computations of Multiphase Flow. *Journal of Computational Physics*, 169:708–759, 2001.

[10] J. Du, B. Fix, J. Glimm, X. Jia, X. Li, Y. Li, and L. Wu. A Simple Package for Front Tracking. *Journal of Computational Physics*, 213:613–628, 2006.

[11] C. W. Hirt and B. D. Nichols. Volume of fluid (VOF) Method for the Dynamics of Free Boundaries. *Journal of Computational Physics*, 39:201–225, 1981.

[12] W. J. Rider and D. B. Kothe. Reconstructing Volume Tracking. *Journal of Computational Physics*, 141:112–152, 1998.

[13] Y. Renardy and M. Renardy. PROST: A Parabolic Reconstruction of Surface Tension for the Volume-of-Fluid Method. *Journal of Computational Physics*, 183:400–421, 2002.

[14] 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*, 35:1011–1032, 2006.

[15] S. Osher and J. Sethian. Fronts Propagating with Curvature Dependent Speed: Algorithms Based on Hamilton-Jacobi Formulations. *Journal of Computational Physics*, 79:12–49, 1988.

[16] M. Sussman, P. Smereka, and S. Osher. A Level Set Approach for Computing Solutions to Incompressible Two-Phase Flow. *Journal of Computational Physics*, 114:146–159, 1994.

[17] E. Olsson and G. Kreiss. A Conservative Level Set Method for Two Phase Flow. *Journal of Computational Physics*, 210:225–246, 2005.

[18] O. Desjardins, V. Moureau, and H. Pitsch. An Accurate Conservative Level Set/Ghost Fluid Method for Simulating Turbulent Atomization. *Journal of Computational Physics*, 227:8395–8416, 2008.

[19] M. Sussman and E. G. Puckett. A Coupled Level Set and Volume-of-Fluid Method for Computing 3D and Axisymmetric Incompressible Two-Phase flows. *Journal of Computational Physics*, 162:301–337, 2000.

[20] M. Sussman. A Second Order Coupled Level Set and Volume-of-Fluid Method for Computing Growth and Collapse of Vapor Bubbles. *Journal of Computational Physics*, 187:110–136, 2003.

[21] X. Yang, A. J. James, J. Lowengrub, X. Zheng, and V. Cristini. An Adaptive Coupled Level-Set/Volume-of-Fluid Interface Capturing Method for Unstructured Triangular Grids. *Journal of Computational Physics*, 217:364–394, 2006.

[22] B. Maury. Direct Simulations of 2D Fluid-Particle Flows in Biperiodic Domains. *Journal of Computational Physics*, 156:325–351, 1999.

[23] G. D'Avino, M. A. Hulsen, and P. L. Maffettone. Dynamics of Pairs and Triplets of Particles in a Viscoelastic Fluid Flowing in a Cylindrical Channel. *Computers & Fluids*, 86:45–55, 2013.

[24] A. Esmaeeli and G. Tryggvason. Direct Numerical Simulations of Bubbly Flows. Part 1. Low Reynolds Number Arrays. *Journal of Fluid Mechanics*, 377:313–345, 1998.

[25] A. Esmaeeli and G. Tryggvason. Direct Numerical Simulations of Bubbly Flows. Part 2. Moderate Reynolds Number Arrays. *Journal of Fluid Mechanics*, 385:325–358, 1999.

[26] S. Dabiri, J. Lu, and G. Tryggvason. Transition Between Regimes of a Vertical Channel Bubbly Upflow due to Bubble Deformability. *Physics of Fluids*, 25:102110, 2013.

[27] D. Fuster, A. Bagué, T. Boeck, L. Le Moyne, A. Leboissetier, S. Popinet, P. Ray, R. Scardovelli, and S. Zaleski. Simulation of Primary Atomization with an Octree Adaptive Mesh Refinement and VOF Method. *International Journal of Multiphase Flow*, 35:550–565, 2009.

[28] S. Afkhami, A. M. Leshansky, and Y. Renardy. Numerical Investigation of Elongated Drops in a Microfluidic T-Junction. *Physics of Fluids*, 23:022002, 2011.

[29] M. Ohta and M. Sussman. The Buoyancy-Driven Motion of a Single Skirted Bubble or Drop Rising Through a Viscous Liquid. *Physics of Fluids*, 24:112101, 2012.

[30] S. Muzaferija and M. Perić. Computation of Free-Surface Flows Using Interface-Tracking and Interface-Capturing Methods. In *Nonlinear Water Wave Interaction*, pages 1–24. Computational Mechanics Publications, 1998.

[31] O. Ubbink and R. Issa. A Method for Capturing Sharp Fluid Interfaces on Arbitrary Meshes. *Journal of Computational Physics*, 153:26–50, 1999.

[32] M. Darwish and F. Moukalled. Convective Schemes for Capturing Interfaces of Free-Surface Flows on Unstructured Grids. *Numerical Heat Transfer, Part B: Fundamentals*, 49:19–42, 2006.

[33] X. Lv, Q. Zou, Y. Zhao, and D. Reeve. A Novel Coupled Level Set and Volume of Fluid Method for Sharp Interface Capturing on 3D Tetrahedral Grids. *Journal of Computational Physics*, 229:2573–2604, 2010.

[34] 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*, 195:718–742, 2004.

[35] 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*, 58:897–921, 2008.

[36] R. Scardovelli and S. Zaleski. Interface Reconstruction with Least-Squares Fit and Split Eulerian-Lagrangian Advection. *International Journal for Numerical Methods in Fluids*, 41:251–274, 2003.

[37] D. L. Youngs. Time-Dependent Multi-Material Flow with Large Fluid Distortion. In *Numerical Methods for Fluid Dynamics*, pages 273–285. Academic Press, New York, 1982.

[38] J. E. Pilliod and E. G. Puckett. Second-Order Volume-of-Fluid Algorithms for Tracking Material Interfaces. Technical Report LBNL-40744, Lawrence Berkeley National Laboratory, 1997.

[39] J. E. Pilliod and E. G. Puckett. Second-Order Accurate Volume-of-Fluid Algorithms for Tracking Material Interfaces. *Journal of Computational Physics*, 199:465–502, 2004.

[40] G. H. Miller and P. Colella. A Conservative Three-Dimensional Eulerian Method for Coupled Solid-Fluid Shock Capturing. *Journal of Computational Physics*, 183:26–82, 2002.

[41] E. S.-C. Fan and M. Bussmann. Piecewise Linear Volume Tracking in Spherical Coordinates. *Applied Mathematical Modelling*, 37:3077–3092, 2013.

[42] D. B. Kothe, W. J. Rider, S. J. Mosso, J. S. Brock, and J. I. Hochstein. Volume Tracking of Interface Having Surface Tension in Two and Three Dimensions. In *Proceedings of the 34th AIAA Aerospace Sciences Meeting and Exhibit*, pages 1–24, 1996.

[43] S. J. Mosso, B. K. Swartz, D. B. Kothe, and R. C. Ferrell. A Parallel, Volume-Tracking Algorithm for Unstructured Meshes. Technical Report LA-UR-96-2420, Los Alamos National Laboratory, 1996.

[44] J. Mencinger and I. Žun. A PLIC-VOF Method Suited for Adaptive Moving Grids. *Journal of Computational Physics*, 230:644–663, 2011.

[45] T. Marić, H. Marschall, and D. Bothe. voFoam - A Geometrical Volume of Fluid Algorithm on Arbitrary Unstructured Meshes with Local Dynamic Adaptive Mesh Refinement using OpenFOAM. *arXiv:1305.3417*, pages 1–30, 2013.

[46] H. T. Ahn and M. Shashkov. Multi-Material Interface Reconstruction on Generalized Polyhedral Meshes. *Journal of Computational Physics*, 226:2096–2132, 2007.

[47] V. Dyadechko and M. Shashkov. Moment-of-Fluid Interface Reconstruction. Technical Report LA-UR-05-7571, Los Alamos National Laboratory, 2005.

[48] S. J. Mosso, C. Garasi, and R. Drake. A Smoothed Two- and Three-Dimensional Interface Reconstruction Method. *Computing and Visualization in Science*, 12:365–381, 2009.

[49] K. Shahbazi, M. Paraschivoiu, and J. Mostaghimi. Second Order Accurate Volume Tracking Based on Remapping for Triangular Grids. *Journal of Computational Physics*, 188:100–122, 2003.

[50] N. Ashgriz, T. Barbat, and G. Wang. A Computational Lagrangian-Eulerian Advection Remap for Free Surface Flows. *International Journal for Numerical Methods in Fluids*, 44:1–32, 2004.

[51] C. Ivey and P. Moin. Conservative Volume Of Fluid Advection Method on Unstructured Grids in Three Dimensions. Center for Turbulence Research Annual Research Briefs, 2012.

[52] L. Jofre, O. Lehmkuhl, J. Castro, and A. Oliva. A PLIC-VOF Implementation on Parallel 3D Unstructured Meshes. In *Proceedings of the Fifth European Conference on Computational Fluid Dynamics*, pages 1–16, 2010.

[53] B. Parker and D. Youngs. Two and Three Dimensional Eulerian Simulation of Fluid Flow with Material Interfaces. Technical Report 01/92, UK Atomic Weapons Establishment, 1992.

[54] B. P. Leonard and H. S. Niknafs. Sharp Monotonic Resolution of Discontinuities without Clipping of Narrow Extrema. *Computers & Fluids*, 19:141–154, 1991.

[55] M. Darwish. A New High-Resolution Scheme Based on the Normalized Variable Formulation. *Numerical Heat Transfer, Part B: Fundamentals*, 24:353–373, 1993.

[56] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C++*. Cambridge University Press, 2002.

[57] A. Haselbacher and V. Vasilyev. Commutative Discrete Filtering on Unstructured Grids based on Least-Squares Techniques. *Journal of Computational Physics*, 187:197–211, 2003.

[58] J. López and J. Hernández. Analytical and Geometrical Tools for 3D Volume of Fluid Methods in General Grids. *Journal of Computational Physics*, 227:5939–5948, 2008.

[59] A. V. Tuzikov, S. A. Sheynin, and P. V. Vasiliev. Computation of Volume and Surface Body Moments. *Pattern Recognition*, 36:2521–2529, 2003.

[60] D. J. E. Harvie and D. F. Fletcher. A New Volume of Fluid Advection Algorithm: The Stream Scheme. *Journal of Computational Physics*, 162:1–32, 2000.

[61] O. Lehmkuhl, C. D. Pérez-Segarra, R. Borrell, M. Soria, and A. Oliva. TERMOFLUIDS: A New Parallel Unstructured CFD Code for the Simulation of Turbulent Industrial Problems on Low Cost PC Cluster. In *Proceedings of the Parallel CFD Conference*, pages 1–8, 2007.

[62] 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*, 225:2301–2319, 2007.

[63] R. J. LeVeque. High-Resolution Conservative Algorithms for Advection in Incompressible Flow. *SIAM Journal of Numerical Analysis*, 33:627–665, 1996.

3

Parallelization of the Volume-of-Fluid method

Main contents of this chapter have been published in:

L. Jofre, R. Borrell, O. Lehmkuhl, and A. Oliva. Parallel Load Balancing Strategy for Volume-of-Fluid Methods on 3-D Unstructured Meshes. Under review in *Journal of Computational Physics*, 2014.

Abstract. Volume-of-Fluid (VOF) is one of the methods of choice to reproduce the interface motion in the simulation of multi-fluid flows. One of its main strengths is its accuracy in capturing sharp interface geometries, although requiring for it a number of geometric calculations. Under these circumstances, achieving parallel performance on current supercomputers is a must. The main obstacle for the parallelization is that the computing costs are concentrated only in the discrete elements that lie on the interface between fluids. Consequently, if the interface is not homogeneously distributed throughout the domain, standard domain decomposition (DD) strategies lead to imbalanced workload distributions. In this paper, we present a new parallelization strategy for general unstructured VOF solvers, based on a dynamic load balancing process complementary to the underlying DD. Its parallel efficiency has been analyzed and compared to the DD one using up to 1024 CPU-cores on an Intel SandyBridge based supercomputer. The results obtained on the solution of several artificially generated test cases show a speedup of up to $\sim 12\times$ with respect to the standard DD, depending on the interface size, the initial distribution and the number of parallel processes engaged. Moreover, the new parallelization strategy presented is of general purpose, therefore, it could be used to parallelize any VOF solver without requiring changes on the coupled flow solver. Finally, note that although designed for the VOF method, our approach could be easily adapted to other interface-capturing methods, such as the Level-Set, which may present similar workload imbalances.

3.1 Introduction

The numerical simulation of immiscible multi-fluid flows is fundamental to better understand many phenomena of interest in different disciplines such as engineering, hydrodynamics, geophysics or fundamental physics. Typical examples are the simulation of sprays, injection processes, bubbles, breakup of drops, wave motion, etc. These type of flows are characterized by the existence of an interface, separating the different fluids, which needs to be reproduced by the simulation method. So far, different numerical methods exist to reproduce the interface motion. These can be classified into two main groups: interface-tracking and interface-capturing methods. On the one hand, the interface-tracking approaches chase the interface as it moves: (1) defining the interface as a boundary between subdomains of a moving mesh [1–3]; (2) following the Lagrangian trajectories of massless particles [4–6]. On the other hand, the interface-capturing approaches describe the motion of the interface by embedding the different fluids into a static mesh. In particular, from this last group, the two main options of choice are the Volume-of-Fluid (VOF) [7–9] and Level-Set (LS) [10–12] methods, as well as algorithms based on combinations of both. From all these options, this paper is focused on the VOF method. This is based on geometrically reconstruct the fluids interface and, according to it, evaluate the portion of advected volumetric flux corresponding to each fluid. Its major strength is the accuracy achieved by some of its implementations on capturing sharp interfaces and their complex deformation, including breakups, while complying with the volume preservation constraint. This accuracy results in high computational costs. However, in the last decade, with the increase of the available computing power, different interfacial problems have been successfully tackled using it. Examples are the simulation of the drop breakup phenomenon by Renardy [13], the bubble motion by Annaland et al. [14], the solution of wave impact problems by Kleefsman et al. [15] or the numerical study of primary and impinging jet atomizations by Fuster et al. [16], Tomar et al. [17] and Chen et al. [18].

In general, on the simulation of interfacial multi-fluid flows with VOF methods, the computing costs are dominated by the Navier-Stokes (NS) flow solver, and specifically by the solution of the Poisson system derived from the incompressibility constraint. Even so, the cost of the VOF calculations is not negligible at all. Its relative weight depends on different factors, such as the algorithms chosen, the effectiveness of its implementation, the physical case being considered, the type of geometric discretization used, the computing system employed, etc. As an example, on the sequential simulation of the Richtmyer-Meshkov instability [19] with an unstructured tetrahedral mesh of 250K cells, our VOF solver represents 22% of the computing costs. A similar percentage was reported by Le Chenadec and Pitsch [20], on the solution of a diesel jet with a Cartesian mesh of $256 \times 256 \times 1152$ cells. Anyway, beyond the percentage obtained for any particular simulation, it is a certainty that, in the high performance computing context, the cost of the VOF calculations will become

more and more important while the algorithmic solutions adopted disregard parallel performance issues. Besides, by contrast, many efforts are employed by the scientific community on the parallelization of NS flow solvers and, in particular, on Poisson solvers [21,22]. Considering, for example, the aforementioned Richtmyer-Meshkov instability case, with the DD approach we measured a raise of the VOF cost up to 84% when engaging 128 CPU-cores while, with the new parallelization strategy presented in this paper, the percentage is kept at 24%.

The limitations of the standard DD approach can also be observed in the work by Aráujo et al. [23] focused on the 3-D simulation of injection processes. Their tests show a maximum parallel efficiency of 50% with up to 80 CPU-cores, including both the momentum and the VOF solvers. Another study on parallel algorithms for multiphase flows is the work of Sussman [24], based on solving the pressure Poisson equation by means of a multi-level solver and the interface motion through a coupled LS and VOF method [25]. This last work, however, is mainly focused on the performance of the pressure solver and, after all, no more than 16 CPU-cores were used in the parallel performance tests. Surprisingly, we could not find other relevant works on the literature presenting new alternatives for the parallelization of VOF methods.

Broadening the literature search to LS-based interface-capturing approaches, we found an additional parallelization alternative studied by Herrmann [26], which may be adapted to VOF methods. In particular, LS methods require the solution of an extra partial differential equation (PDE) to maintain the interface sharp. Similarly to VOF methods, this interface re-initialization process is not well balanced if the interface is not homogeneously distributed throughout the domain. Herrmann proposes to generate two independently adapted grids for the solution of the flow and interface motion, respectively. While no restrictions are imposed on the Navier-Stokes grid, an equidistant Cartesian grid is adopted for the interface motion solution, with enough resolution to ensure accuracy of the LS method at any part of the domain, avoiding the application of complex adaptive mesh refinement (AMR) algorithms. This configuration also simplifies the LS parallelization since, in order to achieve a good workload balance, tasks can be easily reassigned between parallel processes without geometric information exchange. This approach was tested on the solution of the Zalesak's disk case, obtaining a slightly sub-optimal speedup with up to 128 CPU-cores [26]. In a later work, Herrmann applied the same strategy on a multi-scale Eulerian/Lagrangian two-phase flow algorithm [27], where the LS grid method was used for the Eulerian part, the overall algorithm showed an excellent speedup with up to 2048 CPU-cores.

Therefore, considering the good parallel performance achieved by Herrmann with his load balancing strategy, our purpose has been to develop a similar strategy for the parallelization of VOF methods on general unstructured meshes. Moreover, we solve

both the motion of the flow and of the interface in the same mesh, without imposing any restriction to it. Consequently, the load balancing algorithm and its computing profile undergo major changes with respect to the Herrmann approach. For instance, when a task is reassigned, in the Cartesian case no geometric information needs to be transmitted since the mesh is homogeneous, contrary, in the general case the geometric characteristics of the discrete elements engaged on the task need to be transmitted as well. Additionally, our load balancing approach is based on a precise optimization algorithm, rather than iteratively reassign tasks until some threshold imbalance is reached or the process stalls. Finally, note that although our algorithm has been developed for VOF methods, it could be easily adapted to the parallelization of LS methods on unstructured grids.

Hence, this paper presents a new strategy for the parallelization of VOF methods on unstructured meshes, which is based on a dynamic load balancing process complementary to the DD. The rest of the document is organized as follows: in the next section, the mathematical formulation of the VOF method on unstructured meshes is presented. The standard domain decomposition and our new load balancing parallelization strategy are detailed in Sec. 3.3. An exhaustive analysis and comparison of the parallel performance issues of both methods are presented in Sec. 3.4. Finally, conclusions are drawn in Sec. 3.5.

3.2 Volume-of-Fluid method

Volume-of-Fluid methods capture the fluids interface by embedding it into a fixed grid. In particular, a fraction scalar field, C_k, is defined for each fluid k, determining the fraction of volume that occupies within each grid cell. Basically, $C_k = 0$ for cells that do not contain fluid k, $C_k = 1$ for cells which only contain the k'th fluid, and finally $0 < C_k < 1$ if part but not all of a cell's volume is occupied by the k'th fluid. These cells in which different fluids coexist are referred to as interface cells. Indeed, C_k can be defined as the normalized integral of a fluid's characteristic function $C_k(\mathbf{x}, t)$, such that

$$C_k(\mathbf{x}, t) = \begin{cases} 1 & \text{if there is fluid } k \\ 0 & \text{otherwise,} \end{cases} \tag{3.1}$$

where $\mathbf{x}$ is a position in space and t refers to a time instant. Therefore, for each cell c, its k'th fluid volume fraction value is evaluated as

$$C_k[c, t] = \frac{\int C_k(\mathbf{x}, t) dV_c}{V_c}, \tag{3.2}$$

where V_c refers to the cell volume.

Assuming that the fluids are immiscible, and that their movement is defined by a unique velocity field, i.e., $u^k = \mathbf{u}$ for each fluid k, the interface motion can then be

captured by solving the respective conservation equation

$$\frac{\partial C_k}{\partial t} + \nabla \cdot (C_k \mathbf{u}) = 0. \tag{3.3}$$

Applying the divergence theorem and using a first-order explicit time scheme, the relative discrete equation reads

$$C_k^{n+1} - C_k^n + \frac{1}{V_c} \sum_{f \in F(c)} V_{k,f}^n = 0, \tag{3.4}$$

where the superscript n refers to the discrete time level, $F(c)$ to the set of faces of cell c, and $V_{k,f}$ is the volumetric flow of fluid k across face f.

VOF methods are characterized by the geometric evaluation of the volumetric flows, which is split into two consecutive phases: (1) the interface reconstruction according to the volume fraction fields; (2) the evaluation of the advection of each fluid, in accordance with the velocity field and the interface geometry previously reconstructed. Both phases are described in more detail in the following subsections. Additionally, further details can be found in our previous work [28].

3.2.1 Interface reconstruction

In this work, the fluids interface is reconstructed following the piecewise linear interface calculation (PLIC) approach. This means that within each grid cell, the interface is represented by a plane described with the equation

$$\mathbf{n} \cdot \mathbf{x} - d = 0, \tag{3.5}$$

where $\mathbf{n}$ is a unit normal vector to the plane and d sets its position.

Specifically, we evaluate $\mathbf{n}$ by means of the standard first-order Youngs method [29]. This is based on the normalized gradient of the volume fraction scalar field, C_k, that is

$$\mathbf{n} = \frac{-\nabla C_k}{|\nabla C_k|}. \tag{3.6}$$

In particular, with the aim of obtaining smooth solutions avoiding sharp angles between adjacent planes, we evaluate the gradient by means of a vertex-connectivity least-squares method [30].

Once fixed the unitary normal vector $\mathbf{n}$, d is found by placing the plane at the position that fulfills the initial condition

$$C_k = \frac{V_k}{V_c}, \tag{3.7}$$

where V_k is the volume occupied by fluid k within the cell. Particularly, we perform this search by means of the iterative Brent's minimization method [31].

It is important to note that the interface reconstruction within each cell is an independent process. In other words, for any interface cell, given its geometric description and some values of the field C_k, its interface reconstruction can be evaluated independently. This is a crucial point for our load balancing strategy, since it means that the global reconstruction calculation can be decomposed into unitary tasks, which can be then reassigned through the parallel processes in order to balance the workload. In particular, in the load balancing process we are only reassigning the evaluation of constant d, which is the most time-consuming part of the reconstruction process. Therefore, when the interface reconstruction within a cell is reassigned, the information to be transmitted is the geometric description of the cell and the corresponding values of the fields C_k and ∇C_k.

3.2.2 Interface advection

Once the interface has been reconstructed, the advection is performed by geometrically calculating the volumetric fluxes $V_{k,f}$; see Eq. 3.4. The interface geometry evaluated in the previous step is necessary in order to discriminate, in the zones where two or more fluids coexist, which part of the volumetric flux corresponds to each fluid. Note that when two fluids coexist, it is only necessary to advect one of them, the solution of the other is obtained as the complement. The steps required to evaluate the volumetric fluxes, $V_{k,f}$, at any face f are presented below:

1. *Quantify the total volumetric flux*. The value of the total advection volume is calculated as
$$V_f = |\mathbf{u}_f \cdot \mathbf{n}_f| A_f \Delta t, \tag{3.8}$$
where Δt is the time step, $\mathbf{u}_f$ the velocity at face f, and $\mathbf{n}_f$ and A_f correspond, respectively, to the unit-outward normal and the area of face f. Particularly, in order to limit the stencil of neighboring cells engaged, the CFL restriction is fixed to one. Thus, the flux polyhedron will always be contained in the stencil of cells that share at least one vertex with the face being considered. Consequently, this is the stencil of neighboring cells being used on the calculations.

2. *Construct the polyhedron representing the volumetric flux*. A polyhedron with volume V_f is constructed over face f. In particular, we are employing a vertex-matched approach with the aim to minimize flux over/underlapping situations that degrade the volume conservation principle. This approach is based on setting the direction of the extrusion edges equal to the velocity vectors at the face vertices. A 2-D illustration of it is shown in Fig. 3.1a, while an extended description can be found in [28].

3. *Truncate the part of the volumetric flux polyhedron corresponding to each fluid*. If the polyhedron only contains one fluid, truncation is not necessary, since $V_{k,f}$ is equal to 0 or V_f; this situation is illustrated by cases A and C of Fig. 3.2, respectively. Therefore, in this case, computing costs are negligible. Otherwise, it is necessary to truncate the part of the polyhedron corresponding to each fluid; case B in Fig. 3.2. This operation is performed independently on each cell of the face neighboring cells stencil; see Fig. 3.1b. In particular, three actions are performed for each of these neighboring cells: (1) evaluate its intersection with the flux polyhedron; (2) if it is an interface cell, truncate the resulting polyhedron by the interface plane; (3) add to $V_{k,f}$ the volume of the polyhedron resulting from the two previous actions. Note that the basic geometric operation used in the first two steps is the truncation of a polyhedron by a plane. A general algorithm to perform it is described in the work by López et al. [32].

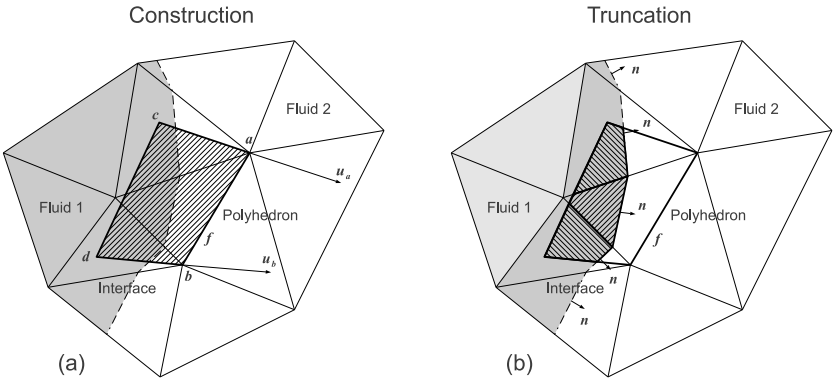


Figure 3.1: (a) Construction of the total volumetric flux polyhedron (**abdc**), point **c** is evaluated by tracing back the Lagrangian trajectory of point **a** for the time step Δt, i.e., $\mathbf{c} = \mathbf{a} - \Delta t \mathbf{u}_a$; idem for point **d**. (b) Truncation of the part of the volumetric flux polyhedron corresponding to fluid 1.

As in the reconstruction phase, the evaluation of the fluids advection through any face is an independent process. Therefore, the advection calculation can also be decomposed into unitary tasks. Note that in the load balancing process, we only count as unitary task the evaluation of the volumetric fluxes at faces with a neighboring interface cell. As explained above, the other cases are trivial and irrelevant in terms of computing cost. In particular, the information required to evaluate the fluids advection through any face is: the velocity vector at its vertices and, for all the elements of the stencil of neighboring cells, its geometric description, the respective volume fraction value and, in the case of being an interface cell, the interface reconstruction plane.

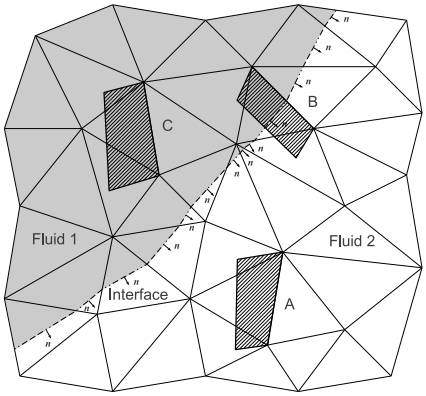


Figure 3.2: Possible initial situations on the evaluation of the fluids advection: (1) The volumetric flux polyhedron contains only one fluid (cases A and C); (2) Different fluids coexist within the volumetric flux polyhedron (case B).

3.3 Parallelization strategy

3.3.1 Standard domain decomposition

The domain decomposition is a standard strategy for the parallel solution of PDEs. The initial discretized domain is divided into P subdomains with similar number of cells, distributed then between P parallel processes to perform the computations. The subset of discrete elements assigned to a parallel process is referred as its *owned* elements, while the rest of elements are named *external*. Thus, for any parallel process, we may talk about owned cells, owned nodes, owned components of a scalar field, external nodes, external faces, etc. Since the system of equations generally links unknowns owned by different subdomains, to perform calculations in parallel is necessary the transmission of data between parallel processes. Here we refer to the external elements required by any parallel process as its *halo* elements. Each parallel process obtains its halo elements from neighboring subdomains by means of communications throughout the network, referred to as halo updates. In particular, note that a halo element that varies on its owner parallel process needs to be updated before being used, otherwise, the parallel and sequential executions would differ.

The DD approach has been extensively used in many VOF-based codes for the simulation of immiscible multi-fluid flows; see for example [33–35]. Using the DD for the VOF calculations is relatively simple, since it is just necessary to define the halo requirements of the parallel processes and introduce some halo updates. Moreover, since the rest of calculations, like the solution of the Navier-Stokes equations, may be

parallelized using the same DD strategy, it becomes easy to assemble the solution of the whole system. In particular, two halo updates are needed to solve Eq. 3.4: (1) to the volume fraction scalar fields, before the reconstruction phase; and (2) to the field composed of the interface reconstruction planes, before the advection phase.

Since in the VOF calculations the work is concentrated on the discrete elements around the fluids interface, the workload of the parallel processes will only be well balanced if the interface is homogeneously distributed through the different subdomains. Unfortunately, the contrary occurs in many situations. For example, the simulation of gas bubbles within a liquid media may produce really imbalanced distributions or, in hydrodynamics simulations, the sea surface is generally located in a specific zone of the domain, involving only the subdomains covering it. In particular, Fig. 3.3 illustrates an imbalanced situation for a simplified case where two fluids coexist in a discrete domain divided in four parts.

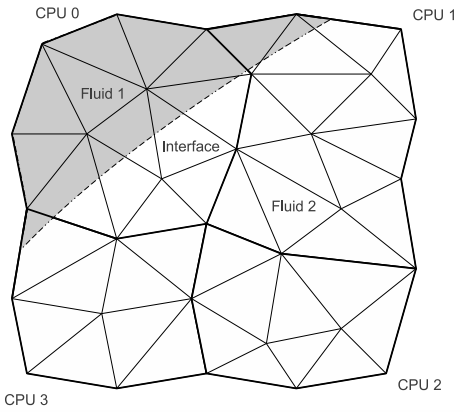


Figure 3.3: Decomposition of an unstructured grid where two fluids coexist. The interface between fluids is not homogeneously distributed throughout the domain.

In order to overcome the degradation of the parallel performance produced by the load imbalance, a possible strategy is to adapt the mesh partition to the interface distribution. In cases with predictable and constant interface location this adaptive strategy can be very convenient. However, in a general case some drawbacks appear: (1) the location of the interface may be not known a priori; (2) it may vary during the simulation, having to readapt the domain partitions; (3) the VOF adapted partition may be inappropriate or perform poorly for other parts of the code. For example, the numerical simulation of gas bubbly flows [14] requires, usually, a random initialization of the bubbles pattern inside the domain. Thus, in these cases the adapted mesh partition cannot be evaluated a priori. Moreover, any possible adapted partition

would no longer suit the pattern of the bubbles as they evolve in time, having to readapt the partition several times. In addition, it has been found that adapting the mesh partition to the interface distribution, instead of prioritizing the minimization of halo requirements, optimizes the parallelization of the VOF algorithm but can decrease significantly the parallel performance of the Navier-Stokes solver [36].

3.3.2 New parallelization strategy

We propose a new parallelization strategy based on a dynamic load balancing process that reduces the common imbalance obtained from the standard domain decomposition. With this objective in mind, the reconstruction and advection unitary tasks are transported between different parallel processes overpassing the initial mesh partition. Consequently, when an unitary task is reassigned to a new parallel process, all the discrete data required to perform it (geometric and algebraic information), need to be transported to the new committed parallel process.

Note that, in the advection process, the geometry of the interface around any face is required in order to discriminate the portion of its volumetric flux corresponding to each fluid (terms $V^n_{k,f}$ of Eq. 3.4). This coupling between the resconstruction and advection phases makes it difficult to perform only one communication episode for all the algorithm. Indeed, a second level of data transfer, after the interface reconstruction and before the advection, seems inevitable to ensure the availability of the interface geometry around any face through which two or more fluids are advected. Under these circumstances, in order to avoid a complex data interdependence management and better adjust the result, we prefer to perform separately the load balancing of the reconstruction and advection phases.

The load balancing algorithm presented in this work consists in the five main steps outlined in the next items. Further details about them are given in the subsections below.

1. *Determine the workload*. Each parallel process, p, evaluates its workload, W_p. When the cost of the tasks is variable, weights are used in order to optimize the accuracy of the assigned loads. Further details are given in Sec. 3.3.2.

2. *Define a new balanced distribution*. This is performed in two steps. First, an optimal workload per parallel process, W_{opt}, is determined taking into account possible overheads on the solution of the tasks being reassigned; see Sec. 3.3.2. Second, a new tasks distribution is determined according to the previous load per process target. The corresponding algorithm, namely Alg. 3, defines also the communication scheme to transfer the data.

3. *Move data*. The data needed to perform the reassigned tasks is transported, through the local memories of the parallel processes involved in the solution,

according to the scheme determined in the previous step. This redistribution is performed by means of non-blocking point-to-point communications. However, to avoid inconsistencies, any parallel process does not start the next step until the communications in which it is involved are completed. Buffering is used to group all the data transactions between two parallel processes in only one message, using an independent buffer for each communication.

4. *Solve VOF tasks*. These tasks may be a set of interface reconstructions within interface cells, or fluids advection evaluations at faces around the interface. The parallel processes committed to solve both external (received from other parallel processes) and owned tasks, start with the solution of the external ones. In this way, the communications required to send back the results to the owner processes can be overlapped with the solution of the owned tasks.

5. *Collect solutions*. The processes which reassigned part of their tasks to others, receive the solutions back in buffers and store them in the corresponding memory space.

To summarize, the main steps of our load balancing strategy are outlined in Alg. 1. Remaining details are attained in the following subsections. Note that Alg. 1 is applied twice: first on the reconstruction phase and, afterward, on the advection phase.

Algorithm 1 Parallel load balancing strategy

1: Determine the workload
2: Define a new balanced distribution
3: Move data
4: Solve VOF tasks
5: Collect solutions

Analysis of the algorithm

The diagram shown in Fig. 3.4 illustrates the computing time distribution for the VOF algorithm using the new parallelization strategy. In particular, the test case represented is a translation applied to 64 spheres contained in a cubic domain discretized by means of an unstructured mesh of 1000K cells; see Fig. 3.8.b. This test was executed using 128 CPU-cores. Note that the advection costs dominate the VOF execution, while the overhead produced by the load balancing is around $\sim$5%. In the rest of tests presented in the next section, we have observed that the relative weight of the load balancing step varies with the number of parallel processes engaged. On the contrary, the ratio between the reconstruction and advection phases has shown to be almost constant, meaning that same parallel performance is obtained for both.

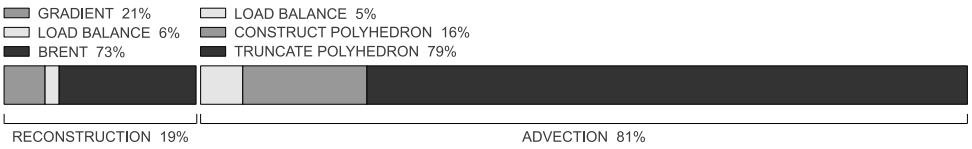


Figure 3.4: Computing time distribution for the new parallelization strategy.

More in detail, Fig. 3.5 shows the distribution of the computing time through the different steps of Alg. 1 for the advection phase. The left part of the figure illustrates the flowchart for a parallel process *overloaded*, i.e., which reassigns some of its tasks to others. While the right part represents an *underloaded* parallel process, receiving tasks from the overloaded ones. The height of each rectangular box is proportional to the cost of the corresponding step of Alg. 1. The communications between both groups are illustrated with lines or boxes across the two columns. These occur in steps 3 and 5 ("Move data" and "Collect solutions"). Note that the communications of step 5 are asynchronous and, consequently, are represented by means of a line that couples different levels of the flowcharts. Step 2 ("Define a new balanced distribution") is also represented with a horizontal box across both columns because collective communications are required to perform it. These three steps constitute the part of the algorithm which increases its cost with the number of parallel processes. Therefore, it becomes a degradation factor for the speedup. The rest of the algorithm can be executed independently by each parallel process and reduces its cost when the number of parallel processes increases. More details on these aspects are shown in the numerical tests of Sec. 3.4.

The main difference in the flowchart of the overloaded and underloaded parallel processes occurs around the data movement of step 3 ("Move data"). Before it, the first ones pack in buffers the information to be sent while the last ones become idle. After it, the underloaded parallel processes need to unpack the required information from the received buffers before performing any external VOF task. Note that the tasks distribution can be balanced in order to compensate the overcosts produced by the unpacking operations and, hence, reduce idle times; see Sec. 3.3.2. The same situation is repeated on the communication required to collect the solution of the reassigned tasks in step 5 ("Collect solutions"). However, in this case, the size of the communication is much smaller and its cost, compared with the one of the pack and unpack operations, is almost negligible. For this reason, they are all represented by means of a simple line.

Finally, note that all the steps of Alg. 1, except the solution of the VOF tasks (step 4), can be considered pure overcosts, since they are not part of the solution itself but part of the balancing process. However, in the next section it is demonstrated that these overcosts are widely outweighed by the gain achieved with the load balancing.

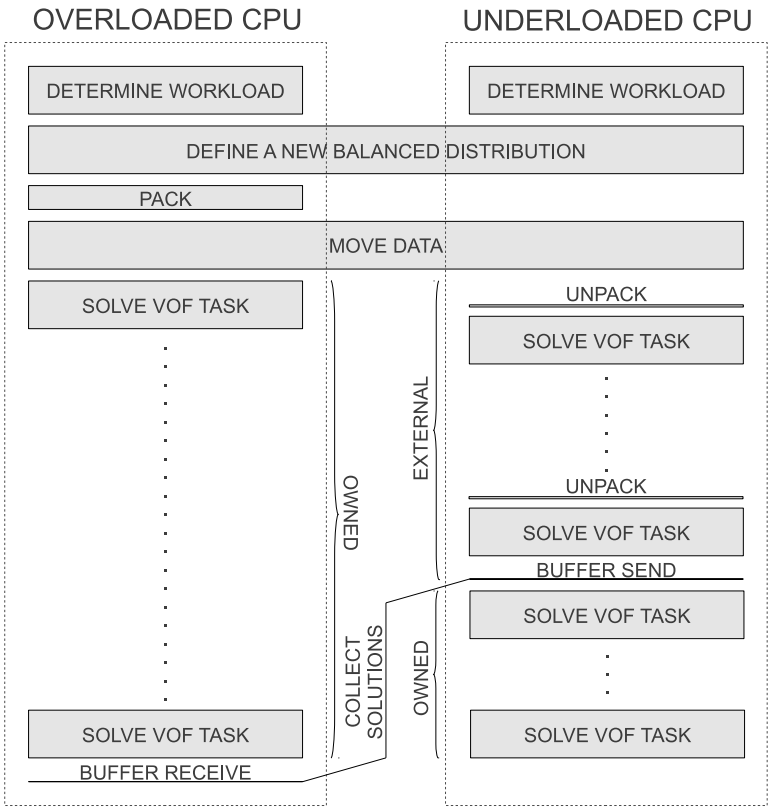


Figure 3.5: Flowchart of the advection process, from the perspective of an "overloaded CPU" (left) and an "underloaded CPU" (right). The height of each rectangular box is proportional to the cost of the corresponding step of the algorithm.

Buffering

The geometric and algebraic data required to perform the VOF tasks are heterogeneous and not continuously stored in memory. Consequently, in order to move them through the network, we have explicitly defined pack functions, to store them into communication buffers, and unpack functions, to read the received information and reconstruct the stored objects before performing the calculations. Moreover, buffers are also used to group all data moves between two parallel processes in only one message, and thus reduce latency costs.

In particular, we have optimized our implementation of the pack and unpack

functions for unstructured tetrahedral meshes, which is the type of meshes that we have used in the numerical experiments. In this case, in order to reassign a reconstruction task, 16 floating point elements need to be sent: 12 accounting for the vertices components, 1 for the volume fraction value and 3 for the gradient of the volume fraction field; an example is illustrated in Fig. 3.6. Note that the faces of a tetrahedron are just defined by the different combinations of its vertices, therefore, it is not necessary to explicitly determine its composition. Similar strategies are possible for prismatic cells, however, in a general case with more complex polyhedra, information of the faces composition may be required for the cell description into the buffer.

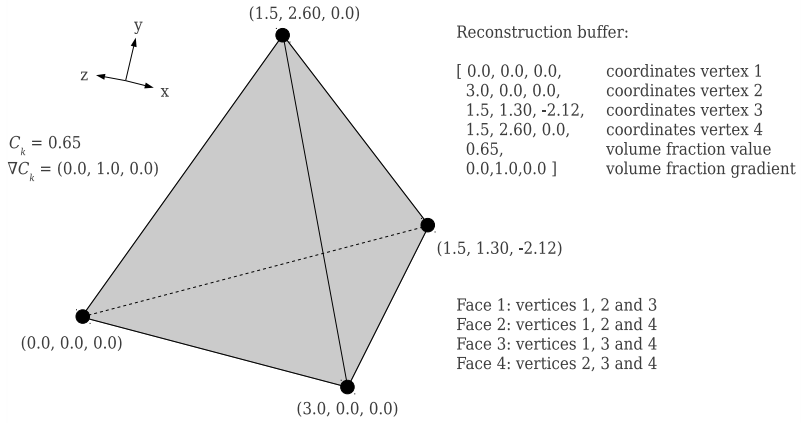


Figure 3.6: Illustration of the data packed into the communication buffer for a reassigned reconstruction task.

On the other hand, the advection tasks require many more elements to be transmitted. For each mesh face, any element of the stencil of neighboring cells sharing at least one vertex with it, could be engaged on the calculation of the fluids advection through it. However, in order to minimize the communication costs, we try to discard some of the neighboring cells that are not required for the calculations. In particular, we can restrict to the neighboring cells that: (1) contain the fluid being considered and (2) have at least one vertex at the upstream side of the face plane (with respect to the flux), since the volumetric flux polyhedron is built into that side; see Fig. 3.7. Therefore, for each reassigned advection task are packed: 9 floating point elements, describing the components of the velocity field in the face vertices, and up to 17 floating point elements for each engaged neighboring cell —12 to describe its geometry, 1 for its volume fraction and, in case of interface cell, 4 more defining the interface plane.

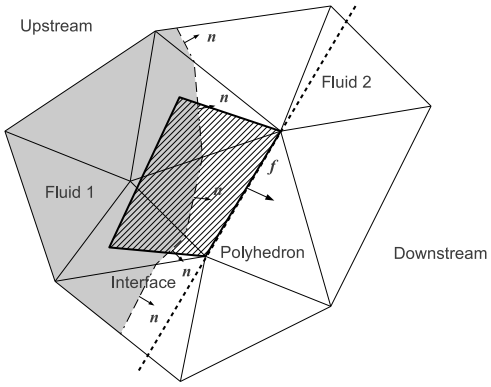


Figure 3.7: Representation of the flux polyhedron used on the fluids advection evaluation at face f.

Weight of a task

In the first step of Alg. 1 each parallel process evaluates its workload, W_p. This is performed by assigning a weight to each owned unitary task and then adding up all these weights.

In the reconstruction phase, different weights are not necessary because reconstructing the interface has almost the same cost for any interface cell. In this case, for each parallel process, p, the workload can be set equal to the number of its owned unitary tasks, N_p.

A different situation occurs in the evaluation of the fluids' advection. As explained in Sec. 3.2.2, the advection calculation in a face only has a significant cost when its flux polyhedron intersects an interface cell. The evaluation of the advection for the faces matching this condition composes the set of unitary tasks to be distributed. At any of these faces, the advection evaluation requires geometric calculations with its neighboring cells that meet three conditions: (1) share at least one vertex with the face, (2) contain the fluid being advected and (3) have no null intersection with the advection polyhedron. The number of cells of this subset may be a good approach for the weight of the corresponding unitary task. Nevertheless, since the evaluation of the workloads is not part of the solution but just part of the process to find a good distribution, it must be a relative fast process. On this regard, constructing the advection polyhedron and checking its intersection with the neighboring cells (condition 3) would be too costly. Consequently, when defining the weight of a task, we substitute condition 3 by the less restrictive but easier to evaluate condition: "being in the upstream side of the face" —the same condition used in the pack

function described in the previous subsection. For example, in Fig. 3.7, considering the advection of fluid 1, with our approximation the relative weight of the task represented would be 5, while the precise (but too costly to evaluate) weight is 3. Even so, as shown in the numerical tests, using weights has a clear positive impact on the load balancing of the advection phase.

Overcost of external tasks

On the "Solve VOF tasks" step of Alg. 1, external tasks have an additional cost due to the buffers unpacking. This overcost should be taken into account when defining the new balanced distribution. We introduce it by means of a coefficient α, such that when a task is reassigned the cost of its solution is multiplied by $(1 + \alpha)$. Note that only the overcost produced by the unpack process is taken into account. The communication costs are not included in the definition of α because affect both overloaded ("senders") and underloaded ("receivers") parallel processes, so they do not produce an additional imbalance. On the other hand, the imbalance produced by the pack process, which is executed only in the overloaded processes, cannot be compensated with a proper tasks distribution, because the subsequent communication synchronizes the parallel processes.

The coefficient α depends on the ratio between the cost of executing a VOF task, and the cost of the process of unpacking the data required to perform it. In the reconstruction phase, both magnitudes are almost constant for all reassigned tasks. On the contrary, both are variable in the advection phase. In this case, we evaluate α as an average overcost for all the reassigned tasks. Note that α mainly depends on the type of grid (e.g., orthogonal or tetrahedral), the VOF algorithm implemented and the computing equipments being used. While, on the other hand, since we are considering unitary tasks, α is independent of the mesh size, of the number of parallel processes and of the interface size and distribution within the domain. Under these conditions, we measure α by running a test with a rather coarse mesh and few parallel processes. This measurement is then valid for any execution with the same equipment and mesh type. In particular, for all the numerical tests shown in Sec. 3.4, the value of α for the reconstruction and advection phases was fixed to 0 and 0.1, respectively. Therefore, with the algorithms being used, in the reconstruction phase, the cost of the unpacking process is negligible, compared to the cost of finding the linear reconstruction of the interface (which requires an iterative Brent's root search). While, in the advection phase, the average overcost produced by the unpacking process represents 10%.

Define a new balanced distribution

The second step of Alg. 1 is divided in the two substeps described in detail below.

Optimal workload per CPU. We first determine the optimal workload per parallel process, W_{opt}, independently of the particular reassignment of tasks required to achieve it.

When the cost of the external and owned tasks is equivalent ($\alpha = 0$), the theoretical optimum, referred as W^*_{opt}, is the average workload

$$W^*_{opt} = W_{avg} = \frac{\sum_{p=0}^{P} W_p}{P}. \tag{3.9}$$

Nevertheless, as stated in the previous subsection, in general, external tasks may suffer an overcost produced by the unpack process, which multiplies its cost by $1 + \alpha$, with $\alpha \geq 0$. This means that in general $W^*_{opt} \geq W_{avg}$. Under this circumstances, given an initial distribution, finding an optimal redistribution or, what is the same, an optimal workload per parallel process, becomes a NP-complete problem. Equivalent formulations of it can be found in [37]. Therefore, we have to focus on heuristic approaches.

In the present application, there is an important advantage: given an estimated optimal workload per process, W_{opt}, it can be easily determined if it is higher or lower than the theoretical optimum, W^*_{opt}. On the one hand, the leftover workload from parallel processes with $W_p > W_{opt}$ is

$$L\left(W_{opt}\right) = \sum_{p=0}^{P} max(0, W_p - W_{opt}). \tag{3.10}$$

On the other hand, the workload required to reach W_{opt} by the underloaded processes, is

$$R\left(W_{opt}\right) = \sum_{p=0}^{P} \frac{max(0, W_{opt} - W_p)}{1 + \alpha}. \tag{3.11}$$

Finally, the balance is

$$B\left(W_{opt}\right) = L\left(W_{opt}\right) - R\left(W_{opt}\right). \tag{3.12}$$

Hence, if $B\left(W_{opt}\right) = 0$, the optimal workload has been found. Otherwise, if $B\left(W_{opt}\right) > 0$, it means that $W_{opt} < W^*_{opt}$ and, finally, $B\left(W_{opt}\right) < 0$ indicates that $W_{opt} > W^*_{opt}$. In fact, $B\left(W_{opt}\right)$ is a continuous function and the more closer to zero is its value, the better is the corresponding approximation. Under these circumstances, a root finding algorithm can be used in order to approach the theoretical optimum. In particular, we adopt the simple and well known bisection method, detailed in Alg. 2.

The bisection method requires two initial guesses, W^a_{opt} and W^b_{opt}, such that $B\left(W^a_{opt}\right)$ and $B\left(W^b_{opt}\right)$ have different sign. On the one hand, we can take $W^a_{opt} =$

Algorithm 2 Iterative calculation of the optimal weight per process

1: $W^a_{opt} = W_{avg}$
2: $W^b_{opt} = (1+\alpha)W_{avg}$
3: **for** $0 \leq i < numIte$ **do**
4: **if** $B((W^a_{opt} + W^b_{opt})/2) < 0$ **then**
5: $W^b_{opt} = (W^a_{opt} + W^b_{opt})/2$
6: **else**
7: $W^a_{opt} = (W^a_{opt} + W^b_{opt})/2$
8: **end if**
9: **end for**
10: $W_{opt} = (W^a_{opt} + W^b_{opt})/2$

W_{avg} with $B(W^a_{opt}) \geq 0$, since, as previously stated, $W_{avg} \leq W^*_{opt}$. On the other hand, given an initial tasks distribution and considering $\alpha > 0$, note that the optimal workload depends on the percentage of tasks that need to be reassigned. The larger the movements required, the larger the number of tasks that multiply its cost by $1+\alpha$ and, consequently, the larger becomes W^*_{opt}. In particular, if all the tasks were reassigned, the theoretical optimum would be $(1+\alpha)W_{avg}$. However, this extreme is not possible because some tasks will always remain in its owner parallel process. Hence, we can affirm that $W^*_{opt} \leq (1+\alpha)W_{avg}$ and, therefore, $B((1+\alpha)W_{avg}) \leq 0$. In conclusion, we can take as initial guesses $W^a_{opt} = W_{avg}$ and $W^b_{opt} = (1+\alpha)W_{avg}$. Note that the length of the initial interval is αW_{avg}. In our case, this is 0 and $0.1W_{avg}$ for the reconstruction and advection phases, respectively. Therefore, in the advection case, since the initial maximal error is 10% and each iteration of the bisection method halves it, we can affirm that in 4 iterations the error of our approach is less than 1%. This precision is more than enough for our application context and, what is more, the cost of these 4 iterations is almost negligible compared to the overall solution time. For the reconstruction it is not necessary any iterative process, being $\alpha = 0$ the optimal solution is just the average workload.

Tasks reassignment algorithm. Once an optimal workload per parallel process is calculated with the algorithm defined above, it is necessary to determine a new distribution of tasks fulfilling it. With the aim of better understandability, in Alg. 3 we first describe this process for the case of tasks with equal cost.

For each parallel process p the only input of the Alg. 3 is its initial workload $W_p = N_p$, while the output are two arrays, *SendTo* and *RecvFrom*, of dimension P, storing in the k'th position the number of tasks to be sent and to be received to/from

process k, respectively. For instance, $SendTo[0] = 5$ would mean that the process being considered, i.e., process p, has to reassign 5 of its owned tasks to process 0. Note that, since we are assuming that all tasks have the same cost, it is not relevant which particular tasks are redistributed.

At the first line of the algorithm, a collective all-gather communication is performed in order to get the whole interface distribution on each parallel process. This is stored in an array I such that $I[k] = N_k$. The rest of the algorithm is executed independently at each parallel process. This implies that some calculations are repeated but, since their cost is very low, it is more efficient to replicate calculations rather than using communications.

In the second line of Alg. 3 it is executed Alg. 2, described in the previous subsection, in order to find the optimal workload W_{opt}.

From lines 3 to 7, the vectors S and R of dimension P are evaluated, containing in the k'th position the number of owned tasks to be sent (reassigned) and the number of external tasks to be received by the k'th parallel process, respectively. Their evaluation is straightforward from the comparison of $I[k]$ with N_{opt}, where N_{opt} refers to the closest integer to W_{opt}. Some adjustment may be necessary to minimize the errors produced by the integer round-offs. Note that for any $k \in [0, ..., P-1]$, it is not possible that both $S[k]$ and $R[k]$ are different than zero. For instance, $S[k] > 0$ indicates that process k is overloaded, i.e., $I[k] > N_{opt}$. Thus, some of its tasks need to be reassigned to other processes. Obviously, this means that it does not require additional external tasks, i.e., $R[k] = 0$. In the same way, if $R[k] > 0$ then $S[k] = 0$. Finally, at line 7, the total number of tasks to be reassigned on the load balancing process, referred as N_{re}, is evaluated as $N_{re} = \sum_p S[p]$, which equals $\sum_p R[p]$.

In the next loop of the algorithm, lines 8-17, the reassignment of tasks is organized. In detail, for each of the N_{re} tasks that need to be reassigned, an overloaded and an underloaded parallel process are committed to send and receive it, respectively. This information is stored in the arrays $SendTask$ and $RecvTask$, of dimension N_{re}, storing in the i'th position the rank of the process sending and receiving the i'th reassigned task, respectively. There is not a unique form to organize this redistribution, in this case we arrange it by the rank of the parallel processes.

Finally, once the overall tasks redistribution is defined, the evaluation of the vectors $SendTo$ and $RecvFrom$, which define the particular communications involving process p, is straightforward. This is performed in the last loop of the algorithm, corresponding to lines 18-22.

In the case of tasks with different computing costs additional complexities need to be considered. In particular, the new distribution is defined according to the weight of the tasks being reassigned. This was not necessary in the previous case since all tasks had the same cost. The new implementation is shown in Alg. 4. For each parallel process, the inputs of the algorithm are its initial workload, W_p; its number of owned

elements, N_p; and an array of dimension N_p containing the weight of each owned task, WI. The outputs are the same *SendTo* and *RecvFrom* vectors obtained with Alg. 3. In fact, the second and third loops of Alg. 3, used to determine the overall distribution of tasks and the particular movements involving process p, are repeated at the end of Alg. 4. The difference between both algorithms is on the determination of vectors S and R. The steps of the new algorithm are described next.

Algorithm 3 Tasks reassignment for process p (task weights not considered)

1: **AllGather** communication of initial tasks distribution: $I[k] = N_k$
2: Apply **Alg. 2** to find W_{opt}
3: **for** $0 \leq k < P$ **do**
4: $aux = min(N_{opt}, I[k])$
5: $S[k] = I[k] - aux$
6: $R[k] = min\left(0, \frac{N_{opt} - aux}{1+\alpha}\right)$
7: **end for**
8: $N_{re} = \sum_k S[k]$
9: $count\_send = count\_recv = 0$
10: **for** $0 \leq k < P$ **do**
11: **if** $S[k] > 0$ **then**
12: **for** $0 \leq i < S[k]$ **do**
13: $SendTask[count\_send] = k$
14: $++count\_send$
15: **end for**
16: **else**
17: **for** $0 \leq i < R[k]$ **do**
18: $RecvTask[count\_recv] = k$
19: $++count\_recv$
20: **end for**
21: **end if**
22: **end for**
23: **for** $0 \leq i < N_{re}$ **do**
24: **if** $SendTask[i] == p$ **then**
25: $++SendTo[RecvTask[i]]$
26: **end if**
27: **if** $RecvTask[i] == p$ **then**
28: $++RecvFrom[SendTask[i]]$
29: **end if**
30: **end for**

Algorithm 4 Tasks reassigment for process p (weights considered)

1: **AllGather** communication of initial workload distribution: $W[k] = W_k$
2: Apply **Alg. 2** to find W_{opt}
3: **if** $W_p > W_{opt}$ **then**
4: $count\_solve = count\_weight = 0$
5: **for** $0 \leq i < N_p$ **do**
6: **if** $(count\_weight + WI[i]) \leq W_{opt}$ **then**
7: $count\_weight += WI[i]$
8: $++count\_solve$
9: **else**
10: break;
11: **end if**
12: **end for**
13: **for** $count\_solve \leq i < N_p$ **do**
14: $WI_{re,p}[i - count\_solve] = WI[i]$
15: **end for**
16: $S[p] = N_p - count\_solve$
17: **else**
18: $S[p] = 0$
19: **end if**
20: **AllGather** communication to get the whole vector S on each parallel process
21: **AllGatherv** communication of reassigned tasks weights: $WI_{re} = \bigoplus_k WI_{re,k}$
22: $N_{re} = \sum_k S[k]$
23: $count\_recv = 0$
24: **for** $0 \leq k < P$ **do**
25: $R[k] = 0$
26: $recv\_weight = W_{opt} - W[k]$
27: **if** $recv\_weight > 0$ **then**
28: **for** $count\_recv \leq i < N_{re}$ **do**
29: **if** $(recv\_weight - (1 + \alpha)WI_{re}[i]) \geq 0$ **then**
30: $++count\_recv$
31: $++R[k]$
32: $recv\_weight -= (1 + \alpha)\ WI_{re}[i]$
33: **end if**
34: **end for**
35: **end if**
36: **end for**
37: *...lines 8 - 22 of* **Alg. 3**

At the first line of Alg. 4, a collective all-gather communication is performed in order to get the workload distribution vector, W, such that $W[k] = W_k$. Subsequently, in the second line, it is executed the Alg. 2 described in the previous subsection in order to find the optimal workload, W_{opt}.

In the next loop, lines 3-15, are evaluated: (1) $S[p]$, the number of tasks being reassigned (sent) by the executing process, i.e., process p; (2) $WI_{re,p}$, a vector of dimension $S[p]$ containing the weight of the particular tasks being reassigned by process p. The criteria used is that the overloaded parallel processes reassign the last elements of its tasks list, so the values of $WI_{re,p}$ correspond to those elements.

The whole vector S is then obtained at each parallel process by means of an all-gather communication, line 16. However, in order to evaluate R we also need to gather the weight of all the tasks being reassigned. This is performed by means of a "vector" all-gather communication rather than a simple one, line 17, because the size of the vectors $WI_{re,k}$ is variable. In fact, this information is contained in vector S, which is used to define the collective communication. As a result of the communication, the complete set of weights of tasks being reassigned are obtained in the vector WI_{re}, arranged in ascending number of the owner parallel process.

In the next loop, lines 18-28, vector R is evaluated according to the values of WI_{re} and the initial load of each parallel process. Note that the coefficient $(1 + \alpha)$ is used in order to take into account the unpacking overcosts. Finally, as stated above, once S and R have been evaluated, the algorithm continues with the last two loops of Alg. 3.

3.4 Numerical tests

In this section, the new load balancing (LB) strategy is tested and compared with the standard DD approach. Both methods have been implemented within the *TermoFluids* (TF) parallel CFD software platform [38]. Therefore, its comparison accounts only for differences on the parallelization strategy. Tests have been performed on the *IBM MareNostrum-III* supercomputer at the *Barcelona Supercomputing Center* [39]. *MareNostrum-III* is based on Intel SandyBridge 8-core processors at 2.6 GHz (2 per node), iDataPlex Compute Racks, a Linux Operating System and an Infiniband FDR10 interconnection network. The number of CPU-cores engaged in our numerical experiments ranges between 16 and 1024 units.

Since we are only interested in parallel performance issues, we consider a canonical test case consisting of a translation applied to a set of spheres, which represent an interface between two fluids, and are placed in a cubic domain; see Fig. 3.8. In this way, we can easily control the size and distribution of the interface within the domain, and measure their influence on the parallel performance. In a general case, the interface may be deformed by a shifting velocity field. However, the computing pattern of the VOF part of the code would be exactly the same than the one of our

canonical test case. Therefore, the conclusions about the parallel performance of VOF algorithms derived from this paper are generic.

Our measurements have been obtained after averaging over several iterations of the same time step in order to avoid dispersion by canceling outlier results. In particular, the translation applied is defined by the vector $\mathbf{u}_t = 1/\sqrt{3}(1,1,1)$, the radius of the spheres is 0.0425 and they are uniformly distributed in a $1\times1\times1$ cubic domain. Unless otherwise stated, the underlying geometric discretization is a mesh of 1000K tetrahedral cells, and the domain decomposition is performed by means of the graph partitioning tool METIS [40]. In Fig. 3.8, three interface configurations used in the following numerical experiments are shown. Moreover, their detailed characteristics are given in Tab. 3.1.

Name	No. interface cells	% interface cells
$2\times2\times2$	3120	0.3
$4\times4\times4$	25254	2.5
$8\times8\times8$	204778	20.0

Table 3.1: Detailed characteristics of different interface configurations used in the numerical experiments.

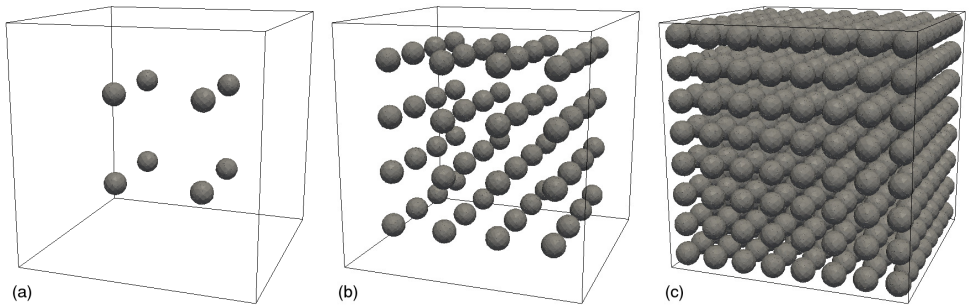


Figure 3.8: Representation of different grids of spheres, which define the interface between two fluids, used in the numerical experiments: (a) $2\times2\times2$, (b) $4\times4\times4$ and (c) $8\times8\times8$.

The first test considered is the strong speedup of the complete VOF algorithm using the standard DD approach. Note that with the DD strategy, acceleration can only be achieved when the overall domain partition further splits the interface and, consequently, divides the VOF computing costs. Results are shown in Fig. 3.9 for the interface configurations mentioned above, ranging the number of CPU-cores between 16 and 1024. Two a priory expected trends are clearly observed: (1) the strong speedup

improves with the size and the extension covered by the interface within the domain; (2) increasing the number of parallel processes engaged in the execution implies that the parallel efficiency (PE) falls. In particular, regarding the second trend, for the $2\times2\times2$ configuration the PE decreases from 59% (with 32 CPU-cores) down to 3% (with 1024 CPU-cores). In fact, the total acceleration achieved in this case from 16 to 1024 CPU-cores is around $2\times$, while the number of parallel processes increases $64\times$. Note that in this case the interface is relatively very small and concentrated around eight points; see Fig. 3.8a. The situation improves when the interface covers a larger part of the domain and, consequently, a larger percentage of CPU-cores become involved in the VOF calculations: for the $4\times4\times4$ configuration the PE varies from 87% to 11%, and for the $8\times8\times8$ one from 92% to 48%. Note also that with 1024 CPU-cores the workload per parallel process is rather low: in ascending order of number of spheres, the ideal workload per process would be around 3, 24 and 192 interface cells, respectively. This fact relativizes the poor results achieved with the highest number of CPU-cores for the coarser interfaces. However, these cases allow us to better analyze aspects of the speedup degradation that may become hidden when the computing costs dominate.

By looking a little deeper into the causes that degrade the acceleration of the DD approach, we have, on the one hand, the effects of the poor workload distribution and, on the other hand, the cost of the communications required on the halo updates. The influence of the second aspect is considered in Fig. 3.10, where the percentage of the communications cost over the total cost of the VOF algorithm is presented. Again, the general picture looks as expected: (1) the percentage of the communications cost grows with the number of CPU-cores; (2) when the size of the interface grows, the relative weight of the communications falls. Nevertheless, the most relevant aspect shown in Fig. 3.10 is that in all cases the percentage of the communications cost is below 1.7%. The subsequent conclusion is that the communications cost is negligible compared to computations. Therefore, the acceleration depends only on the workload distribution condition. This statement is reinforced by the result shown in Fig. 3.11, where the imbalance obtained for each of the mesh partitions used in the previous tests is represented. The imbalance is evaluated as the difference between the number of interface cells of the most overloaded parallel process and the average of interface cells per process, divided by the latter. The values obtained agree with our statement. For example, looking at the $8\times8\times8$ case, the imbalance obtained using 1024 CPU-cores is $1.2\times$ the average number of interface cells, thus, the parallel process with maximum workload has to solve $(1\times) + (1.2\times) = 2.2\times$ the average. According to this, and assuming for simplicity an ideal load balance with 16 CPU-cores and an equal solution cost for all interface cells, the strong speedup obtained should be $64/2.2 = 29\times$, which is close to the observed strong speedup of $30\times$.

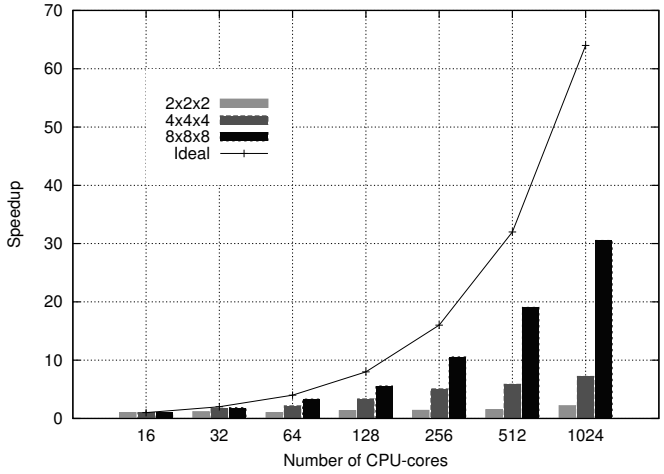


Figure 3.9: Speedup of the VOF solution using the DD strategy for the $2\times2\times2$, $4\times4\times4$ and $8\times8\times8$ interface configurations.

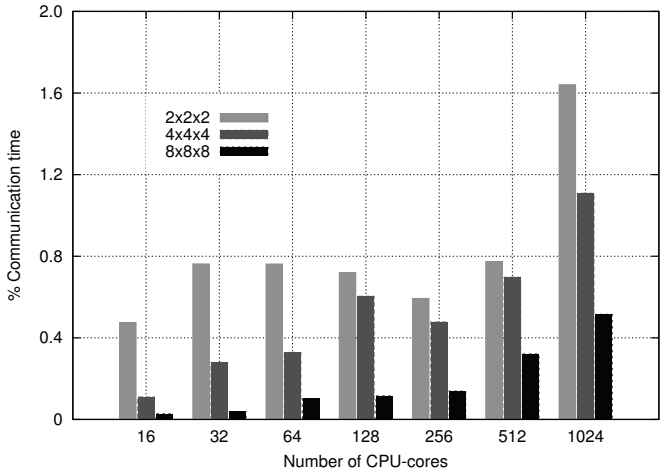


Figure 3.10: Percentage of the communication costs over the total cost of the VOF algorithm with the DD parallelization strategy.

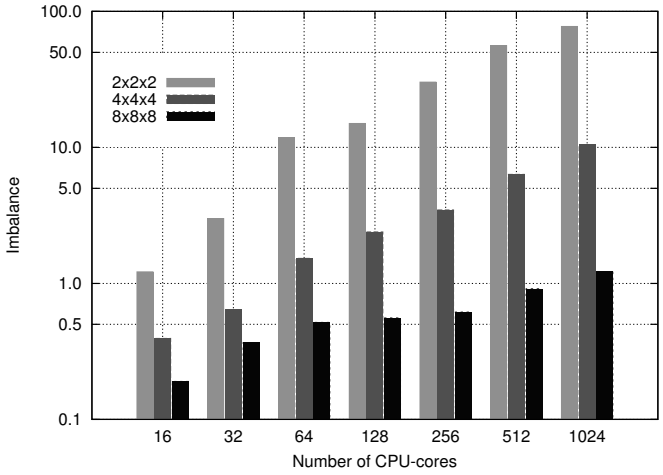


Figure 3.11: Imbalance obtained on each of the test cases studied. The imbalance is evaluated as the difference between the number of interface cells for the most overloaded parallel process and the average of interface cells per process, divided by the average.

The strong speedup is now analyzed for the new parallelization strategy, with identical test conditions to those set for the DD approach. The results, depicted in Fig 3.12, show a speedup qualitetively similar to that obtained with the DD algorithm, but quantitatively better. The improvement achieved is more noticeable the more imbalanced the case is. In the most extreme situation, using 1024 CPU-cores, the leap obtained in the PE by using the LB instead of the DD is from 48% to 63% for case $8\times8\times8$, from 11% to 50% for case $4\times4\times4$ and from 3% to 28% for case $2\times2\times2$; see Figs. 3.9 and 3.12. This improvement is also evident with lower numbers of CPU-cores. For example, with 128 CPU-cores, the leap is from 70% to 93%, from 42% to 83% and from 16% to 67% for the $8\times8\times8$, $4\times4\times4$, and $2\times2\times2$ interfaces, respectively. In conclusion, we observe that the LB strategy consistently outperforms the DD one.

In Fig. 3.13, as previously done for the DD strategy, we show the relative cost of the communications over the total cost of the VOF algorithm. These occur in the steps 2, 3 and 5 of Alg. 1. Again, the relative weight of the communications is proportional to the number of CPU-cores engaged and inversely proportional to the interface size, i.e., the workload. However, there is a major difference with respect to the results obtained for the DD parallelization: while the communications cost always represents less than 1.7% of the total time for the DD strategy, it reaches up to 50% with the LB one. In particular, in ascending order of number of spheres, with 1024 CPU-cores, communications represent 48%, 23% and 11% of the total compute time, respectively.

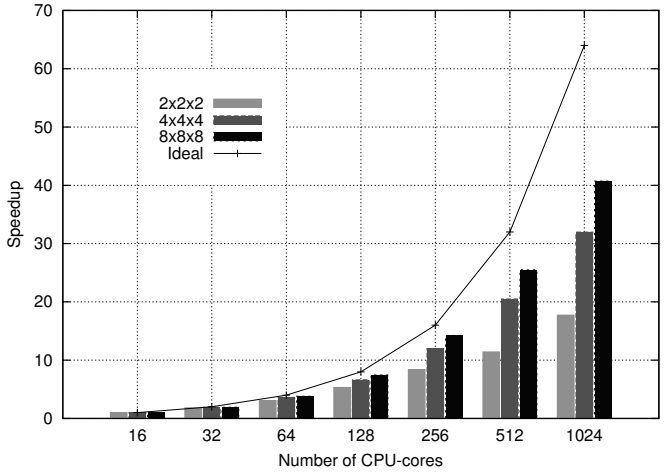


Figure 3.12: Speedup of the VOF solution using the LB strategy for the $2\times2\times2$, $4\times4\times4$ and $8\times8\times8$ interface configurations.

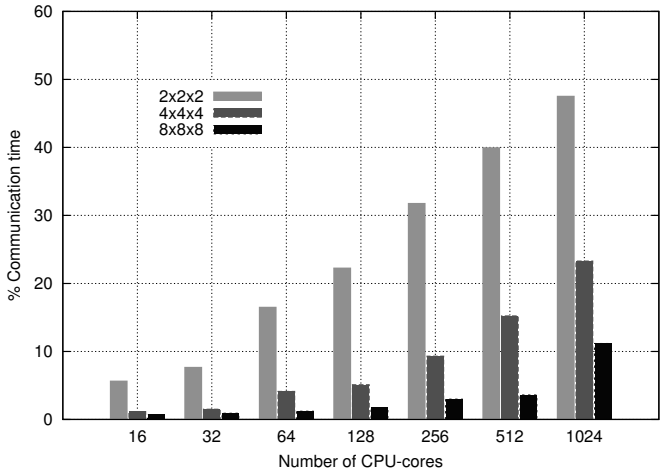


Figure 3.13: Percentage of the communication costs over the total cost of the VOF algorithm with the LB parallelization strategy.

Therefore, in contrast to what happens with the DD strategy, the speedup of the LB approach is limited by the cost of the communications required to move the data between parallel processes. Nevertheless, the degree of initial imbalance determines also the parallel performance, since the more imbalance there is, the larger is the amount of data that needs to be reassigned.

All tests presented up to this point refer to the strong speedup of the DD and LB strategies. Nevertheless, in order to contrast their real performance, we must compare their solution times for a VOF iteration, instead of their acceleration with respect to themselves. Accordingly, the ratio between both solution times is shown in Fig. 3.14 for the test cases studied in the previous figures. At the initial point, with 16 CPU-cores, all cases present a certain imbalance that favors the LB approach. This produces a speedup that ranges from 1.15 for the $8\times8\times8$ case up to 1.43 for the $2\times2\times2$ one. The rest of values derive from the differences already shown in the acceleration trends of both methods; see Figs. 3.9 and 3.12. Consequently, the initial speedup widens much more for the coarser interfaces. At the end, with 1024 CPU-cores, the speedup achieved by using our new approach ranges from $1.5\times$ for the $8\times8\times8$ configuration, up to $11.6\times$ for the coarser interface case.

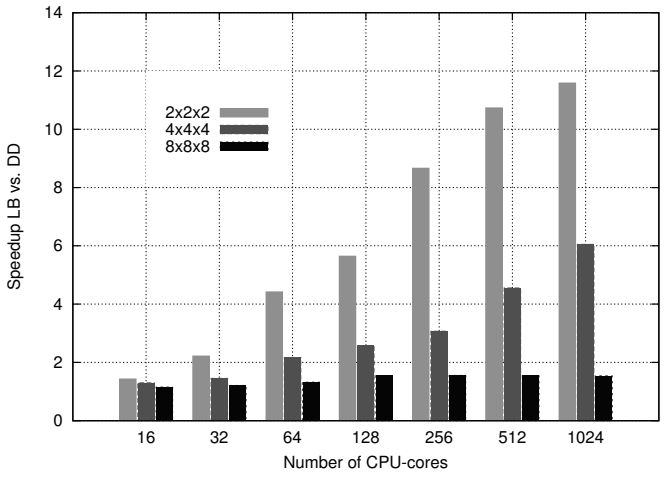


Figure 3.14: Speedup of the LB strategy versus the DD one for the VOF solution of the $2\times2\times2$, $4\times4\times4$ and $8\times8\times8$ interface configurations with different number of CPU-cores.

The next test is devoted to further analyze the effects of the initial interface distribution on the parallel performance. For this purpose, three new configurations are considered; see Fig. 3.15. On the one hand, the new configuration $4\times8\times8$ is

obtained by removing the spheres of the 8×8×8 interface that are located in one half of the domain. On the other hand, the configurations R-8×8×8 and R-4×8×8 are obtained by assigning random positions to the spheres of the respective grids, restricting them to one half of the domain for the case R-4×8×8. In Fig. 3.16, it is compared the VOF solution time for these three new configurations together with the 8×8×8 one, using 512 CPU-cores and both parallelization strategies. Analyzing first the effect of randomly placing the grids of spheres, i.e., comparing the first and second, and the third and forth columns of the DD and LB blocks in Fig. 3.16; it is clear that it produces a negative effect only for the DD strategy. The DD degradation was predictable, since setting the spheres positions randomly widens the imbalance. On the other hand, for the LB one, this additional imbalance should increase a little the data transfer requirements on the load balancing process. However, this effect is not perceptible in the solution time because, as shown in Fig 3.13, the weight of communications is relatively low in this case (∼5%). Note that we have carried out all the previous tests on grids of spheres uniformly distributed throughout the domain, the imbalance was therefore principally produced by the sparsity of the interface that leaves many processes with little or zero workload. As shown in the present test, random distributions may increase the imbalance and, thus, the performance of the LB with respect to the DD methodology.

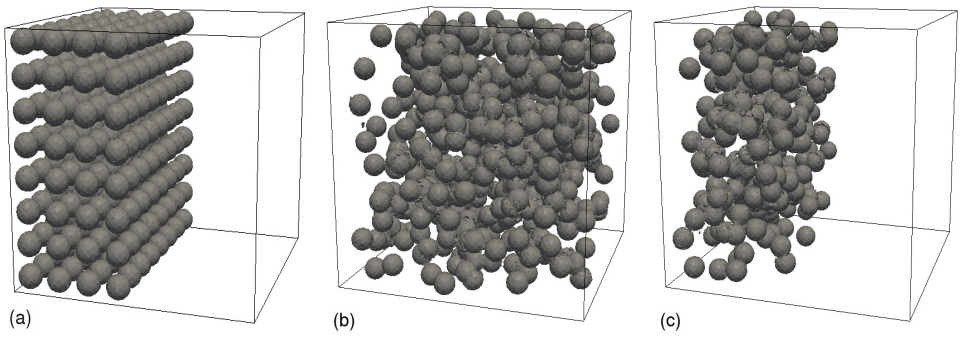


Figure 3.15: Three additional configurations of spheres: (a) 4×8×8, (b) R-8×8×8 and (c) R-4×8×8.

In the test shown in Fig. 3.16 we have also proposed the artificially generated situation in which half of the domain is empty. By doing this, the VOF workload is almost halved. Therefore, we would ideally expect that the solution time was halved as well. The real effect is observed by comparing columns first and third, and second and fourth of the DD and LB blocks, respectively. Using the DD approach, there is no time reduction, since the processors of the non-emptied half retain its workload. On

the other hand, with the LB strategy the workload is redistributed and the solution time is halved in both situations. This test shows the robustness of the new strategy in situations of imbalance in which the parallel performance of the DD is seriously degraded.

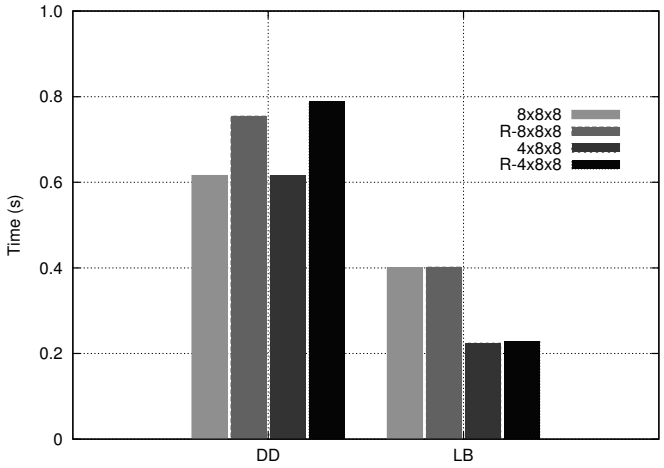


Figure 3.16: Comparative of the time required by the DD and LB strategies to solve the interface configurations $4\times8\times8$ and $8\times8\times8$ using 512 CPU-cores.

In the above tests different interface configurations have been considered, however, the underlying 3-D discretization has been kept constant. The effects of varying it are shown in Fig. 3.17. In particular, the strong speedup of the LB algorithm on the solution of the $4\times4\times4$ interface for two additional 3-D meshes of sizes 250K and 4000K, together with the results presented previously for the 1000K mesh, are depicted. When the 3-D discretization is varied, the number of interface cells also varies but in a lower degree, since the fluids interface is bidimensional. On the other hand, the distribution of the interface within the domain remains constant. However, this does not ensure that the partitions imbalance is the same, since the mesh partitioning is not based on geometrical criteria, but on topological criteria. The results obtained look as expected: the speedup improves by increasing the computing load, i.e., the mesh size. There are two main reasons for this: (1) the relative weight of the communications decreases; (2) the relative weight of any residual imbalance remaining after the load balancing process decreases as well. Particularly, the strong speedup results shown in Fig. 3.17 are very similar for all the 3-D meshes up to 128 CPU-cores. Indeed, in this range the communication overcosts remain rather low for all cases. As these costs grow and become more significant, differences appear on the

speedup. For instance, with 1024 CPU-cores, where the differences are the largest, the communications cost represents 48%, 24% and 18% of the solution time for the 250K, 1000K and 4000K mesh, respectively. Additionally, note that with 1024 CPU-cores, for the 250K mesh the ideal workload per CPU-core is minimal, only around 9 interface cells. As a consequence, a residual imbalance of only one interface cell on the new distribution degrades the imbalance by 10%. On the contrary, for the 4000K mesh the ideal load per CPU grows up to 63 interface cells, so this potential degradation is below 1.5%. In any case, note that the situation described in this test is practically the same one which occurs when the interface is varied by increasing or decreasing its size on a fixed 3-D grid. In fact, since only the interface cells are engaged on the VOF calculations, the size of the 3-D mesh is only important as it determines the size of the interface.

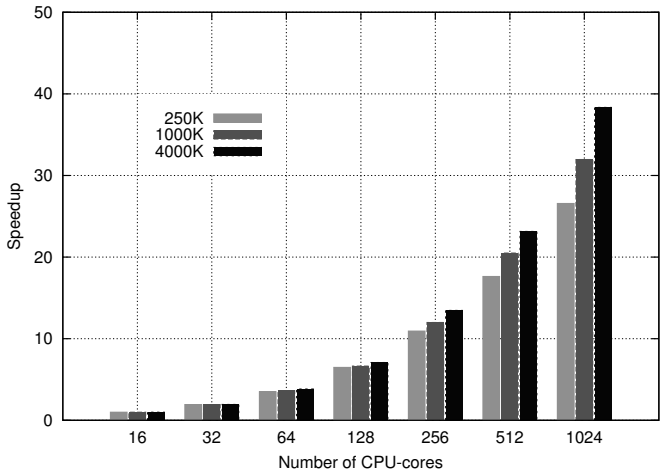


Figure 3.17: Speedup of the LB strategy on the solution of the $4\times4\times4$ interface configuration for three different 3-D meshes of sizes 250K, 1000K and 4000K.

Once our LB parallelization strategy has been extensively compared to the standard DD approach, the influence of the improvements introduced during the development of the algorithm are analyzed in the last test. In particular, these are two: (1) considering the overcost caused by the unpack process on the solution of the reassigned tasks (coefficient α); (2) assigning a weight to each task according to its relative cost. Neither of these two optimizations are necessary on the reconstruction phase because, on the one hand, the unpack operation cost is negligible with respect to the reconstruction calculations ($\alpha = 0$) and, on the other, the unitary tasks have all the same cost. Therefore, results are shown only for the advection phase. Indeed,

with our implementation the advection phase represents always around 85% of the solution time, so it essentially determines the overall performance. In Fig. 3.18 the reduction achieved on the advection solution time by the different optimizations is shown: (1) considering coefficient α (LB Alpha); (2) introducing weights (LB Weight); (3) considering both optimizations together (LB Optimal). The tests are executed on the 1000K mesh for the $4\times4\times4$ interface configuration, on the range of CPU-cores previously considered. At first sight, it is clear that the larger the CPU-cores engaged, the larger is the influence of the optimizations. Indeed, the optimizations are more relevant because the load balancing itself becomes more significant too. In particular, the benefit obtained by considering only the unpack overcosts (LB Alpha) is rather limited, not reaching 3%. In fact, there is an intrinsic limitation of 10%, since $\alpha = 0.1$. For VOF algorithms, geometric discretizations or computing systems that result in a larger α, this optimization could be much more important. On the other hand, using weights in order to optimize the balancing process (LB Weight) results in greater benefits that reach up to 19%. Finally, by setting both optimizations together (LB Optimal), the benefits reach up to 23%. Note also that both optimizations are mutually beneficial since, in general, the time reduction achieved by their interaction is superior to the sum of the reductions achieved separately.

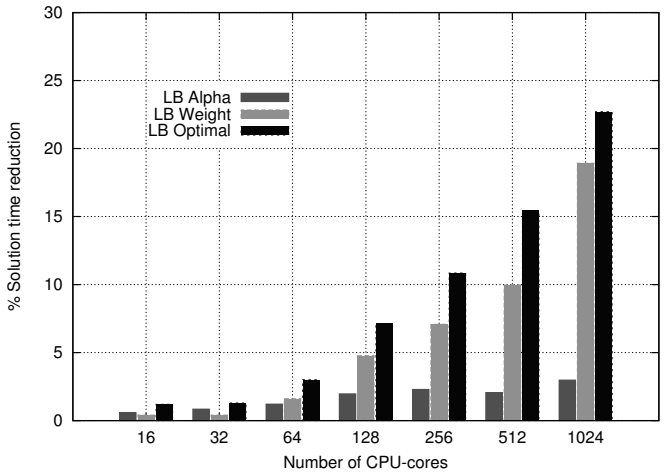


Figure 3.18: Reduction achieved on the advection solution time by different optimizations of the LB algorithm: (1) considering coefficient α (LB Alpha); (2) introducing weights (LB Weight); (3) considering both optimizations together (LB Optimal). The test case is the $4\times4\times4$ interface configuration on the 1000K mesh.

3.5 Conclusions

A new parallelization strategy for VOF methods has been presented and studied in detail. It has been developed with the aim of overcoming the workload imbalance obtained with the standard domain decomposition when the fluids interface is not homogeneously distributed throughout the domain. Basically, it consists in a dynamic load balancing process, complementary to the underlying domain decomposition, that reassigns tasks from processes with higher workload to processes with lower workload. This process is applied separately to the reconstruction and advection phases of the VOF algorithm. Since the initial domain decomposition is surpassed and the algorithm is applied to general unstructured discretizations, all the geometric and algebraic data required to perform any reassigned task need to be transmitted with it. In particular, communications are managed by means of buffers, and specific pack and unpack functions to, respectively, read and write data from them. To better achieve the desired load balance, two important issues need to be considered: the variable cost of the tasks being distributed and the overcost produced when a task is reassigned. An optimal workload balance leads to an NP-complete problem, for which a fast heuristic has been found giving a solution with 99% precision in few steps. Moreover, all the algorithms necessary to implement the new strategy have been described in detail.

An exhaustive analysis and comparison of the standard domain decomposition and our load balancing strategy has been performed. Several test cases, based on grids of spheres (representing the interface between fluids) distributed within a cubic domain, have been generated in order to measure the influence of the initial imbalance and of the problem size. These tests have been executed in the *MareNostrum-III* supercomputer of the *Barcelona Supercomputing Center*, engaging up to 1024 CPU-cores. It has been asserted that the efficiency of the DD strategy depends only on the load balancing or, equivalently, the interface distribution within the domain. Our LB strategy overcomes the imbalance, but the redistribution cost cancels part of the gains achieved from it. Anyway, when directly comparing both strategies, the result is that the larger the initial imbalance, the larger the speedup achieved by the LB algorithm respect to the DD one. We have observed speedups up to $\sim 12\times$ for the most ill-conditioned situations, but even in situations where the interface is almost spread throughout all the domain, the speedup achieved is $\sim 1.5\times$ in average.

With this scenario in mind, the new parallelization strategy presented may be a feasible option to be considered when solving multi-fluid flows by means of VOF methods. Moreover, our approach could be easily adapted to other interface-capturing methods, like the Level-Set, which suffer from a similar workload imbalance.

Acknowledgements

This work has been financially supported by the *Ministerio de Economía y Competitividad, Secretaría de Estado de Investigación, Desarrollo e Innovación*, Spain (ENE-2010-17801), a FPU Grant by the *Ministerio de Educación, Cultura y Deporte*, Spain (AP-2008-03843) and by *Termo Fluids S.L.*

The computations presented in this work have been carried out on the *IBM MareNostrum-III* supercomputer at the *Barcelona Supercomputing Center* (BSC), Spain (FI-2012-3-0021 and FI-2013-1-0024). The authors thankfully acknowledge this Institution.

We also thank the anonymous reviewers for their comments and remarks which helped to improve the quality of this work.

References

[1] C. W. Hirt, J. L. Cook, and T. D. Butler. A Lagrangian Method for Calculating the Dynamics of an Incompressible Fluid with Free Surface. *Journal of Computational Physics*, 5:103–124, 1970.

[2] C. W. Hirt, A. A. Amsden, and J. L. Cook. An Arbitrary Lagrangian-Eulerian Computing Method for All Flow Speeds. *Journal of Computational Physics*, 135:203–216, 1997.

[3] H. H. Hu, N. A. Patankar, and M. Y. Zhu. Direct Numerical Simulations of Fluid-Solid Systems Using the Arbitrary Lagrangian-Eulerian Technique. *Journal of Computational Physics*, 169:427–462, 2001.

[4] F. H. Harlow and J. E. Welch. Numerical Calculation of Time-Dependent Viscous Incompressible Flow of Fluid with Free Surface. *Physics of Fluids*, 8:2182–2189, 1965.

[5] C. S. Peskin. Numerical Analysis of Blood Flow in the Heart. *Journal of Computational Physics*, 25:220–252, 1977.

[6] G. Tryggvason, B. Bunner, A. Esmaeeli, D. Juric, N. Al-Rawashi, W. Tauber, J. Han, S. Nas, and Y. J. Jan. A Front-Tracking Method for the Computations of Multiphase Flow. *Journal of Computational Physics*, 169:708–759, 2001.

[7] C. W. Hirt and B. D. Nichols. Volume of fluid (VOF) Method for the Dynamics of Free Boundaries. *Journal of Computational Physics*, 39:201–225, 1981.

[8] 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*, 35:1011–1032, 2006.

[9] T. Marić, H. Marschall, and D. Bothe. voFoam - A Geometrical Volume of Fluid Algorithm on Arbitrary Unstructured Meshes with Local Dynamic Adaptive Mesh Refinement using OpenFOAM. *arXiv:1305.3417*, pages 1–30, 2013.

[10] S. Osher and J. Sethian. Fronts Propagating with Curvature Dependent Speed: Algorithms Based on Hamilton-Jacobi Formulations. *Journal of Computational Physics*, 79:12–49, 1988.

[11] E. Olsson and G. Kreiss. A Conservative Level Set Method for Two Phase Flow. *Journal of Computational Physics*, 210:225–246, 2005.

[12] N. Balcázar, L. Jofre, O. Lehmkuhl, J. Castro, and J. Rigola. A Finite-Volume/Level-Set Method for Simulating Two-Phase Flows on Unstructured Grids. *International Journal of Multiphase Flow*, 64:55–72, 2014.

[13] Y. Renardy. Effect of Startup Conditions on Drop Breakup under Shear with Inertia. *International Journal of Multiphase Flow*, 34:1185–1189, 2008.

[14] M. S. Annaland, N. G. Deen, and J. A. M. Kuipers. Numerical Simulation of Gas Bubbles Behaviour using a Three-Dimensional Volume of Fluid Method. *Chemical Engineering Science*, 60:2999–3011, 2005.

[15] K. M. T. Kleefsman, G. Fekkena, A. E. P. Veldman, B. Iwanowski, and B. Buchner. A Volume-of-Fluid Based Simulation Method for Wave Impact Problems. *Journal of Computational Physics*, 206:363–393, 2005.

[16] D. Fuster, A. Bagué, T. Boeck, L. Le Moyne, A. Leboissetier, S. Popinet, P. Ray, R. Scardovelli, and S. Zaleski. Simulation of Primary Atomization with an Octree Adaptive Mesh Refinement and VOF Method. *International Journal of Multiphase Flow*, 35:550–565, 2009.

[17] G. Tomar, D. Fuster, S. Zaleski, and S. Popinet. Multiscale Simulations of Primary Atomization. *Computers & Fluids*, 39:1864–1874, 2010.

[18] X. Chen, D. Ma, V. Yang, and S. Popinet. High-Fidelity Simulations of Impinging Jet Atomization. *Atomization and Sprays*, 23:1079–1101, 2013.

[19] P. R. Chapman and J. W. Jacobs. Experiments on the Three-Dimensional Incompressible Richtmyer-Meshkov Instability. *Physics of Fluids*, 18:074101, 2006.

[20] V. Le Chenadec and H. Pitsch. A 3D Unsplit Forward/Backward Volume-of-Fluid Approach and Coupling to the Level Set Method. *Journal of Computational Physics*, 233:10–33, 2013.

[21] S. P. MacLachlan, J. M. Tang, and C. Vuik. Fast and Robust Solvers for Pressure-Correction in Bubbly Flow Problems. *Journal of Computational Physics*, 227:9742–9761, 2008.

[22] R. Borrell, O. Lehmkuhl, F. X. Trias, and A. Oliva. Parallel Direct Poisson Solver for Discretizations with one Fourier Diagonalizable Direction. *Journal of Computational Physics*, 230:4723–4741, 2011.

[23] B. J. Araújo, J. C. F. Teixeira, A. M. Cunha, and C. P. T. Groth. Parallel Three-Dimensional Simulation of the Injection Molding Process. *International Journal for Numerical Methods in Fluids*, 59:801–815, 2009.

[24] M. Sussman. A Parallelized, Adaptive Algorithm for Multiphase Flows in General Geometries. *Computers & Structures*, 83:435–444, 2005.

[25] M. Sussman and E. G. Puckett. A Coupled Level Set and Volume-of-Fluid Method for Computing 3D and Axisymmetric Incompressible Two-Phase flows. *Journal of Computational Physics*, 162:301–337, 2000.

[26] 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*, 58:897–921, 2008.

[27] M. Herrmann. Detailed Numerical Simulations of the Primary Atomization of a Turbulent Liquid Jet in Crossflow. *Journal of Engineering for Gas Turbines and Power*, 132:061506–10, 2010.

[28] 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.

[29] D. L. Youngs. Time-Dependent Multi-Material Flow with Large Fluid Distortion. In *Numerical Methods for Fluid Dynamics*, pages 273–285. Academic Press, New York, 1982.

[30] A. Haselbacher and V. Vasilyev. Commutative Discrete Filtering on Unstructured Grids based on Least-Squares Techniques. *Journal of Computational Physics*, 187:197–211, 2003.

[31] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C++*. Cambridge University Press, 2002.

[32] J. López and J. Hernández. Analytical and Geometrical Tools for 3D Volume of Fluid Methods in General Grids. *Journal of Computational Physics*, 227:5939–5948, 2008.

[33] W. Wall, S. Genkinger, and E. Ramm. A Strong Coupling Partitioned Approach for Fluid-Structure Interaction with Free Surfaces. *Computers & Fluids*, 36:169–183, 2007.

[34] P. Liovic and D. Lakehal. Interface-Turbulence Interactions in Large-Scale Bubbling Processes. *International Journal of Heat and Fluid Flow*, 28:127–144, 2007.

[35] Z. Wang, J. Yang, B. Koo, and F. Stern. A Coupled Level Set and Volume-of-Fluid Method for Sharp Interface Simulation of Plunging Breaking Waves. *International Journal of Multiphase Flow*, 35:227–246, 2009.

[36] L. Jofre, O. Lehmkuhl, R. Borrell, J. Castro, and A. Oliva. Parallelization Study of a VOF/Navier-Stokes Model for 3D Unstructured Staggered Meshes. In *Proceedings of the Parallel CFD Conference*, pages 1–5, 2011.

[37] J. Y-T. Leung. *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*. CRC Press, 2004.

[38] O. Lehmkuhl, C. D. Pérez-Segarra, R. Borrell, M. Soria, and A. Oliva. TERMOFLUIDS: A New Parallel Unstructured CFD Code for the Simulation of Turbulent Industrial Problems on Low Cost PC Cluster. In *Proceedings of the Parallel CFD Conference*, pages 1–8, 2007.

[39] Barcelona Supercomputing Center. Webpage: http://www.bsc.es.

[40] G. Karypis and K. Vipin. A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs. *SIAM Journal on Scientific Computing*, 20:359–392, 1998.

4

Discretization of the Navier-Stokes equations on unstructured meshes

Main contents of this chapter have been published in:

L. Jofre, O. Lehmkuhl, J. Ventosa, F. X. Trias, and A. Oliva. Conservation Properties of Unstructured Finite-Volume Mesh Schemes for the Navier-Stokes Equations. *Numerical Heat Transfer, Part B*, 65:53–79, 2014.

Abstract. The continuity and Navier-Stokes equations describe fluid flow by conserving mass and momentum. There are two main mesh discretizations for the calculation of these equations, the collocated and staggered schemes. Collocated schemes locate the velocity field at the same grid points as the pressure one, while staggered discretizations locate variables at different points within the mesh. One of the most important characteristics of the discretization schemes, aside from accuracy, is their capacity to discretely conserve kinetic energy, specially when solving turbulent flow. Hence, this work analyzes the accuracy and conservation properties of two particular collocated and staggered mesh schemes by solving a Rankine vortex, an exact sinusoidal function and the turbulent flow over a circular cylinder at $Re = 3900$.

4.1 Introduction

The continuity and Navier-Stokes equations are a general model that describes fluid flow by conserving mass and momentum, the latter being derived from Newton's second law applied to a fluid. These set of equations cannot be solved analytically. Instead, one of the multiple approaches is to set up a discrete system of equations that can be solved with computers. One of the main difficulties when discretizing these equations is the location of velocity and pressure node points, since an inadequate arrangement may produce a checkerboard pressure solution caused by the decoupling of velocity and pressure fields. Over the years, two main mesh discretizations for the calculation of the discrete Navier-Stokes equations have been developed, the collocated and staggered schemes.

Collocated mesh schemes locate the velocity field at the same grid points as the pressure one, which can result in a checkerboard pressure problem as shown by Patankar [1]. In order to minimize this problem, Rhie and Chow [2] proposed a special interpolation to compute the velocity field at cell faces for curvilinear grids. Later, Davidson [3] and Marthur et al. [4] extended the methodology to unstructured meshes. All these strategies did not conserve kinetic energy, as analyzed by Morinishi et al. [5], who stated that collocated mesh methods contain a kinetic energy conservation error of the form $\mathcal{O}(\Delta t^m, \Delta h^n)$ due to the improper pressure gradient formulation. In recent years, the scheme's kinetic energy conservation has been improved on unstructured meshes: (1) using a least-squares procedure to calculate the pressure-gradient term that advances cell-centered velocities by Mahesh et al. [6], although making the formulation not stable enough for all kind of grids; (2) utilizing vectors that span the null space of the discrete pressure Laplacian to obtain a smooth pressure field by Shashank et al. [7], despite being only accomplished for Cartesian grids. Another approach has been presented by Felten and Lund [8] for curvilinear grids and recast in a slightly different manner to unstructured meshes by Lehmkuhl et al. [9,10], which proposes a special definition for the projected velocity face flux that exactly conserves mass, resulting in a kinetic energy conservation error of the form $\mathcal{O}(\Delta t^m, \Delta h^2)$.

On the other hand, a staggered mesh scheme is any numerical scheme where variables are located at different points within the mesh. While different staggering schemes are possible [11–13], this work is interested in the scheme presented for 2-D unstructured meshes by Perot [14] and analyzed on 3-D unstructured meshes by Zhang et al. [15], since it is a generalization to unstructured meshes of the one originally presented by Harlow and Welch [16]. This scheme locates pressure at cell centers and normal velocities at cell faces while not displaying spurious pressure modes, i.e., there is no red-black uncoupling of the pressure unknowns. As a counterpart, normal face velocities are discretized in time, thus, cell-centered velocities need to be interpolated from face normal values.

The main purposes of this work are to accurately formulate the collocated scheme utilized extensively by Lehmkuhl et al. [9, 10] and Rodríguez et al. [17, 18], extend Perot's [14] staggered discretization by studying a different cell-centered velocity interpolation, and to analyze the conservation properties and accuracy of both schemes. In this way, an improved knowledge of both schemes will be gained. Going forward, mesh discretizations will be chosen according to the main flow properties to be favored in complex problems regarding, for example, multiphase flow, combustion problems, or fluids with nonconstant physical properties. First, both discretization strategies are explained in detail in Sec. 4.2. Next, their conservation of mass, momentum and kinetic energy is studied in Sec. 4.3. Finally, different problems are solved in Sec. 4.4 to test their conservation properties and accuracy, such as a Rankine vortex, an exact sinusoidal function and the turbulent flow over a circular cylinder.

4.2 Discrete Navier-Stokes equations

The divergence form of the incompressible continuity and Navier-Stokes equations is

$$\nabla \cdot \mathbf{u} = 0, \tag{4.1}$$

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot (\mathbf{u}\mathbf{u}) = -\frac{1}{\rho}\nabla p + \nu \Delta \mathbf{u}, \tag{4.2}$$

where $\mathbf{u}$ is the velocity, p the pressure and ρ and ν the constant density and kinematic viscosity, respectively. The finite-volume spatial discretization of these equations on a general arbitrary mesh scheme, using discrete matrix operators, is written as

$$\mathbf{M}\mathbf{u} = \mathbf{0}, \tag{4.3}$$

$$\mathbf{\Omega}\frac{d\mathbf{u}}{dt} + \mathbf{C}(\mathbf{u})\mathbf{u} + \nu \mathbf{D}\mathbf{u} + \frac{1}{\rho}\mathbf{G}\mathbf{p} = \mathbf{0}, \tag{4.4}$$

where $\mathbf{u}$ and $\mathbf{p}$ are the vectors of velocities and pressures. The diagonal matrix $\mathbf{\Omega}$ describes the volume of cells, matrices $\mathbf{C}(\mathbf{u})$ and $\mathbf{D}$ are the convective and diffusive operators, and matrices $\mathbf{G}$ and $\mathbf{M}$ represent the gradient and divergence operators.

Discrete conservation properties are related to the symmetries of these matrices as studied in detail by Verstappen and Veldman [19]. Hence, kinetic energy is conserved if and only if the discrete convective operator is skew-symmetric, i.e., the transpose of the matrix is also its negative, $\mathbf{C}(\mathbf{u}) = -\mathbf{C}(\mathbf{u})^*$, and if the negative conjugate transpose of the discrete gradient operator is equal to the divergence operator, $\mathbf{M} = -\mathbf{G}^*$. On the other hand, since diffusive terms must be dissipative, the diffusive operator must be symmetric and positive-definite, i.e., the matrix is equal to its transpose $\mathbf{D} = \mathbf{D}^*$, and $\mathbf{z}^*\mathbf{D}\mathbf{z} > \mathbf{0}$ for all nonzero $\mathbf{z}$.

4.2.1 Collocated mesh scheme

The collocated mesh scheme calculates velocity and pressure fields at cell centers and needs particular interpolations and special velocity fluxes at faces, in order to minimize the kinetic energy error and conserve mass exactly, respectively.

The velocity-pressure coupling of the momentum equation, Eq. 4.2, is solved by means of a classical fractional step projection method along with a first-order explicit time advancement, written as

$$\mathbf{u}^{n+1} - \mathbf{u}^{p} = -\frac{\Delta t}{\rho}\nabla p^{n+1}, \tag{4.5}$$

$$\mathbf{u}^{p} = \mathbf{u}^{n} - \Delta t\left[\nabla\cdot(\mathbf{u}^{n}\mathbf{u}^{n}) - \nu\Delta\mathbf{u}^{n}\right], \tag{4.6}$$

where superscript n refers to time instant, $\mathbf{u}^{p}$ is the predictor velocity, and Δt is the time step.

First, the predictor velocity is discretized by integrating Eq. 4.6 over a cell c and applying the divergence theorem to its bordering faces, $f \in F(c)$, giving

$$\mathbf{u}_c^{p} = \mathbf{u}_c^{n} - \frac{\Delta t}{V_c}\left[\sum_{f\in F(c)} \boldsymbol{\phi}_f^{n}\hat{U}_f^{n}A_f - \nu\sum_{f\in F(c)}(\mathbf{u}_{nb}^{n} - \mathbf{u}_c^{n})\frac{A_f}{\delta d_f}\right], \tag{4.7}$$

where V_c is the volume of cell c, $\boldsymbol{\phi}_f$ is the convected face velocity, $\hat{U}_f$ is the normal face velocity, $\hat{\mathbf{n}}_f$ is the outward-unit face normal, A_f is the face surface, subscripts c and nb refer to the cell itself and the face-neighbor one, and length δd_f is the normal-projected distance between the centroids of cells c and nb; see Fig. 4.1.

Next, taking the divergence of Eq. 4.5, applying the incompressibility condition, Eq. 4.1, and discretizing over cell c, yields a discrete Poisson equation

$$\sum_{f\in F(c)} \hat{U}_f^{p}A_f = \frac{\Delta t}{\rho}\sum_{f\in F(c)}(p_{nb}^{n+1} - p_c^{n+1})\frac{A_f}{\delta d_f}, \tag{4.8}$$

which solves the pressure field. When the solution of p^{n+1} is obtained, $\mathbf{u}^{n+1}$ results from discretizing Eq. 4.5 over cell c as

$$\mathbf{u}_c^{n+1} = \mathbf{u}_c^{p} - \frac{\Delta t}{\rho V_c}\sum_{f\in F(c)} p_f^{n+1}\hat{\mathbf{n}}_f A_f, \tag{4.9}$$

where p_f is the pressure interpolated to face f.

Notice that no specific interpolations for $\boldsymbol{\phi}_f^{n}$, $\hat{U}_f^{p}$ and p_f^{n+1} have been explained yet. Therefore, in order to fulfill the skew-symmetric conservation requirement of the

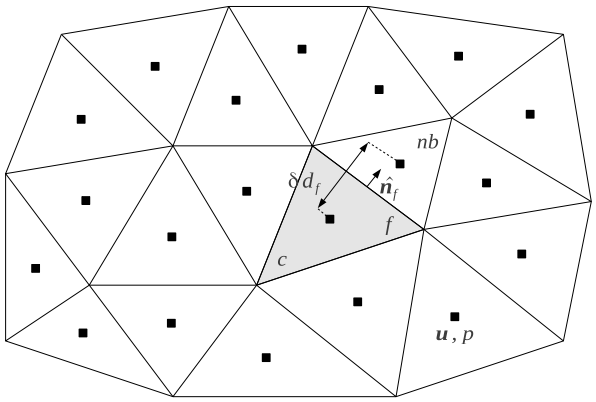


Figure 4.1: Variable arrangement and notation for the collocated scheme on a 2-D unstructured mesh. The schematic representation shows the collocated position of velocity, $\mathbf{u}$, and pressure, p. The cell c where the discretization is analyzed is shown in gray, with an example of face f and its corresponding neighbor cell nb, outward-unit normal $\hat{\mathbf{n}}_f$ and centroids' distance, δd_f.

discrete convective operator, the convected face velocity is evaluated as $\boldsymbol{\phi}_f^n = \frac{1}{2}(\mathbf{u}_c^n + \mathbf{u}_{nb}^n)$ [19]. On the other hand, the normal face predictor velocity and face pressure are calculated as $\hat{U}_f^p = \frac{1}{2}(\mathbf{u}_c^p + \mathbf{u}_{nb}^p)\cdot \hat{\mathbf{n}}_f$ and $p_f^{n+1} = \frac{1}{2}(p_c^{n+1} + p_{nb}^{n+1})$, minimizing the kinetic energy conservation error as analyzed by Felten and Lund [8].

Finally, the evaluation of the normal face velocity, $\hat{U}_f^{n+1}$, needs to be studied in detail in order to exactly conserve mass. Thus, taking again the divergence of Eq. 4.5 and discretizing over cell c gives

$$\sum_{f\in F(c)} \hat{U}_f^{n+1} A_f - \sum_{f\in F(c)} \hat{U}_f^{p} A_f = -\frac{\Delta t}{\rho} \sum_{f\in F(c)} (p_{nb}^{n+1} - p_c^{n+1}) \frac{A_f}{\delta d_f}, \tag{4.10}$$

which can be arranged in the following form

$$\sum_{f\in F(c)} \left[\hat{U}_f^{n+1} A_f - \hat{U}_f^{p} A_f + \frac{\Delta t}{\rho} (p_{nb}^{n+1} - p_c^{n+1}) \frac{A_f}{\delta d_f} \right] = 0. \tag{4.11}$$

Next, imposing a more restrictive but easier condition that asks each face to equal zero, the following equation is obtained

$$\hat{U}_f^{n+1} = \hat{U}_f^{p} - \frac{\Delta t}{\rho} \frac{(p_{nb}^{n+1} - p_c^{n+1})}{\delta d_f}. \tag{4.12}$$

Then, if the predictor normal face velocity is evaluated as the semi-sum $\hat{U}_f^p = \frac{1}{2}(\mathbf{u}_c^p + \mathbf{u}_{nb}^p)\cdot\hat{\mathbf{n}}_f$ and $\mathbf{u}^p$ is substituted using Eq. 4.9, Eq. 4.12 is rewritten as

$$\begin{aligned}\hat{U}_f^{n+1} =& \frac{1}{2}(\mathbf{u}_c^{n+1} + \mathbf{u}_{nb}^{n+1})\cdot\hat{\mathbf{n}}_f - \frac{\Delta t}{\rho}\left[\frac{(p_{nb}^{n+1} - p_c^{n+1})}{\delta d_f}\right] \\ &+ \frac{\Delta t}{\rho}\left[\frac{1}{2}\left[\frac{1}{V_c}\sum_{f\in F(c)} p_f^{n+1}\hat{\mathbf{n}}_f A_f + \frac{1}{V_{nb}}\sum_{f\in F(nb)} p_f^{n+1}\hat{\mathbf{n}}_f A_f\right]\right]\cdot\hat{\mathbf{n}}_f,\end{aligned} \tag{4.13}$$

which is similar to the mass-conserving normal face velocity proposed by Felten and Lund [8].

4.2.2 Staggered mesh scheme

The staggered mesh scheme stores pressure and other scalar quantities at cell centers while normal velocities are distributed to cell faces. Each face stores only the normal component of velocity, therefore, the cell-centered velocity vector has to be recovered from face normal values. This recovery or interpolation of velocity vector from face normal quantities is not unique, and it is a defining characteristic of each staggered mesh scheme, leading to different properties for the solution. This work focuses on the staggered mesh discretization developed for 2-D unstructured meshes by Perot [14] and extended to 3-D meshes by Zhang et al. [15].

In order to develop the staggered discretization, some preliminary remarks are needed. First, face-centered control volumes are defined for each face f as $V_f = (W_f^a + W_f^b)A_f$, where W_f is the distance between the face and each neighboring cell circumcenter and A_f is the surface of face f; see Fig. 4.2. Second, convective and diffusive terms are calculated at cell centers as non-volumetric quantities and distance-interpolated to faces using W_f.

Thus, integrating Eq. 4.5 and 4.6 over face f control volume and taking a dot product with the face normal vector, $\mathbf{n}_f$, results in the discrete staggered form of the fractional step projection method

$$U_f^{n+1} = U_f^p - \frac{\Delta t}{\rho V_f}(p_b^{n+1} - p_a^{n+1})A_f, \tag{4.14}$$

$$U_f^p = U_f^n - \frac{\Delta t}{V_f}\left[W_f^a(\mathbf{c}_a - \mathbf{d}_a) + W_f^b(\mathbf{c}_b - \mathbf{d}_b)\right]\cdot\mathbf{n}_f A_f. \tag{4.15}$$

Subscripts a and b refer to the two cells adjacent to face f, and $\mathbf{c}$ and $\mathbf{d}$ are the non-volumetric cell-centered discretizations of the convective and diffusive terms

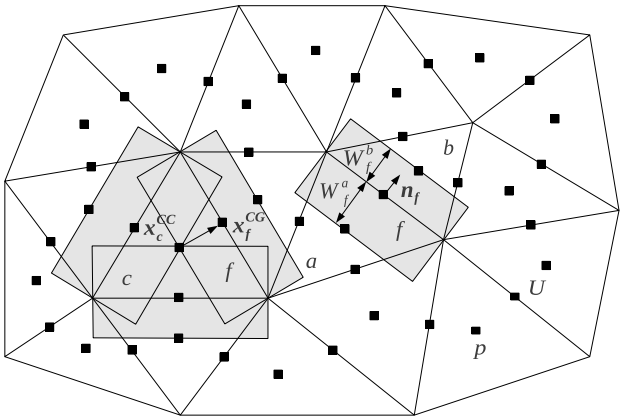


Figure 4.2: Variable arrangement and notation for the staggered scheme on a 2-D unstructured mesh. The schematic representation shows the staggered position of the normal velocity, U, and the cell-centered location of pressure, p. Face f and its neighboring cells a and b, where the cell-to-face operator is explained, are shown together with distances W_f^a and W_f^b. On the other hand, the face-to-cell operator is shown by representing cell c and an example of face f where the interpolation is taken using distance $\mathbf{x}_f^{CG} - \mathbf{x}_c^{CC}$.

evaluated for each cell c as

$$\mathbf{c}_c = \frac{1}{V_c} \sum_{f \in F(c)} \boldsymbol{\phi}_f^n \hat{U}_f^n A_f, \quad \mathbf{d}_c = \frac{1}{V_c} \sum_{f \in F(c)} \nu(\mathbf{u}_{nb}^n - \mathbf{u}_c^n) \frac{A_f}{\delta d_f}, \tag{4.16}$$

where the convected face velocity, $\boldsymbol{\phi}_f$, is evaluated as previously defined for the collocated formulation and length δd_f is once again the distance between cell nodes.

Next, taking the divergence of Eq. 4.5, using the incompressibility condition, and discretizing over cell c gives the discrete Poisson equation already presented, Eq. 4.8, but in this case no interpolation is needed since the normal face predictor velocity, U_f^p, is given by Eq. 4.15 and δd_f is now the distance between cell circumcenters. When the solution of p^{n+1} is calculated, Eq. 4.14 is used to obtain the normal face velocities at instant $n+1$, U_f^{n+1}.

Finally, the staggered mesh scheme discretizes normal face velocities in time, then, cell-centered velocities need to be interpolated from face normal values. In this work, two different interpolations will be analyzed: the one presented by Perot [14] (staggered *a*) and a different one proposed by the authors (staggered *b*).

On the one hand, Perot (staggered *a*) proposes to apply Gauss' divergence theorem for cell c to the product of position $\mathbf{r}$ and velocity $\mathbf{u}$, giving

$$\int_{\Omega_c} \mathbf{u}\, dV + \int_{\Omega_c} \mathbf{r}(\nabla \cdot \mathbf{u})\, dV = \sum_{f \in F(c)} \int_{\partial\Omega_f} \hat{U}\mathbf{r}\, dA, \tag{4.17}$$

where $\mathbf{r} = \mathbf{x} - \mathbf{x}_0$ is the position vector from the cell circumcenter and $\hat{U}$ is the outward normal face velocity. Then, if a first-order approximation of the velocity field (constant $\mathbf{u}$) is assumed, Eq. 4.17 is rewritten as

$$\mathbf{u}_c = \frac{1}{V_c} \sum_{f \in F(c)} \hat{U}_f \mathbf{r}_f^c A_f, \tag{4.18}$$

and $\mathbf{r}_f^c = \mathbf{x}_f^{CG} - \mathbf{x}_c^{CC}$ is the vector from cell circumcenter, $\mathbf{x}_c^{CC}$, to face centroid, $\mathbf{x}_f^{CG}$.

On the other hand, this study presents a different approach for the reconstruction of cell-centered velocities (staggered *b*) based on a least-squares procedure [20], which resembles to the work of Vidovic [21]. In this way, the cell-centered velocity is thought to be approximated in the vicinity of the cell centroid, $\mathbf{r}_0$, as a polynomial of the form

$$\mathbf{u}_c(\mathbf{r}) = \mathbf{a} + \mathbf{b}x + \mathbf{c}y + \mathbf{d}z, \tag{4.19}$$

where $\mathbf{r}$ is the position vector, relative to point $\mathbf{r}_0$, of a point where $\mathbf{u}_c$ is to be reconstructed, and $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$ and $\mathbf{d}$ are the unknowns to be determined. Hence, for each cell c a system of equations is created by imposing at its surrounding cell faces, $f \in S(c)$, that the scalar product of velocity, $\mathbf{u}_c(\mathbf{r})$, and outward-unit face normal, $\hat{\mathbf{n}}_f$, equals the normal face velocity, $\hat{U}_f$, written as

$$\mathbf{u}_c(\mathbf{r}) \cdot \hat{\mathbf{n}}_f = \hat{U}_f, \tag{4.20}$$

which sets up the following linear system of equations

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{4.21}$$

where

$$\mathbf{A} = \tag{4.22}$$
$$\begin{bmatrix} \hat{n}_{x,0} & \hat{n}_{y,0} & \hat{n}_{z,0} & \hat{n}_{x,0}x_0 & \hat{n}_{x,0}y_0 & \hat{n}_{x,0}z_0 & \hat{n}_{y,0}x_0 & \hat{n}_{y,0}y_0 & \hat{n}_{y,0}z_0 & \hat{n}_{z,0}x_0 & \hat{n}_{z,0}y_0 & \hat{n}_{z,0}z_0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \hat{n}_{x,f} & \hat{n}_{y,f} & \hat{n}_{z,f} & \hat{n}_{x,f}x_f & \hat{n}_{x,f}y_f & \hat{n}_{x,f}z_f & \hat{n}_{y,f}x_f & \hat{n}_{y,f}y_f & \hat{n}_{y,f}z_f & \hat{n}_{z,f}x_f & \hat{n}_{z,f}y_f & \hat{n}_{z,f}z_f \end{bmatrix},$$

$$\mathbf{x} = [a_0 \quad a_1 \quad a_2 \quad b_0 \quad b_1 \quad b_2 \quad c_0 \quad c_1 \quad c_2 \quad d_0 \quad d_1 \quad d_2]^T, \tag{4.23}$$

$$\mathbf{b} = [\hat{U}_0 \quad \dots \quad \hat{U}_f]^T. \tag{4.24}$$

The resulting system is overdetermined since for each cell there are more surrounding faces than polynomial coefficients, $\mathbf{A}_{f\times 12}\mathbf{x}_{12\times 1} = \mathbf{b}_{f\times 1}$. In this way, the system is modified by applying a least-squares procedure which multiplies both sides of the equation by the transpose $\mathbf{A}^T$, giving a standard square system of linear equations, $(\mathbf{A}^T\mathbf{A})\mathbf{x} = \mathbf{A}^T\mathbf{b}$. Finally, the solution of the square system can be obtained by multiplying both sides by $(\mathbf{A}^T\mathbf{A})^{-1}$, resulting in the following expression

$$\mathbf{x} = [(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T]\mathbf{b}. \tag{4.25}$$

Notice that the matrix product $(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T$ needs to be calculated just once, since it deals only with geometric quantities. Thus, the solution of the linear system is manageable, as it is comprised of just a matrix-vector product.

4.3 Conservation properties

Conservation of mass and momentum is intrinsic to the continuity and Navier-Stokes equations, since they are derived specifically for the conservation of these particular quantities, thus, a suitable discretization has to conserve them. On the other hand, secondary conservation involves the conservation of derived quantities, such as kinetic energy, entropy and vorticity, which are not directly unknowns of the numerical system and, hence, cannot be directly imposed during the construction of numerical methods, but are of great importance for the physics of problems.

4.3.1 Mass conservation

Global conservation of mass invokes the integral of Eq. 4.1 over the whole domain, Ω. Thus, if the domain integral is transformed to a summation of integrals for each control volume that form the domain, $c \in \Omega$, the following expression is obtained

$$\int_\Omega \nabla\cdot\mathbf{u}\, dV = \sum_{c\in\Omega}\int_{\Omega_c} \nabla\cdot\mathbf{u}\, dV = \sum_{c\in\Omega}\sum_{f\in F(c)} \hat{U}_f A_f. \tag{4.26}$$

In the collocated case a special definition for the normal face velocity, Eq. 4.13, has been developed in order to exactly conserve mass for each cell c. On the other hand, for the staggered case no interpolation of the normal face velocity is needed, since it is calculated at cell faces by definition of the scheme. Hence, in both cases the mass is locally conserved, expressed for each cell c as

$$\int_{\Omega_c} \nabla\cdot\mathbf{u}\, dV = \int_{\partial\Omega_c} \mathbf{u}\cdot\hat{\mathbf{n}}\, dS = \sum_{f\in F(c)} \hat{U}_f A_f = 0. \tag{4.27}$$

Then, global mass conservation, Eq. 4.26, equals zero, since it is a summation of locally mass-conserving quantities.

4.3.2 Momentum conservation

The conservation of momentum is a straightforward consequence of writing the equations in divergence form. However, a proof of conservation of momentum may be natural for collocated schemes but not obvious for staggered ones on unstructured meshes. The inherent difficulty is due to the fact that the velocity vector is not a primary variable for staggered schemes.

Collocated momentum conservation

Total conservation of momentum is obtained by integrating Eq. 4.2 over the entire domain, which is transformed to a summation of integrals for each control volume that form the domain and converted to surface integrals by applying the divergence theorem, as previously done for mass conservation, giving

$$\sum_{c\in\Omega}\frac{d\mathbf{u}_c}{dt}V_c+\sum_{c\in\Omega}\sum_{f\in F(c)}\boldsymbol{\phi}_f\hat{U}_f A_f \tag{4.28}$$
$$=-\frac{1}{\rho}\sum_{c\in\Omega}\sum_{f\in F(c)}p_f\hat{\mathbf{n}}_f A_f+\nu\sum_{c\in\Omega}\sum_{f\in F(c)}(\mathbf{u}_{nb}-\mathbf{u}_c)\frac{A_f}{\delta d_f}.$$

Notice that $\hat{U}_f$, $\hat{\mathbf{n}}_f$ and $(\mathbf{u}_{nb}-\mathbf{u}_c)$ are quantities that present equal values but with different sign when evaluating them at a face f from two adjacent interior cells. In this way, interior fluxes cancel out and Eq. 4.28 is evaluated as the summation over boundary faces, $f\in F(\partial\Omega)$, written as

$$\sum_{c\in\Omega}\frac{d\mathbf{u}_c}{dt}V_c+\sum_{f\in F(\partial\Omega)}\boldsymbol{\phi}_f\hat{U}_f A_f \tag{4.29}$$
$$=-\frac{1}{\rho}\sum_{f\in F(\partial\Omega)}p_f\hat{\mathbf{n}}_f A_f+\nu\sum_{f\in F(\partial\Omega)}(\mathbf{u}_f-\mathbf{u}_a)\frac{A_f}{\delta d_f},$$

which is a proof of momentum conservation for collocated meshes since it states that the change in momentum is due only to the fluxes through the boundary of the domain.

Staggered momentum conservation

The primary quantity in staggered mesh schemes is the normal face velocity. Thus, integrating Eq. 4.2 over face f control volume, as explained in detail in Sec. 4.2.2, and

taking a dot product with the face normal vector, $\mathbf{n}_f$, gives the discretized momentum equation for the normal face velocity, U_f, written as

$$(W_f^a + W_f^b)A_f \frac{dU_f}{dt} + (W_f^a \mathbf{c}_a + W_f^b \mathbf{c}_b)A_f \cdot \mathbf{n}_f \tag{4.30}$$
$$= -\frac{1}{\rho}(p_b - p_a)A_f + (W_f^a \mathbf{d}_a + W_f^b \mathbf{d}_b)A_f \cdot \mathbf{n}_f.$$

Discrete staggered conservation of momentum is shown by multipliying Eq. 4.30 by the face normal vector, $\mathbf{n}_f$, and summing over all faces of the domain, $f \in F(\Omega)$, giving the following equation

$$\sum_{f\in F(\Omega)} (W_f^a + W_f^b)A_f \frac{dU_f}{dt}\mathbf{n}_f + \sum_{f\in F(\Omega)} (W_f^a \mathbf{c}_a + W_f^b \mathbf{c}_b)A_f \cdot \mathbf{n}_f \mathbf{n}_f \tag{4.31}$$
$$= -\frac{1}{\rho}\sum_{f\in F(\Omega)} (p_b - p_a)A_f \mathbf{n}_f + \sum_{f\in F(\Omega)} (W_f^a \mathbf{d}_a + W_f^b \mathbf{d}_b)A_f \cdot \mathbf{n}_f \mathbf{n}_f.$$

Then, the goal is to recast this equation as an equation for cell velocity.

First, the sum over faces of the time derivative term in Eq. 4.31 needs to be recast as a summation over cells. There are two situations, depending on which cell-centered velocity reconstruction is chosen: staggered a or b. On the one hand, if Perot's cell velocity interpolation is considered, the following analysis is developed

$$\sum_{f\in F(\Omega)} (W_f^a + W_f^b)A_f \frac{dU_f}{dt}\mathbf{n}_f = \frac{d}{dt}\left[\sum_{f\in F(\Omega)} (\mathbf{r}_f^a - \mathbf{r}_f^b)A_f U_f\right] \tag{4.32}$$
$$= \frac{d}{dt}\left[\sum_{c\in\Omega}\left[\frac{1}{V_c}\sum_{f\in F(c)} \hat{U}_f \mathbf{r}_f^c A_f\right] V_c\right] = \sum_{c\in\Omega} \frac{d\mathbf{u}_c}{dt} V_c,$$

where the first equality is true since for each face f the following expression stands

$$(W_f^a + W_f^b)\mathbf{n}_f = \mathbf{x}_b^{CC} - \mathbf{x}_a^{CC} \tag{4.33}$$
$$= (\mathbf{x}_f^{CG} - \mathbf{x}_a^{CC}) - (\mathbf{x}_f^{CG} - \mathbf{x}_b^{CC}) = \mathbf{r}_f^a - \mathbf{r}_f^b,$$

while the second equality is the transformation from face to cell summation, accounting that $\hat{U}_f^b = -\hat{U}_f^a$, and the third one follows from Eq. 4.18. On the other hand, if the

staggered b cell velocity reconstruction is adopted, the formulation writes as

$$\sum_{f\in F(\Omega)}(W_f^a+W_f^b)A_f\frac{dU_f}{dt}\mathbf{n}_f=\sum_{f\in F(\Omega)}(\mathbf{r}_f^a-\mathbf{r}_f^b)A_f\frac{d(\mathbf{u}_c\cdot\mathbf{n}_f)}{dt} \tag{4.34}$$
$$=\sum_{c\in\Omega}\frac{d\mathbf{u}_c}{dt}\left[\sum_{f\in F(c)}\mathbf{r}_f^c\cdot\hat{\mathbf{n}}_f A_f\right]=\sum_{c\in\Omega}\frac{d\mathbf{u}_c}{dt}V_c,$$

where Eqs. 4.20 and 4.33 are used in the first transformation, the second one applies the equality $\hat{U}_f^b=-\hat{U}_f^a$ and takes out of the face summation the velocity derivative, and finally, the third one identifies the face summation as the volume of cell c.

Second, sums over all faces of the domain for convective and diffusive terms in Eq. 4.31 can be recast as summations over boundary faces, expressed as

$$\sum_{f\in F(\Omega)}(W_f^a\mathbf{c}_a+W_f^b\mathbf{c}_b)A_f\cdot\mathbf{n}_f\mathbf{n}_f=\sum_{c\in\Omega}\mathbf{c}_c\cdot\left[\sum_{f\in F(c)}\mathbf{n}_f\mathbf{n}_f W_f^c A_f\right] \tag{4.35}$$
$$=\sum_{c\in\Omega}\mathbf{c}_c\cdot\mathbf{I}V_c=\sum_{c\in\Omega}\mathbf{c}_c V_c=\sum_{c\in\Omega}\sum_{f\in F(c)}\boldsymbol{\phi}_f\hat{U}_f A_f=\sum_{f\in F(\partial\Omega)}\boldsymbol{\phi}_f\hat{U}_f A_f,$$

$$\sum_{f\in F(\Omega)}(W_f^a\mathbf{d}_a+W_f^b\mathbf{d}_b)A_f\cdot\mathbf{n}_f\mathbf{n}_f=\sum_{c\in\Omega}\mathbf{d}_c\cdot\left[\sum_{f\in F(c)}\mathbf{n}_f\mathbf{n}_f W_f^c A_f\right]=\sum_{c\in\Omega}\mathbf{d}_c\cdot\mathbf{I}V_c$$
$$=\sum_{c\in\Omega}\mathbf{d}_c V_c=\sum_{c\in\Omega}\sum_{f\in F(c)}\nu(\mathbf{u}_{nb}-\mathbf{u}_c)\frac{A_f}{\delta d_f}=\nu\sum_{f\in F(\partial\Omega)}(\mathbf{u}_f-\mathbf{u}_a)\frac{A_f}{\delta d_f}, \tag{4.36}$$

where the term in brackets is a known geometric result from the divergence theorem and is equal to the identity tensor multiplied by the cell volume, $\mathbf{I}V_c$. Terms $\mathbf{c}_c$ and $\mathbf{d}_c$ are expanded using Eq. 4.16 and interior fluxes exactly cancel out, leaving just fluxes through the boundary faces.

Third, the pressure term in Eq. 4.31 can be straightforward rearranged as

$$\sum_{f\in F(\Omega)}(p_b-p_a)A_f\mathbf{n}_f=-\sum_{c\in\Omega}p_c\sum_{f\in F(c)}\hat{\mathbf{n}}_f A_f \tag{4.37}$$
$$+\sum_{f\in F(\partial\Omega)}p_f\hat{\mathbf{n}}_f A_f=\sum_{f\in F(\partial\Omega)}p_f\hat{\mathbf{n}}_f A_f.$$

In summary, it is shown that Eq. 4.31 can be recast to Eq. 4.29 by using Eqs. 4.32 to Eq. 4.37. Hence, the momentum conservation for the staggered schemes is proved, since it states that the change in momentum is due only to the fluxes through the boundary of the domain, as in the collocated scheme case.

4.3.3 Kinetic energy conservation

The conservation of kinetic energy is the most important property when solving turbulent flows. In this type of flow, the energy is convected from the main flow into the large eddies, and from them into the next smaller ones, and so on until it is dissipated in the smallest eddies found. Then, if no external sources are present, the rate of change of total energy is just determined by dissipation. Thus, discretization strategies with excessive numerical dissipation can alter the physics of the problem in a very important proportion.

The transport equation for kinetic energy is derived from the momentum equation, Eq. 4.2, by taking the velocity dot product and assuming incompressible fluid. In this way, the specific kinetic energy $\frac{1}{2}\mathbf{u}\cdot\mathbf{u}$ can be shown to obey

$$\frac{\partial(\frac{1}{2}\mathbf{u}\cdot\mathbf{u})}{\partial t}+\nabla\cdot[\mathbf{u}(\frac{1}{2}\mathbf{u}\cdot\mathbf{u})]=-\frac{1}{\rho}\nabla\cdot(p\mathbf{u})+\nu\nabla\cdot(\mathbf{u}\times\boldsymbol{\omega})-\nu\boldsymbol{\omega}\cdot\boldsymbol{\omega}, \tag{4.38}$$

where $\boldsymbol{\omega}=\nabla\times\mathbf{u}$ is the vorticity. The important characteristic of this equation is that it is conservative except for the negative definite sink term involving the product of viscosity and enstrophy, $\nu\boldsymbol{\omega}\cdot\boldsymbol{\omega}$. In the absence of external forces and viscosity, the kinetic energy is simply redistributed but not created or destroyed. Similarly, discrete systems will be kinetic energy conservative if convective and pressure terms in the discrete kinetic energy equation are shown to be conservative [22].

Collocated kinetic energy conservation

In order to investigate the collocated conservation of kinetic energy, the momentum equation, Eq. 4.2, is discretized over the whole domain and multiplied by the velocity vector, $\mathbf{u}$. Then, the resulting equation can be transformed to a summation of surface integrals for each cell c, written as

$$\begin{aligned}\sum_{c\in\Omega}\mathbf{u}_c\cdot\frac{d\mathbf{u}_c}{dt}V_c+\sum_{c\in\Omega}\mathbf{u}_c\cdot\sum_{f\in F(c)}\boldsymbol{\phi}_f\hat{U}_f A_f \\ =-\frac{1}{\rho}\sum_{c\in\Omega}\mathbf{u}_c\cdot\sum_{f\in F(c)}p_f\hat{\mathbf{n}}_f A_f+\nu\sum_{c\in\Omega}\mathbf{u}_c\cdot\sum_{f\in F(c)}(\mathbf{u}_{nb}-\mathbf{u}_c)\frac{A_f}{\delta d_f},\end{aligned} \tag{4.39}$$

where terms from left to right correspond to time derivative, convection, pressure and diffusion contributions to the kinetic energy equation.

The detailed analysis of Eq. 4.39 is simplified if an important identity involving combinations of interpolation and differentiation operators is introduced. The identity was first presented by Morinishi et al. [5] and restated in finite-volume form by Felten

and Lund [8]. The relation writes

$$\varphi_c \sum_{f\in F(c)} \bar{\psi}_f Q_f + \psi_c \sum_{f\in F(c)} \bar{\varphi}_f Q_f = \sum_{f\in F(c)} \widehat{\varphi\psi} Q_f + (\varphi_c\psi_c) \sum_{f\in F(c)} Q_f, \tag{4.40}$$

where φ and ψ are two general variables, Q_f is a general quantity known on the cell face, i.e., no interpolation is needed, the overbars refer to interpolated values, and $\widehat{\varphi\psi} = \frac{1}{2}(\varphi_c\psi_{nb} + \varphi_{nb}\psi_c)$ is a special interpolator operator for products.

First, the convective term of Eq. 4.39 is transformed by specializing Eq. 4.40 to $\varphi = \mathbf{u}$, $\psi = \boldsymbol{\phi}$ and $Q_f = \hat{U}_f A_f$, then, using the continuity equation, Eq. 4.1, assuming that $\mathbf{u}_f = \frac{1}{2}(\mathbf{u}_c + \mathbf{u}_{nb})$ and canceling out equal terms, the convective expression can be rewritten as

$$\sum_{c\in\Omega} \mathbf{u}_c \cdot \sum_{f\in F(c)} \boldsymbol{\phi}_f \hat{U}_f A_f = \sum_{c\in\Omega} \sum_{f\in F(c)} \frac{1}{2}\mathbf{u}_c \cdot (2\boldsymbol{\phi}_f - \boldsymbol{\phi}_c)\hat{U}_f A_f, \tag{4.41}$$

where $\boldsymbol{\phi}_f$ is evaluated as the semi-sum of the two adjacent cell velocities, i.e., using the symmetry-preserving convective scheme.

Second, if the pressure term in Eq. 4.39 is analyzed in a similar fashion by taking $\varphi = \mathbf{u}$, $\psi = p$ and $Q_f = \hat{\mathbf{n}}_f A_f$, and Eq. 4.13 is used to simplify the expression, the following relation results

$$\begin{aligned}\sum_{c\in\Omega} \mathbf{u}_c \cdot \sum_{f\in F(c)} p_f \hat{\mathbf{n}}_f A_f = \sum_{c\in\Omega} \sum_{f\in F(c)} \widehat{\mathbf{u}p} \cdot \hat{\mathbf{n}}_f A_f - \frac{\delta t}{\rho} \sum_{c\in\Omega} p_c \sum_{f\in F(c)} \left[(p_{nb} - p_c) \frac{A_f}{\delta d_f} \right] \\ + \frac{\delta t}{\rho} \sum_{c\in\Omega} p_c \sum_{f\in F(c)} \frac{1}{2} \left[\frac{1}{V_c} \sum_{f\in F(c)} p_f \hat{\mathbf{n}}_f A_f + \frac{1}{V_{nb}} \sum_{f\in F(nb)} p_f \hat{\mathbf{n}}_f A_f \right] \cdot \hat{\mathbf{n}}_f A_f.\end{aligned} \tag{4.42}$$

Finally, noticing that interior fluxes in Eq. 4.41 and Eq. 4.42 cancel out, Eq. 4.39 can be rewritten as

$$\begin{aligned}\sum_{c\in\Omega} \frac{d(\frac{1}{2}\mathbf{u}_c \cdot \mathbf{u}_c)}{dt} V_c + \sum_{f\in F(\partial\Omega)} \frac{1}{2}\mathbf{u}_a \cdot (2\boldsymbol{\phi}_f - \boldsymbol{\phi}_a)\hat{U}_f A_f = -\frac{1}{\rho} \sum_{f\in F(\partial\Omega)} \widehat{\mathbf{u}p} \cdot \hat{\mathbf{n}}_f A_f \\ - \frac{\delta t}{\rho^2} \sum_{c\in\Omega} p_c \sum_{f\in F(c)} \frac{1}{2} \left[\frac{1}{V_c} \sum_{f\in F(c)} p_f \hat{\mathbf{n}}_f A_f + \frac{1}{V_{nb}} \sum_{f\in F(nb)} p_f \hat{\mathbf{n}}_f A_f \right] \cdot \hat{\mathbf{n}}_f A_f \\ + \frac{\delta t}{\rho^2} \sum_{c\in\Omega} p_c \sum_{f\in F(c)} \left[(p_{nb} - p_c) \frac{A_f}{\delta d_f} \right] + \nu \sum_{c\in\Omega} \mathbf{u}_c \cdot \sum_{f\in F(c)} (\mathbf{u}_{nb} - \mathbf{u}_c) \frac{A_f}{\delta d_f},\end{aligned} \tag{4.43}$$

which states, that in the absence of viscosity ($\nu = 0$), the change in kinetic energy is due to the fluxes through the boundary of the domain and a kinetic energy error from the pressure term. This pressure error term arises from the special definition for the normal face velocity, Eq. 4.13, needed to exactly conserve mass on collocated schemes. Notice that when using first-order interpolations (semi-summed variables from adjacent cells) and a symmetry-preserving convection scheme, the kinetic energy conservation error is minimized.

It is of great interest to evaluate the scaling order of the kinetic energy pressure error since it can not be eliminated. Looking in detail Eq. 4.43, it can be shown that the spatial pressure error scales like $\mathcal{O}(\Delta h^2)$, as deduced by Felten and Lund [8], while the time pressure error scales as $\mathcal{O}(\Delta t)$, but can be reduced through the use of different time integration schemes, as studied by Fishpool and Leschziner [23].

This result can be related to the symmetries of discrete operators in the following way: (1) the convective term in Eq. 4.43 presents no kinetic energy error, since the convection scheme has been chosen to make the convective operator skew-symmetric; (2) the need for a special definition for the normal face velocity, Eq. 4.13, makes the divergence-gradient relation, $\mathbf{M} = -\mathbf{G}^*$, not true, therefore, a pressure gradient error term arises in Eq. 4.43.

Staggered kinetic energy conservation

The staggered kinetic energy equation starts from the staggered momentum equation, Eq. 4.31. First, Eqs. 4.32 to Eq. 4.37 are used to recast the summation over faces as a summation over cells. Second, the resulting equation is multiplied by velocity, $\mathbf{u}$. In this way, the staggered kinetic energy equation is shown to obey the same equation as in the collocated case, Eq. 4.39.

Next, the convective term is converted to flux form as done for the collocated case, Eq. 4.41, while the pressure term is analyzed by specializing Eq. 4.40 as $\varphi = \mathbf{u}$, $\psi = p$, $Q_f = \hat{\mathbf{n}}_f A_f$ and noticing that no special definition is needed for the normal face velocity, giving

$$\sum_{c\in\Omega} \mathbf{u}_c \cdot \sum_{f\in F(c)} p_f \hat{\mathbf{n}}_f A_f = \sum_{c\in\Omega} \sum_{f\in F(c)} \widehat{\mathbf{u}p} \cdot \hat{\mathbf{n}}_f A_f. \tag{4.44}$$

Finally, knowing that interior fluxes cancel out, Eq. 4.39 is rewritten as

$$\begin{aligned} \sum_{c\in\Omega} \frac{d(\frac{1}{2}\mathbf{u}_c \cdot \mathbf{u}_c)}{dt} V_c + \sum_{f\in F(\partial\Omega)} \frac{1}{2}\mathbf{u}_a \cdot (2\boldsymbol{\phi}_f - \boldsymbol{\phi}_a)\hat{U}_f A_f \\ = -\frac{1}{\rho} \sum_{f\in F(\partial\Omega)} \widehat{\mathbf{u}p} \cdot \hat{\mathbf{n}}_f A_f + \nu \sum_{c\in\Omega} \mathbf{u}_c \cdot \sum_{f\in F(c)} (\mathbf{u}_{nb} - \mathbf{u}_c) \frac{A_f}{\delta d_f}, \end{aligned} \tag{4.45}$$

which states that in the absence of viscosity ($\nu = 0$), the change in kinetic energy is due only to the fluxes through the boundary of the domain.

In this case, the two necessary discrete operator properties needed to conserve kinetic energy are fulfilled: (1) the convective term is evaluated by a symmetry-preserving convective scheme making the discrete convective operator skew-symmetric; (2) the normal face velocity does not need a special definition since it is the primary quantity, then, the divergence-gradient relation, $\mathbf{M} = -\mathbf{G}^*$, holds true.

4.4 Conservation and accuracy tests

Three different problems will be solved to test the conservation properties and accuracy of the unstructured mesh schemes previously presented. First, conservation properties will be analyzed by solving a Rankine vortex with zero mass flux at the boundaries. Second, an accuracy assessment will be presented using an exact sinusoidal function. Finally, the schemes will be tested in a turbulent flow over a circular cylinder at Reynolds number 3900.

4.4.1 Rankine vortex

In order to test the conservation properties of the schemes presented in Sec. 4.2, the Rankine vortex problem is chosen because it has zero mass flux at the boundaries, but is inherently unsteady. It is a two-dimensional flow since the motion only occurs in the xy-plane.

The Rankine vortex model is given by the combination of a rigid-body rotation within a core and a decay of angular velocity outside. The tangential velocity, u_θ, of a Rankine vortex with circulation, Γ, and radius, R, is given by

$$u_\theta(r) = \begin{cases} \Gamma r/2\pi R^2 & r \leqslant R, \\ \Gamma/2\pi r & r > R. \end{cases} \tag{4.46}$$

In particular, the Rankine vortex solved in this work is placed at the center of a 3-D domain (1.0×1.0×h), the initial tangential velocity reaches a maximum of 0.16 m/s at radius $R = 0.01\ m$, and circulation equals $\Gamma = 0.032\pi\ m^2/s$. The density and viscosity of the fluid are $\rho = 1.0\ kg/m^3$ and $\nu = 0.01\ m^2/s$, respectively. The domain is meshed with 3665 unstructured triangular prisms, which corresponds to a mesh size of $h = 0.025$, the time step is fixed at $\Delta t = 0.0025\ s$, and all boundaries are slip walls.

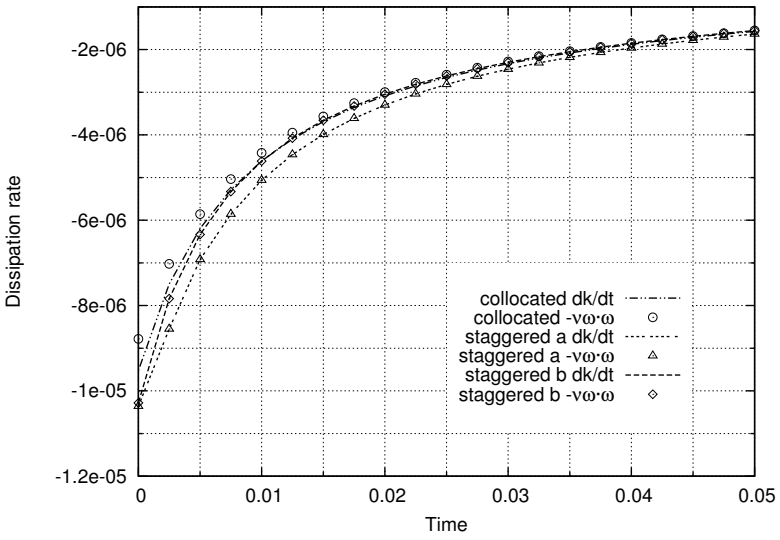


Figure 4.3: Dissipation rate of kinetic energy using collocated, staggered a and b mesh schemes versus time with $\nu = 0.01$.

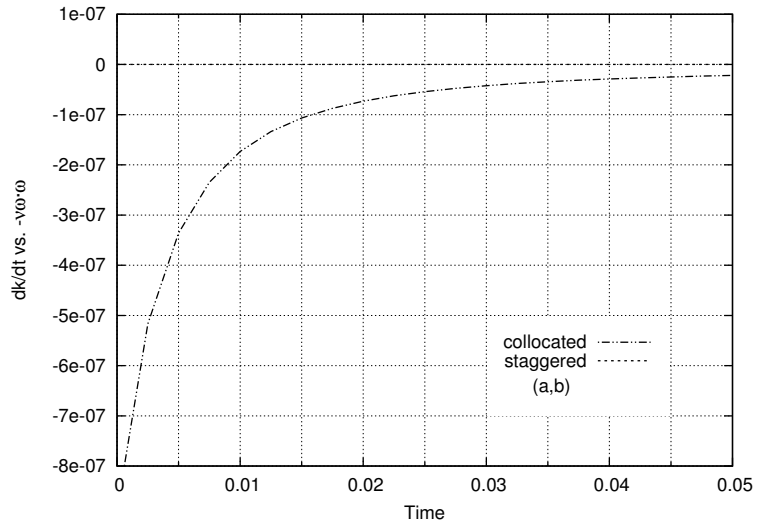


Figure 4.4: Difference between kinetic energy rate of change and physical dissipation using collocated, staggered a and b (same line) mesh schemes versus time with $\nu = 0.0$.

The total discrete momentum for each mesh scheme is calculated at every time step using Eq. 4.29. Results corroborate that collocated and both staggered a and b discretizations conserve total momentum exactly as expected from Eq. 4.29, since there is no flow across the domain boundaries. In this problem, velocity's x and y components are symmetric about the vortex axis and its z-component is zero, thus, the initial total momentum is zero and it remains constant through the test.

The rate of change of total kinetic energy, $dk/dt = d(\frac{1}{2}\mathbf{u}\cdot\mathbf{u})/dt$, and the total physical dissipation, $-\nu\boldsymbol{\omega}\cdot\boldsymbol{\omega}$, for each mesh scheme are calculated at every time step using Eqs. 4.43 and 4.45, and results are plotted in Fig. 4.3. Since there is no flow across the domain boundaries, the change of total kinetic energy with time should be due only to the effect of dissipation, as described by Eq. 4.38, i.e., dk/dt and $-\nu\boldsymbol{\omega}\cdot\boldsymbol{\omega}$ should match for each time instant. Looking in detail at Fig. 4.3, it can be seen that for both staggered schemes the rate of change of total kinetic energy and physical dissipation exactly coincide, while in the collocated case a subtle difference can be appreciated and is related to the pressure error term present in Eq. 4.43.

The kinetic energy error can be studied in detail if the viscosity is set to zero, $\nu = 0.0$. Then, if any difference exists between physical dissipation and rate of change of total kinetic energy, it is due to the pressure error term. Results in Fig. 4.4 show that both staggered schemes numerically conserve kinetic energy since the difference between the two quantities is zero at each time instant, while the collocated scheme presents a decreasing difference of order 10^{-7}.

It is interesting to study numerically the scaling order of this kinetic energy error intrinsic to the collocated mesh scheme, which depends on mesh size and time integration, as previously analyzed in Sec. 4.3.3. First, the comparison between this pressure error term and mesh size is evaluated by solving the Rankine vortex test with zero viscosity on five succesively refined meshes ($h = 0.05$ to $h = 0.0125$) with a fixed time step $\Delta t = 5 \times 10^{-4}\ s$. Second, the relation between the pressure error term and the time integration is analyzed by solving the same test on the $h = 0.025$ mesh, while trying three different time steps (5×10^{-4}, 1×10^{-4} and 5×10^{-5}) using Euler (first-order) and second-order gear like [23] integration schemes.

The mesh size study is plotted in Fig. 4.5. Results show that if the mesh is refined, the difference between rate of change of kinetic energy and physical dissipation is reduced in a second-order manner (overall order equals 1.87). This result matches with the theoretical approach introduced in Sec. 4.3.3, which states that the spatial pressure error scales like $\mathcal{O}(\Delta h^2)$. Additionally, it is important to notice that when solving turbulent problems using direct numerical simulation (DNS) or large-eddy simulation (LES), the mesh size is small enough to make the kinetic energy error imperceptible for the physics of such problems. Proof of the barely affected kinetic energy conservation, if using meshes fine enough, is the works by Rodríguez et al. [17, 18], which solve turbulent flows using the collocated scheme analyzed in this paper.

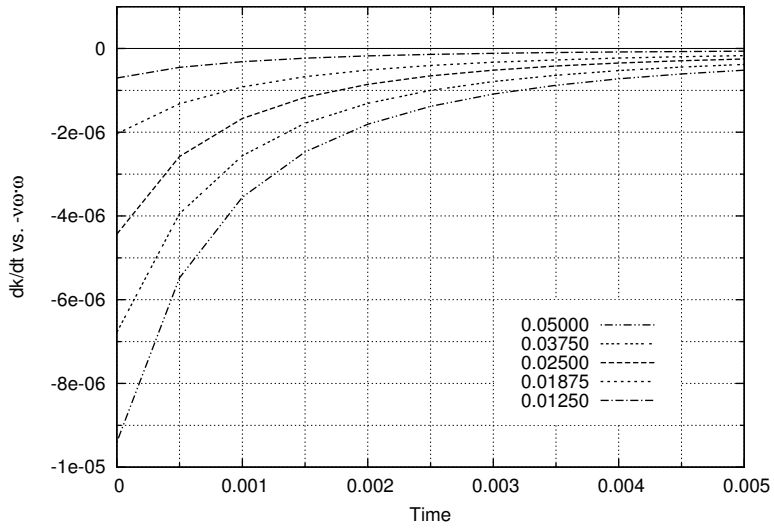


Figure 4.5: Difference between kinetic energy rate of change and physical dissipation for the collocated mesh scheme versus time using different mesh sizes with $\nu = 0.0$.

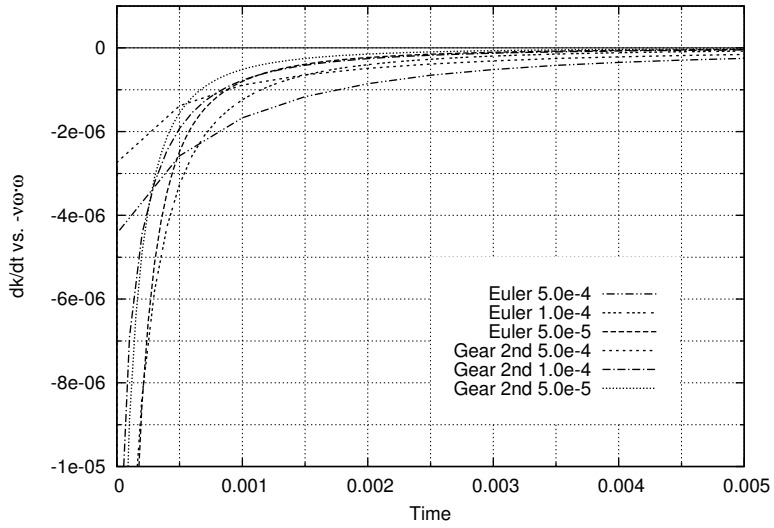


Figure 4.6: Difference between rate of change of total kinetic energy and total physical dissipation for the collocated mesh scheme versus time using different time integrations with $\nu = 0.0$.

Results of the time integration analysis are plotted in Fig. 4.6. Some remarks can be made from close look at the figure. The first conclusion is that successively smaller time steps converge to smaller errors for both time integration methods, although bigger time steps present less error at the initial steps. The second conclusion is related to the time integration method. The analysis of the kinetic energy conservation for the collocated mesh scheme, Sec. 4.3.3, has been developed using a first-order explicit time integration method for simplicity, but, as proposed by Fishpool and Leschziner [23], using other time integration methods may decrease the kinetic energy error. Hence, it is clear from Fig. 4.6 that if a second-order gear like integration time scheme is used, results are greatly improved compared to the ones obtained by a simple Euler time integration. The reason for this improvement arises from the fact that when using gear-like time integration schemes, the time step multiplying the pressure error term in Eq. 4.43 is diminished by a scaling factor (2/3 for a second-order case), therefore, the pressure error term is consequently minimized and so is the difference between the rate of change of kinetic energy and physical dissipation.

In summary, this test demonstrates numerically the discrete conservation properties introduced theoretically in Sec. 4.3. First, staggered mesh schemes discretely preserve momentum and kinetic energy exactly. Second, the collocated mesh scheme conserves momentum, but presents a kinetic energy error of the form $\mathcal{O}(\Delta t^m, \Delta h^2)$, due to an improper pressure gradient formulation.

4.4.2 Numerical tests of accuracy: exact sinusoidal function

Accuracy of the different mesh schemes presented in this work is studied by comparing numerical results to the analytical solution of an exact sinusoidal function. In each case, a sinusoidal function is assigned to the input variables: cell-centered velocities, $\mathbf{u}$, in the collocated case, while normal face velocities, U, in the staggered ones. Then, numerical normal face velocities for the collocated case are obtained from Eq. 4.13, considering the ideal situation where pressure terms vanish, while numerical cell-centered velocities are calculated from Eqs. 4.18 and 4.25 for staggered a and b cases, respectively. Finally, the root-square-mean error (rms), x_{rms}, is calculated by comparing analytical and numerical results, which is defined as

$$x_{rms} = \sqrt{\frac{1}{n}({x_1}^2 + \cdots + {x_n}^2)}, \tag{4.47}$$

where x_i corresponds to each individual error out of n values.

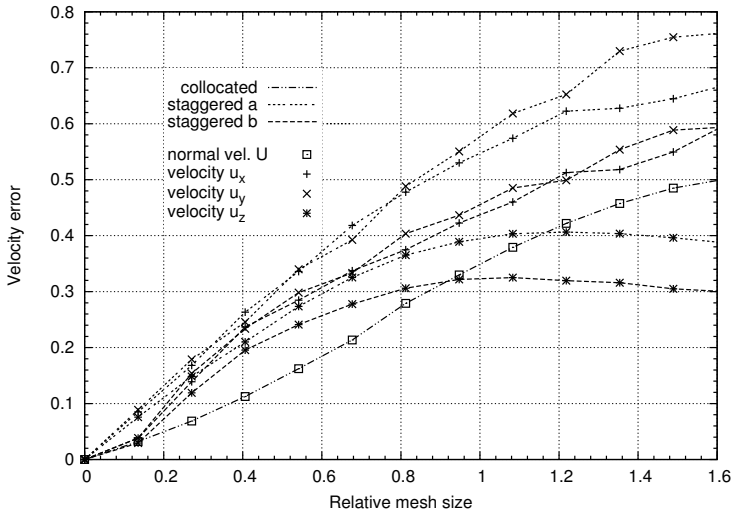


Figure 4.7: Velocity error, x_{rms}, versus relative mesh size. Normal face velocity, U, is analyzed for the collocated case, and the three components of velocity; u_x, u_y and u_z, for the staggered ones.

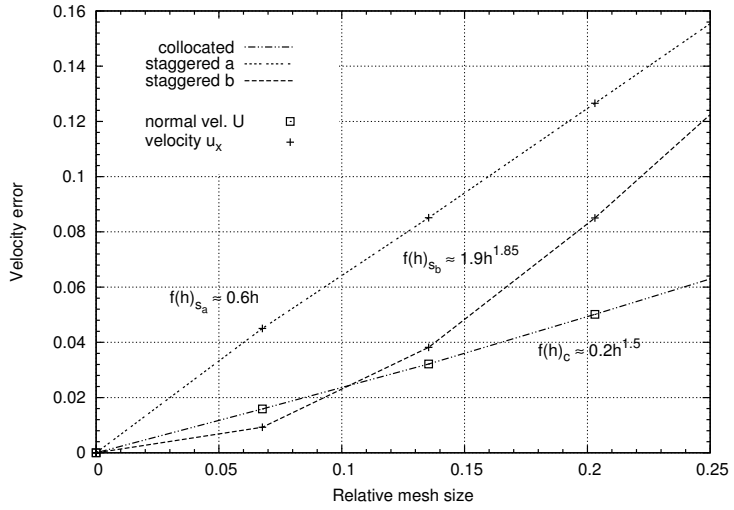


Figure 4.8: Velocity error, x_{rms}, versus relative mesh size. Normal face velocity, U, is analyzed for the collocated case, and the x-component of velocity for the staggered ones. Approximated regression equations are calculated.

The stream function is set to be $\boldsymbol{\psi} = \frac{1}{2\pi N} sin(2\pi Nx)cos(2\pi Ny)\mathbf{k}$. Hence, derivation of $\boldsymbol{\psi}$, $\mathbf{u} = \nabla \times \boldsymbol{\psi}$, gives the following velocity field

$$\begin{aligned} u &= -sin(2\pi Nx)sin(2\pi Ny), \\ v &= -cos(2\pi Nx)cos(2\pi Ny), \\ w &= 0, \end{aligned} \tag{4.48}$$

with a maximum velocity magnitude of one.

Instead of changing the mesh size, which is a cube (1.0×1.0×1.0) meshed with 9676 tetrahedra, mesh refinement is performed by changing the wavelength of the input sine functions. The average mesh volume is calculated as $V_{avg} = \frac{1}{c}\sum_c V_c$, giving an average mesh spacing equal to $\Delta X_{avg} = \sqrt[3]{3V_{avg}} = 0.068$. The effective length of the domain is defined as $L_{eff} = 1/N$, where N is a variable integer, such that larger values of N correspond to coarser effective meshes. Thus, the relative mesh size is defined to be $h = \Delta X_{avg}/L_{eff} = 0.068N$.

Errors of velocity accuracy are obtained for relative mesh sizes ranging from 0 to 1.5 and are plotted in Fig. 4.7. The results show that collocated normal face velocity errors are smaller than staggered cell-centered velocity ones for almost all relative mesh sizes, considering the ideal situation where pressure terms in Eq. 4.13 vanish. On the other hand, the staggered *b* scheme presents slightly smaller errors than the staggered *a* one for all three velocity components and relative mesh sizes. Moreover, Fig. 4.8 takes a close look at U and u_x errors obtained for relative mesh sizes ranging from 0 to 0.25 and calculates their approximated regression equations. It is observed that collocated and staggered *b* errors are almost second-order, while staggered *a* ones are just first-order. Additionally, staggered *a* errors for relative mesh sizes between 0.05 and 0.15 are two times larger than collocated and staggered *b* ones, hence, considerable accuracy differences may be observed when solving problems using this range of relative mesh sizes.

4.4.3 Turbulent flow over a circular cylinder at Re = 3900

As final test, conservation properties and accuracy are assesed by solving the turbulent flow past a circular cylinder. This is an extensively used canonical case to perform studies of the turbulence behavior around bluff bodies. Additionally, it is a flow pattern found in many practical situations where cylindrical structures exist, e.g., towers, power supply wires, heat exchangers, and others.

Experiments provide evidence that the flow around a circular cylinder behaves differently depending on the Reynolds number. For example, steady laminar flow is obtained, with two vortices forming after the cylinder, for Reynolds numbers up to nearly 40. Furthermore, the von Kármán vortex street is formed at Reynolds numbers up to 190. Then, the flow evolves from two to three dimensionality [24]

around Reynolds number 260. Later, the shear layers detaching from the cylinder become turbulent at Reynolds number around 1200, as reported by Prasad and Williamson [25]. However, the value of laminar-to-turbulent transition varies in the literature from 300 to 3000. Finally, the boundary layer on the cylinder becomes turbulent before separation at postcritical Reynolds numbers [26], beyond 3.5×10^6.

In this test, the effects of the mesh discretization on the flow past a circular cylinder at $Re_D = U_{ref} D/\nu = 3900$ are studied. This case has been largely investigated both experimentally and numerically, e.g., by Kravchenko and Moin [27], Ma et al. [28], Parnaudeau et al. [29], and Lehmkuhl et al. [10]. Hence, there is much knowledge reported in the literature. Moreover, notice that it is clearly turbulent, since it is situated beyond the laminar-to-turbulent transition point (3.0×10^3), therefore, the consequences of the unwanted numerical dissipation for the different schemes can be analyzed in detail.

Problem definition and computational domain

The flow over a circular cylinder is solved using a computational domain of dimensions [-4D,10D]×[-6D,6D]×[0,πD] in the stream-, cross-, and span-wise directions, respectively, with a circular cylinder of diameter D placed at position $(0, 0, 0)$. The boundary condition at inflow consists of a uniform velocity $(u, v, w) = (1, 0, 0)$, slip conditions are imposed at top and bottom boundaries, at outlet a pressure-based condition is used, no-slip conditions are prescribed at the cylinder's surface, and periodic boundary conditions are imposed at the spanwise direction.

The Navier-Stokes equations are discretized and the problem is solved for up to 200 time units using an unstructured mesh of 2.78M cells generated by the constant-step extrusion of a two-dimensional grid (43446 triangles $\times$ 64 planes). Under these conditions, the spanwise coupling of the discrete Poisson equation, Eq. 4.8, yields circulant submatrices that are diagonalizable in a Fourier space. This allows us to solve the Poisson equation by means of a Fast Fourier Transform (FFT) method. The algorithm used is based on the explicit calculation and direct solution of a Schur complement system for the independent 2-D systems. For more details the reader is referred to Borrell et al. [30].

The purpose of this study is not to compute the DNS solution of the problem, instead, the main idea is to analyze the accuracy and conservation of kinetic energy for turbulent flows. Consequently, the mesh used is clearly not fine enough, therefore, the problem is solved with and without carrying LES. In the present work, the wall-adapting local eddy-viscosity (WALE) [31] is used to model the subgrid scales (SGS).

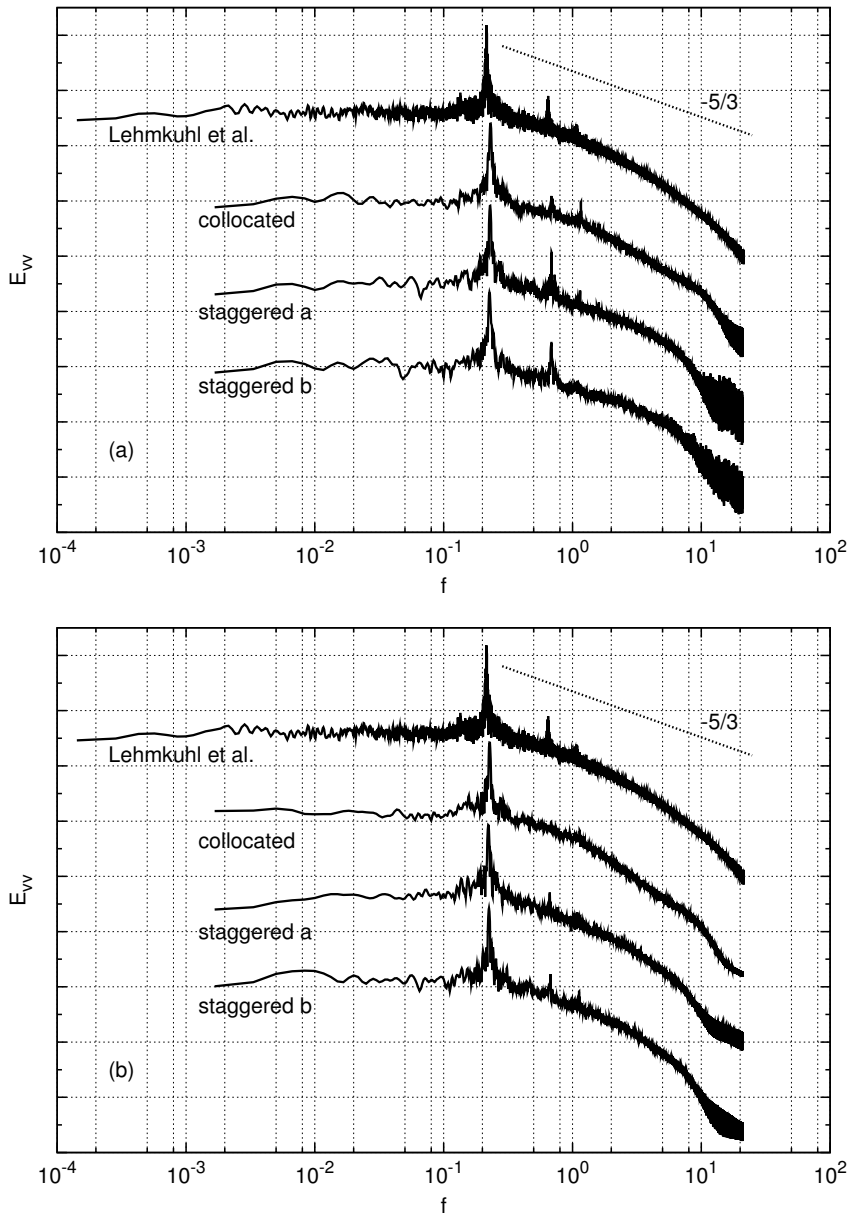


Figure 4.9: Scaled energy spectra of the cross-stream velocity at $P \equiv [x/D = 2.0, y/D = 0.0]$ compared to Lehmkuhl et al. [10] numerical results: (a) using no SGS model; (b) with SGS model.

Energy spectrum

In order to analyze the turbulent steady state, measurements have been carried out by locating probes at different positions and data have been collected from time 50 to 200 time units. In short, results are just presented for the probe located at $P \equiv [x/D = 2.0, y/D = 0.0]$ in the wake centerline. The main frequencies of the fluctuations of the cross-stream velocity component have been computed by using the Lomb periodogram technique. The resulting spectra have also been averaged in the azimutal direction. Results are plotted against DNS results from Lehmkuhl et al. [10] in Fig. 4.9.

For all three mesh schemes, the different spectra follow the main features of the DNS one. For example, the dominant peaks at $f_{vs} = 0.21$, which correspond with the large-scale vortex shedding frequency, are in agreement with the values reported in the literature; e.g., $f_{vs} = 0.21$ (Kravchenko and Moin [27]), $f_{vs} = 0.203$ (Ma et al. [28]), $f_{vs} = 0.208$ (Parnaudeau et al. [29]), and $f_{vs} = 0.215$ (Lehmkuhl et al. [10]).

Some important differences between results are observable. First, collocated results, with and without the SGS model, present larger dynamic ranges than staggered ones. This means that when using fine enough meshes and small time steps, the collocated kinetic energy error is surely minimized, as demonstrated in Sec. 4.4.1. Hence, under these conditions the collocated's accuracy outperforms the staggered ones; see Sec. 4.4.2. Thus, the small dissipative scales are better resolved and able to extract an improved amount of energy. Second, carrying out LES adds artificial dissipation to model the SGS, hence, the high-frequency energy hump, observable in Fig. 4.9a due to the insufficient amount of energy dissipation provided by the mesh, is minimized for all three mesh schemes. However, the addition of this extra dissipation may affect intermediate energy frequencies. For example, when modelling the SGS, the f_{vs}'s second harmonic is minimized for all three mesh schemes, although, it is more noticeable in the collocated case.

Average statistics in the wake

In order to analyze the wake configuration, the time-average streamwise velocity and streamwise velocity profile at $x/D = 1.06$ have been computed. In Figs. 4.10 and 4.11, average results are plotted against values reported in the literature: experimental results from Parnaudeau et al. [29] and numerical results from Lehmkuhl et al. [10] (short recirculation zone, Mode S solution).

Results in Figs. 4.10 and 4.11 clearly demonstrate that the use of a SGS scheme is needed since the mesh used is not fine enough to properly calculate the dissipative small scales. If no LES is carried out, both collocated and staggered a and b solutions are far from matching literature results, although, solutions rapidly coincide with reference ones if SGS models are introduced. However, a reasonable conclusion taken

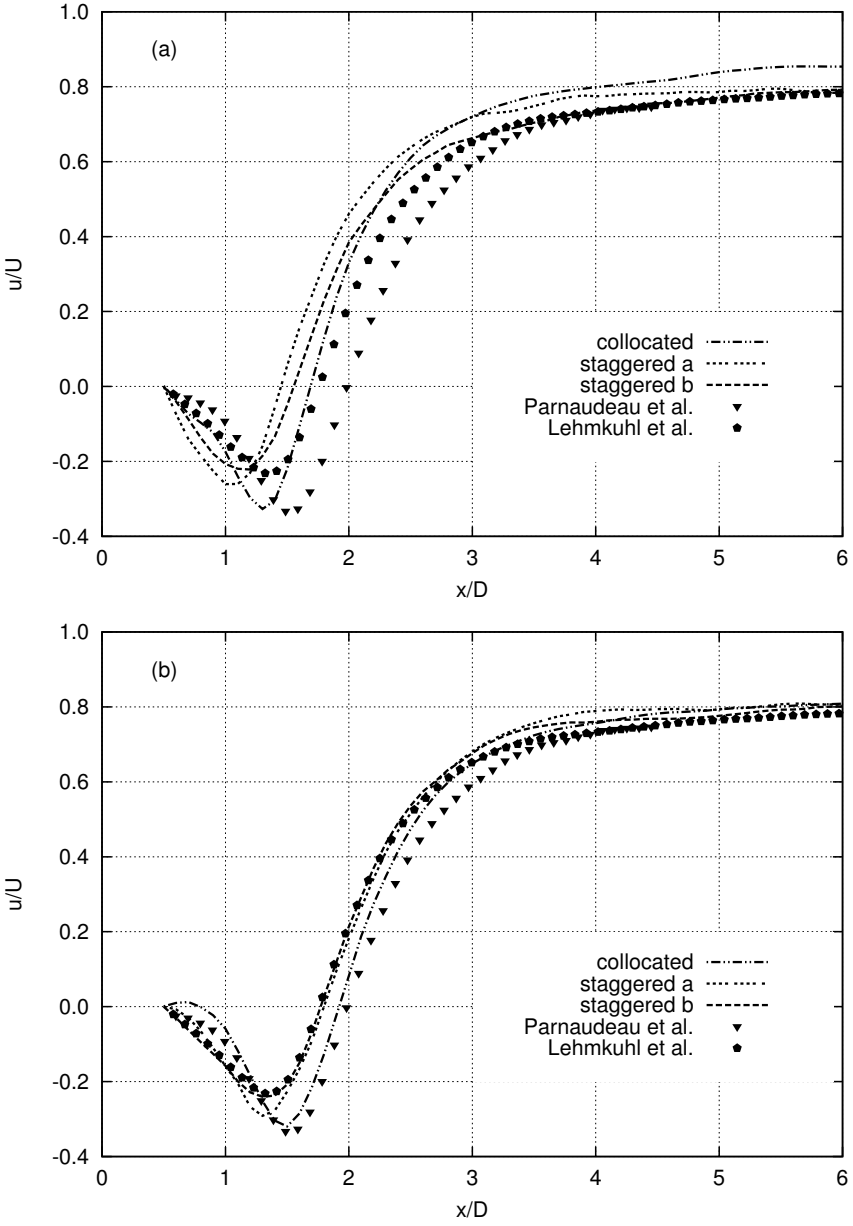


Figure 4.10: Averaged streamwise velocity in the wake compared to Parnaudeau et al. [29] and Lehmkuhl et al. [10] results: (a) without SGS model; (b) using SGS model.

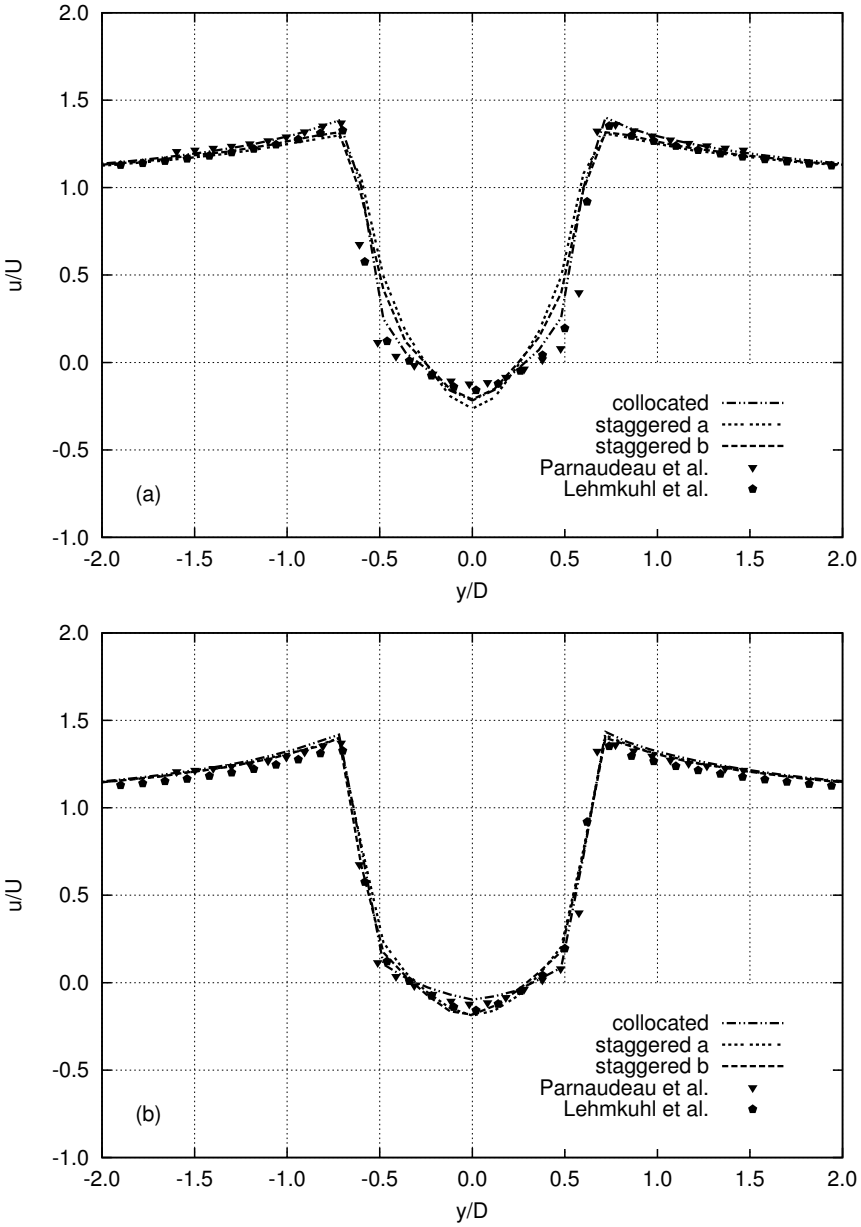


Figure 4.11: Averaged streamwise velocity profile at $x/D = 1.06$ compared to Parnaudeau et al. [29] and Lehmkuhl et al. [10] results: (a) using no SGS model; (b) with SGS model.

from Figs. 4.10 and 4.11 is that using the collocated or staggered schemes tends to favor significantly different solutions, even though larger time units simulations should be analyzed.

The collocated scheme produces an averaged streamwise velocity solution similar to the one obtained by Parnaudeau et al. [29], presenting a long and less energy-concentrated recirculation zone $L_r/D = 1.51$, see Fig. 4.10b, and a more pronounced U-shape average streamwise velocity profile at $x/D = 1.06$; see Fig. 4.11b. On the other hand, streamwise velocity results provided by staggered schemes resemble the ones presented by Lehmkuhl et al. [10], thus, shorter dense energy-concentrated recirculation zones are observed $L_r/D = 1.26$, see Fig. 4.10b, and a V-shape average streamwise velocity profile at $x/D = 1.06$ is recognized in Fig. 4.11b.

It is important to notice that the staggered b streamwise velocity result, using the SGS model, fits in a better manner than the staggered a the solution obtained by Lehmkuhl et al. [10], specifically, the averaged velocity at $L_r/D = 1.26$. This ameliorate performance of staggered b over staggered a is explained by the better accuracy of the first over the latter when fine meshes are used, as seen in Fig. 4.8.

4.5 Conclusions

The continuity and Navier-Stokes equations are specifically derived for the conservation of mass and momentum, thus, collocated and both staggered schemes are shown to discretely conserve them: Eq. 4.27 is the proof of mass conservation, while Eq. 4.29 states that the change in momentum is due only to the fluxes through the domain boundaries. On the other hand, conservation of kinetic energy is the most important property when solving turbulent flow, since the energy is convected from the large eddies to the small dissipative scales. Hence, discretization strategies that incorporate extra numerical dissipation can importantly modify the solution of the problem. Consequently, collocated and staggered discrete expressions of kinetic energy conservation are presented in Eqs. 4.43 and 4.45, respectively. They state that, in the absence of viscosity ($\nu = 0$), the change in kinetic energy is due to the fluxes through the boundary of the domain for the staggered schemes, plus a kinetic energy error from the pressure term for the collocated one. This pressure error term arises from the special definition for the normal face velocity, Eq. 4.13, needed to exactly conserve mass in the collocated scheme.

The Rankine vortex test has shown that staggered mesh schemes preserve momentum and kinetic energy, while the collocated one conserves momentum, but presents a kinetic energy error of the form $\mathcal{O}(\Delta t^m, \Delta h^2)$. Thus, densifying meshes and using small time steps or high-order temporal schemes decreases the collocated kinetic energy error. In this way, when solving turbulent problems using DNS or LES, the mesh size and time steps are small enough to make the kinetic energy error

imperceptible for the physics of such problems.

An accuracy study for the different mesh schemes has been performed by comparing numerical results to the analytical solution of an exact sinusoidal function. Results show that collocated normal face velocity errors are smaller than staggered cell-centered velocity ones for all relative mesh sizes, considering the ideal situation where pressure terms in Eq. 4.13 vanish. On the other hand, staggered *b* accuracy results present slightly smaller errors than staggered *a* ones for all three velocity components and relative mesh sizes. Moreover, regression equations have been calculated for the three schemes in a relative mesh size range from 0 to 0.25, showing that collocated and staggered *b* accuracy errors are nearly second-order, while the staggered *a* ones are first-order.

The turbulent flow over a circular cylinder at $Re = 3900$ has been solved using the collocated and both staggered schemes to test their properties in more demanding problems. Although all three mesh schemes present good agreement with literature results if LES are carried out, some main differences between them have been found. First, collocated results present larger dynamic ranges than staggered ones, due to a better resolution of the small dissipative scales, thus, extracting more energy out of the system. This result demonstrates that if using fine enough meshes and small time steps, the collocated kinetic energy error is certainly minimized, hence, under these conditions the collocated's accuracy outperforms the staggered ones. Second, the collocated scheme tends to produce a long and less energy-concentrated recirculation zone with a more pronounced U-shape average streamwise velocity profile. On the other hand, staggered schemes favor short dense energy-concentrated recirculation zones with V-shape average streamwise velocity profiles. Moreover, as demonstrated in the accuracy test, the staggered *b* scheme presents better accuracy performance, thus, its streamwise velocity result, using the SGS model, fits the reference solution in a better manner than the staggered *a* one.

As final summary, the authors conclude that if incompressible turbulent flow is to be solved, using time-explicit algorithms with fine unstructured meshes and small time steps, the collocated scheme is a better option over the staggered ones: (1) the pressure kinetic energy error is unnoticeable in such situations; (2) presents good accuracy; (3) it is a fast scheme that does not need the calculation of circumcenters. However, the use of the collocated scheme to solve problems regarding other fluid or flow characteristics, e.g., fluids with nonconstant physical properties or with high gradients, presence of discontinuous sources, multiphase flow, combustion problems, or others, may produce spurious pressure modes (checkerboard). In these situations the staggered schemes presented in this study are a good alternative, especially the staggered *b* mesh discretization, since it presents better accuracy than the staggered *a* one, although it requires a more complicated and computationally demanding cell-centered velocity reconstruction.

Acknowledgements

This work has been financially supported by the *Ministerio de Economía y Competitividad, Secretaría de Estado de Investigación, Desarrollo e Innovación*, Spain (ENE-2010-17801), a FPU Grant by the *Ministerio de Educación, Cultura y Deporte*, Spain (AP-2008-03843) and by *Termo Fluids S.L.*

The authors would like to acknowledge sincerely Ivette Rodríguez, Ricard Borrell and Carlos D. Pérez-Segarra for providing their numerical data of the turbulent flow over a circular cylinder at $Re = 3900$.

References

[1] S. V. Patankar. *Numerical Heat Transfer and Fluid Flow*. Hemisphere Publishing Corporation, 1980.

[2] C. M. Rhie and W. L. Chow. Numerical Study of the Turbulent Flow Past an Airfol with Trailing Edge Separation. *AIAA Journal*, 21:1525–1532, 1983.

[3] L. Davidson. A Pressure Correction Method for Unstructured Meshes with Arbitrary Control Volumes. *International Journal for Numerical Methods in Fluids*, 22:265–281, 1996.

[4] S. R. Marthur and J. Y. Murthy. A Pressure-Based Method for Unstructured Meshes. *Numerical Heat Transfer, Part B*, 31:195–215, 1997.

[5] Y. Morinishi, T. S. Lund, O. V. Vasilyev, and P. Moin. Fully Conservative Higher Order Finite Difference Schemes for Incompressible Flow. *Journal of Computational Physics*, 143:90–124, 1998.

[6] K. Mahesh, G. Constantinescu, and P. Moin. A Numerical Method for Large-Eddy Simulation in Complex Geometries. *Journal of Computational Physics*, 197:215–240, 2004.

[7] Shashank, J. Larsson, and G. Iaccarino. A Co-located Incompressible Navier-Stokes Solver with Exact Mass, Momentum and Kinetic Energy Conservation in the Inviscid Limit. *Journal of Computational Physics*, 229:4425–4430, 2010.

[8] F. N. Felten and T. S. Lund. Kinetic Energy Conservation Issues Associated with the Collocated Mesh Scheme for Incompressible Flow. *Journal of Computational Physics*, 215:465–484, 2006.

[9] O. Lehmkuhl, I. Rodríguez, A. Báez, A. Oliva, and C. D. Pérez-Segarra. On the Large-Eddy Simulations for the Flow Around Aerodynamic Profiles Using Unstructured Grids. *Computers & Fluids*, 84:176–189, 2013.

[10] O. Lehmkuhl, I. Rodríguez, R. Borrell, and A. Oliva. Low-Frequency Unsteadiness in the Vortex Formation Region of a Circular Cylinder. *Physics of Fluids*, 25:085109, 2013.

[11] C. A. Hall, J. C. Cavendish, and W. H. Frey. The Dual Variable Method for Solving Fluid Flow Difference Equations on Delaunay Triangulations. *Computers & Fluids*, 20:145–164, 1991.

[12] R. A. Nicolaides. *The Covolume Approach to Computing Incompressible Flows*. Cambridge University Press, 1993.

[13] J. E. Hicken, F. E. Ham, J. Militzer, and M. Koksal. A Shift Transformation for Fully Conservative Methods: Turbulence Simulation on Complex, Unstructured Grids. *Journal of Computational Physics*, 208:704–734, 2005.

[14] B. Perot. Conservation Properties of Unstructured Staggered Mesh Schemes. *Journal of Computational Physics*, 159:58–89, 2000.

[15] X. Zhang, D. Schmidt, and B. Perot. Accuracy and Conservation Properties of a Three-Dimensional Unstructured Staggered Mesh Scheme for Fluid Dynamics. *Journal of Computational Physics*, 176:764–791, 2002.

[16] F. H. Harlow and J. E. Welch. Numerical Calculation of Time-Dependent Viscous Incompressible Flow of Fluid with Free Surface. *Physics of Fluids*, 8:2182–2189, 1965.

[17] I. Rodríguez, R. Borrell, O. Lehmkuhl, C. D. Pérez-Segarra, and A. Oliva. Direct Numerical Simulation of the Flow over a Sphere at $Re = 3700$. *Journal of Fluid Mechanics*, 679:263–287, 2011.

[18] I. Rodríguez, O. Lehmkuhl, R. Borrell, and A. Oliva. Direct Numerical Simulation of a NACA 0012 in Full Stall. *International Journal of Heat and Fluid Flow*, 43:194–203, 2013.

[19] R. W. C. P. Verstappen and A. E. P. Veldman. Symmetry-Preserving Discretization of Turbulent Flow. *Journal of Computational Physics*, 187:343–368, 2003.

[20] A. Haselbacher and V. Vasilyev. Commutative Discrete Filtering on Unstructured Grids based on Least-Squares Techniques. *Journal of Computational Physics*, 187:197–211, 2003.

[21] D. Vidovic. Polynomial Reconstruction of Staggered Unstructured Vector Fields. *Theoretical and Applied Mechanics*, 36:85–99, 2009.

[22] J. B. Perot. Discrete Conservation Properties of Unstructured Mesh Schemes. *Annual Review of Fluid Mechanics*, 43:299–318, 2011.

[23] G. M. Fishpool and M. A. Leschziner. Stability Bounds for Fractional-Step Schemes for the Navier-Stokes Equations at High Reynolds Number. *Computers & Fluids*, 38:1289–1298, 2009.

[24] C. H. K. Williamson. Vortex Dynamics in the Cylinder Wake. *Annual Review of Fluid Mechanics*, 28:447–539, 1996.

[25] A. Prasad and C. H. K. Williamson. The Instability of the Separated Shear Layer from a Bluff Body. *Physics of Fluids*, 8:1347–1349, 1996.

[26] A. Roshko. Experiments on the Flow Past a Circular Cylinder at very High Reynolds Number. *Journal of Fluid Mechanics*, 10:345–356, 1961.

[27] A. G. Kravchenko and P. Moin. Numerical Studies of Flow over a Circular Cylinder at Re = 3900. *Physics of Fluids*, 12:403–417, 2000.

[28] X. Ma, G. S. Karamanos, and G. E. Karniadakis. Dynamics and Low-Dimensionality of a Turbulent Wake. *Journal of Fluid Mechanics*, 410:29–65, 2000.

[29] P. Parnaudeau, J. Carlier, D. Heithz, and E. Lamballais. Experimental and Numerical Studies of the Flow over a Circular Cylinder at Reynolds Number 3900. *Physics of Fluids*, 20:85101–85115, 2008.

[30] R. Borrell, O. Lehmkuhl, F. X. Trias, and A. Oliva. Parallel Direct Poisson Solver for Discretizations with one Fourier Diagonalizable Direction. *Journal of Computational Physics*, 230:4723–4741, 2011.

[31] F. Nicoud and F. Ducros. Subgrid-Scale Stress Modeling based on the Square of the Velocity Gradient Tensor. *Flow, Turbulence and Combustion*, 62:183–200, 1999.

5

Conservative discretization of multiphase immiscible flow

Main contents of this chapter have been published in:

L. Jofre, O. Lehmkuhl, and A. Oliva. Conservation Properties of Finite-Volume Mesh Schemes for the Simulation of Multiphase Immiscible Flow. To be submitted to *International Journal of Multiphase Flow*, 2014.

Abstract. The simulation of separated multiphase flow, in which the fluids involved are immiscible, is of great importance for different fundamental physics problems and for a large variety of industrial applications. For instance, in the simulation of liquid-gas interfaces, such as bubbly flow, atomization and wave motion, in the design of sprays and combustion processes or in the study of atmospheric phenomena. This particular type of multiphase flow is governed by the continuity and Navier-Stokes equations in the variable-density incompressibility limit, which constitute a general model that describes fluid flow by conserving mass and momentum. However, the conservation of secondary derived quantities, such as kinetic energy —which is fundamental for the correct resolution of turbulence—, cannot be imposed explicitly during the construction of the equivalent discrete model. Therefore, this work presents and analyzes two unstructured finite-volume mesh discretizations, collocated and staggered, able to simulate multiphase flow presenting fluids with different physical properties. In particular, these mesh schemes are constructed such that conserve mass and momentum numerically, while minimize errors in the conservation of kinetic energy. These properties are analyzed both theoretically and numerically, the latter by considering a three-dimensional vortex, an exact sinusoidal function and the drag force on a spherical bubble in a turbulent pipe flow.

5.1 Introduction

Separated multiphase flows of immiscible fluids are found in many engineering applications and fundamental physics problems. For example, in the study of fuel injection processes, in the formation, movement and deformation of bubbles and drops, in the design of sprays and jets or in the simulation of wave motion. These type of flows are usually denominated interfacial flows, since the contact of immiscible fluids produces a thin region named interface that separates them. In detail, these interfacial flows are governed by the continuity and Navier-Stokes equations in the variable-density incompressibility limit. Additionally, they require the resolution of an extra equation that describes the topology of the interface as it changes due the velocity field. On the one hand, the collocated and staggered mesh schemes are the principal models for the calculation of the discrete Navier-Stokes equations. On the other hand, the approaches based on tracking or capturing the interface have emerged as the main methods to describe the interface motion between fluids.

Any collocated mesh discretization calculates velocity and pressure at centers of cells, while requires specific interpolations for some variables and special evaluations of mass fluxes at cell faces. This particular placement of variables may produce checkerboard-pressure solutions caused by the decoupling of velocity and pressure. On the contrary, staggered mesh schemes directly solve mass fluxes at faces, and pressure at centers of cells, thus, these do not display spurious pressure modes. As a counterpart, velocities at centers of cells need to be interpolated from face mass fluxes. In addition, staggered schemes require the utilization of cell circumcenters, what may result in difficulties, since, if cells with poor aspect ratios are used, the circumcenters may be found outside their corresponding cells. Even so, both schemes are suitable for the numerical simulation on three-dimensional (3-D) Cartesian and unstructured grids.

A major difficulty when solving interfacial flow is caused by the interface itself since it is subject to different length and time scales [1]. For example, usually the dominant small-scale is determined by the relation between surface tension and viscosity, while the largest scale corresponds to the change in interface topology due to the velocity field provided by the solution of the momentum equations. Therefore, numerical methods for the simulation of interfacial flow require the selection of the model that properly represents the interface in a discrete manner, as well as the determination of the forces that need to be considered to correctly simulate the physics. Given these requirements, this work chooses to consider interfaces as two-dimensional (2-D) smooth surfaces that represent the discontinuity of density and viscosity, while gravity and surface tension are the forces taken into account.

In the last decade, important efforts have been made to improve the stability and robustness of the discrete models for the simulation of multiphase immiscible flows: (1) studying in detail the Poisson's pressure matrix [2–5] and (2) proposing complex

interpolations of density at cell faces [6–8]. However, hardly any attention has been paid to analyze the conservation properties of such discrete models, in a contrary way to the recent numerical techniques for the simulation of turbulent flow [9–13], which have evolved to discretely preserve mass, momentum and, specifically, kinetic energy, by using first-order skew-symmetric formulations at expenses of increasing the local truncation error. In particular, this new approach has resulted successful for turbulent problems [14–17], changing the priorities of the discretization schemes for the Navier-Stokes equations. Moreover, this new thinking has also been extended to the simulation of variable-density low-Mach number flows on the basis of finite-difference schemes [18–20]. Therefore, this work aims at introducing the idea of conservative discretizations for the simulation of multiphase immiscible flow on 3-D unstructured meshes, extending in this way the recently proposed approach on Cartesian grids by Fuster [21]. In addition, density and viscosity, instead of being evaluated as discontinuous variables at the interface, they are convoluted, since it has been found by Denner and van Wachem [22] that, although questionable from a physical perspective, this remarkably improves the results and is crucial to mantain numerical stability for high density ratios.

Hence, the main purpose of this work is to accurately formulate two schemes suitable for 3-D unstructured meshes, collocated and staggered, that priorize the conservation of the discrete properties of mass, momentum and kinetic energy. In this way, this work extends the idea of fully conservative schemes to multiphase flows having immiscible fluids. First, Sec. 5.2 presents a brief mathematical description of the interface motion between fluids. Second, both discretization strategies for the Navier-Stokes equations are explained in detail in Sec. 5.3. Next, their discrete conservations of mass, momentum and kinetic energy are studied in Sec. 5.4. Finally, various problems are solved in Sec. 5.5 to test their conservation properties and accuracy, such as a three-dimensional vortex, an exact sinusoidal function and the drag force on a spherical bubble in a turbulent pipe flow.

5.2 Motion of the interface between fluids

The interface between two or more immiscible fluids constitutes a material surface whose motion is described by

$$\frac{d\mathbf{x}_\Gamma}{dt} = \mathbf{u}(\mathbf{x}_\Gamma, t), \tag{5.1}$$

where subscript Γ refers to a point on the interface between fluids. In general, the methods used to locate the interface characterized by Eq. 5.1 may be classified in two large groups: interface-tracking and interface-capturing. On the one hand, the interface-tracking approaches chase the interface as it moves: (1) defining the interface as a boundary between two subdomains of a moving grid [23–25] or (2) following

the Lagrangian trajectories of massless particles [26–28]. This procedure simplifies the analysis near the interface, as a counterpart, large topology changes are not easily handled. On the other hand, the interface-capturing approaches describe the motion of the interface by embedding the different fluids into a static grid with the help of scalar values. In particular, two fundamental methods have emerged during the last decades: Volume-of-Fluid [29–31] and Level-Set [32–34]. The Volume-of-Fluid (VOF) method inherently conserves volume and maintains interfaces sharp, but requires complex geometric algorithms. Otherwise, the Level-Set (LS) technique is a fast way to capture interfaces, represented by the middle contour of a signed distance function, but at expenses of not proprerly conserving volume. However, Olsson and Kreiss [35] have recently solved this issue by proposing a special scalar level-set, resulting in a new family of methods named Conservative Level-Set (CLS). Given its improved volume conservation, this approach has gained importance in the last years; see for example [36, 37].

Particularly, this work chooses the VOF method, since its formulation preserves volume, large changes in interface topology are handled properly and interfaces between fluids are described acutely. In detail, this method defines a fluid-volume fraction, C_k, as the portion of volume filled with fluid k, expressed as

$$C_k = \frac{1}{V_\Omega} \int_\Omega H(\mathbf{x} - \mathbf{x}_\Gamma) d\mathbf{x}, \tag{5.2}$$

where H is the Heaviside function, providing, for each fluid k, a continuity equation for the fluid-volume fraction, written as

$$\frac{\partial C_k}{\partial t} + \nabla \cdot (C_k \mathbf{u}) = 0. \tag{5.3}$$

Then, integrating Eq. 5.3 over a cell c, applying the divergence theorem to its bordering faces, $f \in F(c)$, assuming a first-order explicit time scheme and considering incompressible flow, results in the following fluid k discrete continuity equation

$$C_k^{n+1} - C_k^n + \frac{1}{V_c} \sum_{f \in F(c)} V_{k,f}^n = 0, \tag{5.4}$$

where V_c represents the volume of cell c, the superscript n refers to the discrete time level, $V_{k,f}^n$ is the fluid k volumetric flow across face f and the total volumetric flow is defined as

$$V_f = |\mathbf{u}_f \cdot \mathbf{n}_f| A_f \Delta t = \sum_k V_{k,f}, \tag{5.5}$$

defining $\mathbf{u}_f$, $\mathbf{n}_f$ and A_f as the velocity, the normal outward unit vector and the surface of face f, respectively, while Δt refers to time step.

In particular, volume tracking VOF methods calculate geometrically the solution of Eq. 5.4 in two consecutive steps, denominated: interface reconstruction and advection. First, the interface is reconstructed by approximating its form to a geometric surface from volume fraction values. Once the interface is reconstructed, the advection step constructs volume fluxes at cell faces, cuts them by the reconstructed interface and, finally, computes the amount of fluid k passing through the faces in a time step. For an extended development of the geometrical VOF method the reader is referred to Jofre et al. [38].

5.3 Discrete Navier-Stokes equations

Multiphase flows of immiscible fluids are governed by the continuity and Navier-Stokes equations in the variable-density incompressibility limit, written in divergence form as

$$\nabla \cdot \mathbf{u} = 0, \tag{5.6}$$

$$\frac{\partial(\rho \mathbf{u})}{\partial t} + \nabla \cdot (\rho \mathbf{u}\mathbf{u}) = -\nabla p + \nabla \cdot (\mu[\nabla \mathbf{u} + \nabla^T \mathbf{u}]) + \mathbf{S}, \tag{5.7}$$

where $\mathbf{u}$, p and $\mathbf{S}$ represent velocity, pressure and a general source term, e.g., gravitational acceleration, $\rho\mathbf{g}$, and surface tension, $\mathbf{T}_\sigma$. Additionally, density, ρ, and dynamic viscosity, μ, are interpolated from the properties of each fluid k by means of the fluid-volume fraction values, written as

$$\rho = \sum_k C_k \rho_k \quad \text{and} \quad \mu = \sum_k C_k \mu_k. \tag{5.8}$$

In general, the finite-volume spatial discretization of Eqs. 5.6 and 5.7 may be written, using discrete matrix operators, as

$$\mathbf{M}\mathbf{u} = \mathbf{0}, \tag{5.9}$$

$$\mathbf{\Omega}\frac{d(\rho \mathbf{u})}{dt} + \mathbf{C}(\rho \mathbf{u})\mathbf{u} + \mathbf{G}\mathbf{p} + \mathbf{D}(\mu)\mathbf{u} + \mathbf{\Omega}\mathbf{S} = \mathbf{0}, \tag{5.10}$$

where $\mathbf{u}$, $\mathbf{p}$ and $\mathbf{S}$ are the vectors of velocities, pressures and source terms. The diagonal matrix $\mathbf{\Omega}$ describes the volume of cells, matrices $\mathbf{C}(\rho\mathbf{u})$ and $\mathbf{D}(\mu)$ are the convective and diffusive operators, and matrices $\mathbf{G}$ and $\mathbf{M}$ represent the gradient and divergence operators, respectively.

At this point, like Verstappen and Veldman [11] propose, the discrete conservation properties may be easily analyzed if the symmetries of these matrices are studied. Therefore, kinetic energy is conserved if and only if the discrete convective operator is skew-symmetric, i.e., the transpose of the matrix is also its negative, $\mathbf{C}(\rho\mathbf{u}) =$

$-\mathbf{C}(\rho\mathbf{u})^*$, and if the negative conjugate transpose of the discrete gradient operator is equal to the divergence operator, $\mathbf{M} = -\mathbf{G}^*$. On the other hand, the diffusive operator must be symmetric and positive-definite in order to be dissipative, i.e., the matrix is equal to its transpose $\mathbf{D}(\mu) = \mathbf{D}(\mu)^*$, and $\mathbf{z}^*\mathbf{D}(\mu)\mathbf{z} > \mathbf{0}$ for all nonzero $\mathbf{z}$.

5.3.1 Collocated mesh scheme

Collocated mesh discretizations, independently of the time integration chosen, calculate velocity and pressure at centers of cells, while require specific interpolations for some variables and special evaluations of mass fluxes at cell faces, in order to minimize the kinetic energy error and exactly conserve mass, respectively. In particular, the collocated scheme presented in this work solves the velocity-pressure coupling of the momentum equation, Eq. 5.7, by means of a classical fractional step projection method along with a first-order explicit time advancement —higher order temporal schemes can be used, but for clarity the first-order one is chosen—, written as

$$\rho^{n+1}\mathbf{u}^{n+1} - \rho^{n+1}\mathbf{u}^{p} = -\Delta t \nabla p^{n+1}, \tag{5.11}$$

$$\rho^{n+1}\mathbf{u}^{p} = \rho^{n}\mathbf{u}^{n} - \Delta t \left[\nabla\cdot(\rho^{n}\mathbf{u}^{n}\mathbf{u}^{n}) - \nabla\cdot(\mu^{n}[\nabla\mathbf{u}^{n} + \nabla^{T}\mathbf{u}^{n}]) - \mathbf{S}^{n+1}\right], \tag{5.12}$$

where the superscript n refers to time instant, $\mathbf{u}^p$ is the predictor velocity, and Δt is the time step.

First, the predictor discrete velocity is obtained by dividing Eq. 5.12 by density, ρ^{n+1}, integrating over a cell c and applying the divergence theorem to its bordering faces, $f \in F(c)$, giving

$$\begin{aligned}\mathbf{u}_c^{p} =& \frac{\rho_c^{n}\mathbf{u}_c^{n}}{\rho_c^{n+1}} - \frac{\Delta t}{\rho_c^{n+1}V_c}\sum_{f\in F(c)}\boldsymbol{\phi}_f^{n}\hat{M}_f^{n} \\ &+ \frac{\Delta t}{\rho_c^{n+1}V_c}\left[\sum_{f\in F(c)}\mu_f^{n}\left[(\mathbf{u}_{nb}^{n} - \mathbf{u}_c^{n})\frac{A_f}{\delta d_f} + \nabla^{T}\mathbf{u}_f^{n}\cdot\hat{\mathbf{n}}_f A_f\right]\right] + \frac{\Delta t}{\rho_c^{n+1}V_c}\mathbf{S}_c^{n+1}V_c,\end{aligned} \tag{5.13}$$

where V_c is the volume of cell c, $\boldsymbol{\phi}_f$ is the convected velocity at face f, $\hat{M}_f$, $\hat{\mathbf{n}}_f$ and A_f are the outward mass flux, the normal outward unit vector and the surface of face f, respectively, subscript nb refers to the neighbor cell sharing a face and length δd_f is the normal-projected distance between the centroids of cells c and nb; see Fig. 5.1.

Next, dividing Eq. 5.11 by density, ρ^{n+1}, multiplying by the divergence operator, applying the incompressibility condition, Eq. 5.6, and discretizing over a cell c, yields the discrete Poisson's pressure equation

$$\sum_{f\in F(c)}\frac{\hat{M}_f^{p}}{\rho_f^{n+1}} = \Delta t\sum_{f\in F(c)}\frac{1}{\rho_f^{n+1}}(p_{nb}^{n+1} - p_c^{n+1})\frac{A_f}{\delta d_f}, \tag{5.14}$$

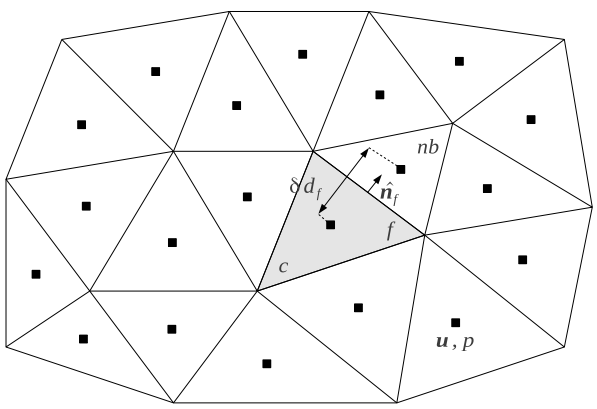


Figure 5.1: Arrangement of variables and notation for the collocated scheme on a 2-D unstructured mesh. The schematic representation shows the collocated position of velocity, $\mathbf{u}$, and pressure, p. The cell c where the discretization is analyzed is shown in gray, with an example of a face f and its corresponding neighbor cell nb, normal outward unit vector $\hat{\mathbf{n}}_f$ and distance δd_f between centroids.

which solves the pressure field at time instant $n+1$. Following the obtention of this p^{n+1} field, $\mathbf{u}^{n+1}$ results from discretizing Eq. 5.11 over a cell c as

$$\mathbf{u}_c^{n+1} = \mathbf{u}_c^p - \frac{\Delta t}{\rho_c^{n+1} V_c} \sum_{f \in F(c)} p_f^{n+1} \hat{\mathbf{n}}_f A_f, \tag{5.15}$$

where p_f is the pressure interpolated to face f.

Notice that the specific interpolations for $\boldsymbol{\phi}_f^n$, $\hat{M}_f^p$, ρ_f^{n+1}, $\mathbf{u}_f^p$, and p_f^{n+1} have not been explained yet. Therefore, in order to fulfill the skew-symmetric requirement of the discrete convective operator, the convected velocity at face f is evaluated by means of a symmetry-preserving scheme [11], written as

$$\boldsymbol{\phi}_f^n = \frac{1}{2}(\mathbf{u}_c^n + \mathbf{u}_{nb}^n). \tag{5.16}$$

Moreover, the predictor mass flux, density, predictor velocity and pressure at face f are calculated as

$$\hat{M}_f^p = \rho_f^{n+1} \mathbf{u}_f^p \cdot \hat{\mathbf{n}}_f A_f, \quad \rho_f^{n+1} = \frac{1}{2}(\rho_c^{n+1} + \rho_{nb}^{n+1}), \quad \mathbf{u}_f^p = \frac{1}{2}(\mathbf{u}_c^p + \mathbf{u}_{nb}^p), \tag{5.17}$$

$$p_f^{n+1} = \frac{1}{2}(p_c^{n+1} + p_{nb}^{n+1}), \tag{5.18}$$

minimizing, as it will be demonstrated in Sec. 5.4, the kinetic energy conservation error.

Finally, the evaluation of the mass flux at face f, $\hat{M}_f^{n+1}$, needs to be studied in detail in order to exactly conserve mass. Thus, taking again the divergence of Eq. 5.11 and discretizing over a cell c, gives

$$\sum_{f\in F(c)} \hat{M}_f^{n+1} - \sum_{f\in F(c)} \hat{M}_f^{p} = -\Delta t \sum_{f\in F(c)} (p_{nb}^{n+1} - p_c^{n+1}) \frac{A_f}{\delta d_f}, \tag{5.19}$$

which may be arranged in the following form

$$\sum_{f\in F(c)} \left[\hat{M}_f^{n+1} - \hat{M}_f^{p} + \Delta t (p_{nb}^{n+1} - p_c^{n+1}) \frac{A_f}{\delta d_f} \right] = 0. \tag{5.20}$$

Next, if for each face f the term between brackets is equalized to zero — it is a more restrictive condition, but at the same time provides an easier formulation —, the mass flux at a face f may be expressed as

$$\hat{M}_f^{n+1} = \hat{M}_f^{p} - \Delta t (p_{nb}^{n+1} - p_c^{n+1}) \frac{A_f}{\delta d_f}. \tag{5.21}$$

At this point, if the predictor mass flux is evaluated by means of Eq. 5.17 and $\mathbf{u}^p$ is substituted using Eq. 5.15, Eq. 5.21 may be rewritten as

$$\begin{aligned} \hat{M}_f^{n+1} &= \rho_f^{n+1} \frac{1}{2} (\mathbf{u}_c^{n+1} + \mathbf{u}_{nb}^{n+1}) \cdot \hat{\mathbf{n}}_f A_f - \Delta t (p_{nb}^{n+1} - p_c^{n+1}) \frac{A_f}{\delta d_f} \\ &+ \frac{\rho_f^{n+1} \Delta t}{2} \left[\frac{1}{\rho_c^{n+1} V_c} \sum_{f\in F(c)} p_f^{n+1} \hat{\mathbf{n}}_f A_f + \frac{1}{\rho_{nb}^{n+1} V_{nb}} \sum_{f\in F(nb)} p_f^{n+1} \hat{\mathbf{n}}_f A_f \right] \cdot \hat{\mathbf{n}}_f A_f. \end{aligned} \tag{5.22}$$

5.3.2 Staggered mesh scheme

The staggered mesh scheme calculates pressure and other scalar quantities at cell centers, while mass fluxes are distributed to cell faces. Each face stores only the mass flux, therefore, the cell-centered velocity vector has to be recovered from face normal values. This recovery of velocity vector from face normal values is not unique and it is a defining characteristic of each staggered mesh scheme, leading to different properties for the solution.

In particular, this work extends the scheme developed by Perot [10], which is suitable for solving incompressible flows on unstructured meshes, to flows having

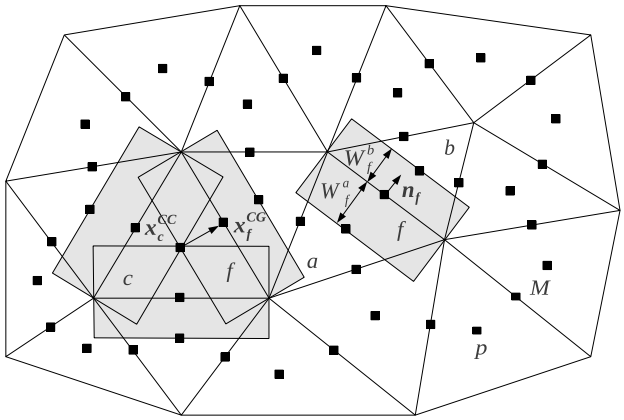


Figure 5.2: Arrangement of variables and notation for the staggered scheme on a 2-D unstructured mesh. The schematic representation shows the staggered position of the mass flux, M, and the location of pressure at a center of a cell, p. The face f and its neighboring cells a and b, where the cell-to-face operator is explained, are shown together with the distances W_f^a and W_f^b. On the other hand, the face-to-cell operator is shown by representing a cell c and an example of face f where the interpolation is explained by showing the distance $\mathbf{x}_f^{CG} - \mathbf{x}_c^{CC}$.

fluids with variable density. For this purpose, some preliminary remarks are needed. First, face-centered control volumes are defined for each face f as $V_f = (W_f^a + W_f^b)A_f$, where W_f is the distance between the face circumcenter and each of the circumcenters of the neighbor cells that share the face, while A_f is the surface of the considered face; see Fig. 5.2. Second, the convective, diffusive and source terms are calculated at centers of cells as non-volumetric quantities and, later, interpolated to faces using W_f.

Thereby, the staggered discrete form of the momentum equation is given by the integration of Eq. 5.7 over the control volume of a face f, taking a dot product with the face normal vector, $\mathbf{n}_f$, and solving the velocity-pressure coupling by means of a classical fractional step method along with a first-order explicit time integration —higher order temporal schemes can be also used—, resulting in

$$M_f^{n+1} = M_f^p - \Delta t(p_b^{n+1} - p_a^{n+1})\frac{A_f}{(W_f^a + W_f^b)}, \tag{5.23}$$

$$M_f^p = M_f^n - \Delta t\left[W_f^a(\mathbf{c}_a^n - \mathbf{d}_a^n - \mathbf{s}_a^{n+1}) + W_f^b(\mathbf{c}_b^n - \mathbf{d}_b^n - \mathbf{s}_b^{n+1})\right]\cdot\mathbf{n}_f\frac{A_f}{(W_f^a + W_f^b)}, \tag{5.24}$$

where subscripts a and b refer to the two cells adjacent to face f and $\mathbf{c}$, $\mathbf{d}$ and $\mathbf{s}$ are the

non-volumetric cell-centered discretizations of the convective, diffusive and source terms, evaluated for each cell c as

$$\mathbf{c}_c^n = \frac{1}{V_c}\sum_{f\in F(c)} \boldsymbol{\phi}_f^n \hat{M}_f^n, \quad \mathbf{s}_c^{n+1} = \frac{1}{V_c}\mathbf{S}_c^{n+1}V_c, \tag{5.25}$$

$$\mathbf{d}_c^n = \frac{1}{V_c}\sum_{f\in F(c)} \mu_f^n \left[(\mathbf{u}_{nb}^n - \mathbf{u}_c^n)\frac{A_f}{\delta d_f} + \nabla^T \mathbf{u}_f^n \cdot \hat{\mathbf{n}}_f A_f\right],$$

considering that the convected velocity at face f, $\boldsymbol{\phi}_f$, is evaluated by a symmetry-preserving scheme [11] and length δd_f is once again the distance between the nodes of the cells adjacent to face f.

Next, dividing Eq. 5.23 by face density, ρ_f^{n+1}, summing over the bordering faces of cell c and making use of the incompressibility constraint, results in the already presented discrete Poisson's pressure equation, Eq. 5.14. However, no interpolation for the predictor mass flux is needed in this case, since it is given directly by Eq. 5.24. In addition, δd_f is now the distance between the circumcenters of the two cells sharing face f. Analogously, once the solution of Eq. 5.14 is calculated, Eq. 5.23 is used to obtain the face mass fluxes at instant $n+1$, M_f^{n+1}.

Finally, the staggered mesh scheme discretizes mass fluxes in time, thus, velocities at centers of cells need to be interpolated from face normal values. In this way, applying the divergence theorem for a cell c to the product of position, $\mathbf{r}$, and momentum, $\rho\mathbf{u}$, gives

$$\int_{\Omega_c} \rho\mathbf{u}\, dV + \int_{\Omega_c} \mathbf{r}(\nabla\cdot(\rho\mathbf{u}))\, dV = \sum_{f\in F(c)} \int_{\partial\Omega_f} \mathbf{r}(\rho\mathbf{u})\cdot\hat{\mathbf{n}}_f dA, \tag{5.26}$$

where $\mathbf{r} = \mathbf{x} - \mathbf{x}_0$ is the position vector from the circumcenter of cell c. Hence, if a first-order approximation of the momentum field (constant $\rho\mathbf{u}$) is assumed, Eq. 5.26 is rewritten as

$$\mathbf{u}_c = \frac{1}{\rho_c V_c}\sum_{f\in F(c)} \mathbf{r}_f^c \hat{M}_f, \tag{5.27}$$

being $\mathbf{r}_f^c = \mathbf{x}_f^{CG} - \mathbf{x}_c^{CC}$ the vector from the circumcenter of cell c, $\mathbf{x}_c^{CC}$, to the centroid of face f, $\mathbf{x}_f^{CG}$.

5.4 Conservation properties

The Navier-Stokes equations are derived specifically for the conservation of momentum, thus, most discretizations found in the scientific literature conserve this property.

On the contrary, the conservation of secondary derived quantities, such as kinetic energy, entropy and vorticity —which are not directly unknowns of the numerical system and, in consequence, cannot be directly imposed during the construction of the numerical methods— is not always considered, even though their remarkable importance in the physics of the problems to be solved. Hence, this section develops and analyzes the conservation of mass, momentum and, more importantly, of kinetic energy for the collocated and staggered schemes previously presented.

5.4.1 Mass conservation

Global mass conservation invokes the integral of Eq. 5.6 over the whole domain, Ω. Thus, if the entire integral is transformed to a summation of integrals for each control volume that form the domain, $c \in \Omega$, the following expression is obtained

$$\int_\Omega \nabla\cdot \mathbf{u}\, dV = \sum_{c\in\Omega}\int_{\Omega_c} \nabla\cdot \mathbf{u}\, dV = \sum_{c\in\Omega}\sum_{f\in F(c)} \hat{U}_f A_f. \tag{5.28}$$

Defining the normal face velocity, U_f, as the mass flux at a face, M_f, divided by face density, ρ_f, and area, A_f, rewrites Eq. 5.28 as

$$\int_\Omega \nabla\cdot \mathbf{u}\, dV = \sum_{c\in\Omega}\sum_{f\in F(c)} \hat{U}_f A_f = \sum_{c\in\Omega}\sum_{f\in F(c)} \frac{\hat{M}_f}{\rho_f}. \tag{5.29}$$

In the collocated case, a special definition for mass fluxes at faces, Eq. 5.22, has been developed in order to exactly conserve mass in each cell c. Thus, the local conservation of mass for the collocated scheme is demonstrated by dividing Eq. 5.19 by face density, rearranging terms and making use of Eq. 5.14, giving

$$\sum_{f\in F(c)} \frac{\hat{M}_f^{n+1}}{\rho_f^{n+1}} = \sum_{f\in F(c)} \left[\frac{\hat{M}_f^{p}}{\rho_f^{n+1}} - \frac{\Delta t}{\rho_f^{n+1}} (p_{nb}^{n+1} - p_c^{n+1}) \frac{A_f}{\delta d_f} \right] = 0. \tag{5.30}$$

On the contrary, for the staggered case no interpolation of mass fluxes at faces is needed, since they are directly calculated there. Hence, dividing Eq. 5.23 by face density, summing over the faces of a cell, reorganizing terms and making use of the staggered version of the discrete Poisson's equation, Eq. 5.14, results in the staggered local conservation of mass, shown as

$$\sum_{f\in F(c)} \frac{\hat{M}_f^{n+1}}{\rho_f^{n+1}} = \sum_{f\in F(c)} \left[\frac{\hat{M}_f^{p}}{\rho_f^{n+1}} - \frac{\Delta t}{\rho_f^{n+1}} (p_{nb}^{n+1} - p_c^{n+1}) \frac{A_f}{(W_f^a + W_f^b)} \right] = 0. \tag{5.31}$$

Summarizing, in both cases mass is locally conserved, consequently, the global mass conservation, Eq. 5.28, equals zero, which is expressed as

$$\int_{\Omega} \nabla \cdot \mathbf{u} \, dV = \sum_{c \in \Omega} \int_{\Omega_c} \nabla \cdot \mathbf{u} \, dV = \sum_{c \in \Omega} \sum_{f \in F(c)} \hat{U}_f A_f = \sum_{c \in \Omega} \sum_{f \in F(c)} \frac{\hat{M}_f}{\rho_f} = 0. \tag{5.32}$$

5.4.2 Momentum conservation

The conservation of momentum is a straightforward consequence of writing the equations in divergence form. However, a proof of conservation of momentum may be natural for collocated schemes, but not obvious for staggered discretizations on unstructured meshes. The inherent difficulty is due to the fact that the velocity vector is not a primary variable for staggered schemes.

Collocated momentum conservation

The total conservation of momentum is obtained by integrating Eq. 5.7 over the entire domain, which is transformed to a summation of integrals for each control volume that form the domain and converted to surface integrals by applying the divergence theorem, giving

$$\begin{aligned} \sum_{c \in \Omega} \frac{d(\rho_c \mathbf{u}_c)}{dt} V_c + \sum_{c \in \Omega} \sum_{f \in F(c)} \boldsymbol{\phi}_f \hat{M}_f &= - \sum_{c \in \Omega} \sum_{f \in F(c)} p_f \hat{\mathbf{n}}_f A_f \\ + \sum_{c \in \Omega} \sum_{f \in F(c)} \mu_f \left[(\mathbf{u}_{nb} - \mathbf{u}_c) \frac{A_f}{\delta d_f} + \nabla^T \mathbf{u}_f \cdot \hat{\mathbf{n}}_f A_f \right] &+ \sum_{c \in \Omega} \mathbf{S}_c V_c. \end{aligned} \tag{5.33}$$

Notice that $\hat{M}_f$, $\hat{\mathbf{n}}_f$ and $(\mathbf{u}_{nb} - \mathbf{u}_c)$ are quantities that present equal values but with different sign when evaluating them at a face f from two adjacent interior cells. In this way, interior fluxes cancel out and Eq. 5.33 is evaluated as the summation over boundary faces, $f \in F(\partial\Omega)$, written as

$$\begin{aligned} \sum_{c \in \Omega} \frac{d(\rho_c \mathbf{u}_c)}{dt} V_c + \sum_{f \in F(\partial\Omega)} \boldsymbol{\phi}_f \hat{M}_f &= - \sum_{f \in F(\partial\Omega)} p_f \hat{\mathbf{n}}_f A_f \\ + \sum_{f \in F(\partial\Omega)} \mu_f \left[(\mathbf{u}_f - \mathbf{u}_a) \frac{A_f}{\delta d_f} + \nabla^T \mathbf{u}_f \cdot \hat{\mathbf{n}}_f A_f \right] &+ \sum_{c \in \Omega} \mathbf{S}_c V_c, \end{aligned} \tag{5.34}$$

which states that the change in momentum is due to the fluxes through the boundary of the domain and the source terms.

Staggered momentum conservation

The primary quantity in staggered mesh schemes is the mass fluxes at faces. Thus, integrating Eq. 5.7 over the control volume of a face f, as explained in detail in Sec. 5.3.2, and taking the dot product with its normal unit vector, $\mathbf{n}_f$, gives the discretized momentum equation for the mass flux at faces, M_f, written as

$$\begin{aligned}(W_f^a + W_f^b)\frac{dM_f}{dt} + (W_f^a\mathbf{c}_a + W_f^b\mathbf{c}_b)A_f\cdot\mathbf{n}_f = -(p_b - p_a)A_f \\ + (W_f^a\mathbf{d}_a + W_f^b\mathbf{d}_b)A_f\cdot\mathbf{n}_f + (W_f^a\mathbf{s}_a + W_f^b\mathbf{s}_b)A_f\cdot\mathbf{n}_f.\end{aligned} \tag{5.35}$$

Next, the discrete staggered conservation of momentum is obtained if Eq. 5.35 is multiplied by the normal unit vector of face f, $\mathbf{n}_f$, and is summed over all the faces of the domain, $f \in F(\Omega)$, giving the following equation

$$\begin{aligned}&\sum_{f\in F(\Omega)}(W_f^a + W_f^b)\frac{dM_f}{dt}\mathbf{n}_f + \sum_{f\in F(\Omega)}(W_f^a\mathbf{c}_a + W_f^b\mathbf{c}_b)A_f\cdot\mathbf{n}_f\mathbf{n}_f \\ &= -\sum_{f\in F(\Omega)}(p_b - p_a)A_f\mathbf{n}_f + \sum_{f\in F(\Omega)}(W_f^a\mathbf{d}_a + W_f^b\mathbf{d}_b)A_f\cdot\mathbf{n}_f\mathbf{n}_f \\ &+ \sum_{f\in F(\Omega)}(W_f^a\mathbf{s}_a + W_f^b\mathbf{s}_b)A_f\cdot\mathbf{n}_f\mathbf{n}_f,\end{aligned} \tag{5.36}$$

then, the goal is to recast this equation as an equation for velocities at centers of cells.

First, the summation over faces of the time derivative in Eq. 5.36 is recast as a summation over cells, developed as

$$\begin{aligned}&\sum_{f\in F(\Omega)}(W_f^a + W_f^b)\frac{dM_f}{dt}\mathbf{n}_f = \frac{d}{dt}\left[\sum_{f\in F(\Omega)}(\mathbf{r}_f^a - \mathbf{r}_f^b)M_f\right] \\ &= \frac{d}{dt}\left[\sum_{c\in\Omega}\left[\frac{1}{\rho_c V_c}\sum_{f\in F(c)}\mathbf{r}_f^c\hat{M}_f\right]\rho_c V_c\right] = \sum_{c\in\Omega}\frac{d(\rho_c\mathbf{u}_c)}{dt}V_c,\end{aligned} \tag{5.37}$$

where the first equality is true, since for each face f the following expression applies

$$\begin{aligned}(W_f^a + W_f^b)\mathbf{n}_f &= \mathbf{x}_b^{CC} - \mathbf{x}_a^{CC} \\ &= (\mathbf{x}_f^{CG} - \mathbf{x}_a^{CC}) - (\mathbf{x}_f^{CG} - \mathbf{x}_b^{CC}) = \mathbf{r}_f^a - \mathbf{r}_f^b,\end{aligned} \tag{5.38}$$

while the second one corresponds to the transformation from face to cell summation, noticing that $\hat{M}_f^b = -\hat{M}_f^a$, and the third one is straightforward from Eq. 5.27.

Second, the summations over all faces of the domain for the convective and diffusive terms in Eq. 5.36 are equivalent to the summations over boundary faces, expressed as

$$\sum_{f\in F(\Omega)} (W_f^a \mathbf{c}_a + W_f^b \mathbf{c}_b) A_f \cdot \mathbf{n}_f \mathbf{n}_f = \sum_{c\in\Omega} \mathbf{c}_c \cdot \left[\sum_{f\in F(c)} \mathbf{n}_f \mathbf{n}_f W_f^c A_f \right] \tag{5.39}$$
$$= \sum_{c\in\Omega} \mathbf{c}_c \cdot \mathbf{I} V_c = \sum_{c\in\Omega} \mathbf{c}_c V_c = \sum_{c\in\Omega} \sum_{f\in F(c)} \boldsymbol{\phi}_f \hat{M}_f = \sum_{f\in F(\partial\Omega)} \boldsymbol{\phi}_f \hat{M}_f,$$

$$\sum_{f\in F(\Omega)} (W_f^a \mathbf{d}_a + W_f^b \mathbf{d}_b) A_f \cdot \mathbf{n}_f \mathbf{n}_f = \sum_{c\in\Omega} \mathbf{d}_c \cdot \left[\sum_{f\in F(c)} \mathbf{n}_f \mathbf{n}_f W_f^c A_f \right] \tag{5.40}$$
$$= \sum_{c\in\Omega} \mathbf{d}_c \cdot \mathbf{I} V_c = \sum_{c\in\Omega} \mathbf{d}_c V_c = \sum_{c\in\Omega} \sum_{f\in F(c)} \mu_f \left[(\mathbf{u}_{nb} - \mathbf{u}_c) \frac{A_f}{\delta d_f} + \nabla^T \mathbf{u}_f \cdot \hat{\mathbf{n}}_f A_f \right]$$
$$= \sum_{f\in F(\partial\Omega)} \mu_f \left[(\mathbf{u}_f - \mathbf{u}_a) \frac{A_f}{\delta d_f} + \nabla^T \mathbf{u}_f \cdot \hat{\mathbf{n}}_f A_f \right],$$

where the first term in brackets is a known geometric result of the divergence theorem and is equal to the identity tensor multiplied by the cell volume, $\mathbf{I}V_c$. Next, terms $\mathbf{c}_c$ and $\mathbf{d}_c$ are expanded using Eq. 5.25 and interior fluxes are canceled out exactly, leaving just fluxes through the boundary faces.

Third, source terms in Eq. 5.36 are converted to a summation over all the cells of the domain, written as

$$\sum_{f\in F(\Omega)} (W_f^a \mathbf{s}_a + W_f^b \mathbf{s}_b) A_f \cdot \mathbf{n}_f \mathbf{n}_f = \sum_{c\in\Omega} \mathbf{s}_c \cdot \left[\sum_{f\in F(c)} \mathbf{n}_f \mathbf{n}_f W_f^c A_f \right] \tag{5.41}$$
$$= \sum_{c\in\Omega} \mathbf{s}_c \cdot \mathbf{I} V_c = \sum_{c\in\Omega} \mathbf{s}_c V_c = \sum_{c\in\Omega} \mathbf{S}_c V_c.$$

Fourth, the pressure term in Eq. 5.36 can be straightforwardly rearranged as

$$\sum_{f\in F(\Omega)} (p_b - p_a) A_f \mathbf{n}_f = -\sum_{c\in\Omega} p_c \sum_{f\in F(c)} \hat{\mathbf{n}}_f A_f \tag{5.42}$$
$$+ \sum_{f\in F(\partial\Omega)} p_f \hat{\mathbf{n}}_f A_f = \sum_{f\in F(\partial\Omega)} p_f \hat{\mathbf{n}}_f A_f.$$

In summary, it is shown that Eq. 5.36 can be recast as Eq. 5.34 by using Eqs. 5.37 to 5.42. Hence, equivalently to the collocated scheme case, the change in momentum for the staggered discretization is due to the fluxes through the boundary of the domain and the source terms.

5.4.3 Kinetic energy conservation

The conservation of kinetic energy is an important property especially when solving turbulent flows, since energy is convected from the main flow into the large eddies, and from them into the next smaller ones, and so on until being dissipated by molecular forces in the smallest eddies. Hence, if no external sources are present, the rate of change of total kinetic energy is just determined by dissipation. Thus, discretization strategies with excessive numerical dissipation can alter the physics of a problem in a very important proportion.

In detail, the transport equation for kinetic energy is derived from the momentum equation, Eq. 5.7, by taking the velocity dot product and assuming incompressible fluid. In this way, the kinetic energy, $\frac{1}{2}\rho\mathbf{u}\cdot\mathbf{u}$, can be shown to obey the following transport equation

$$\begin{aligned}\frac{\partial(\frac{1}{2}\rho\mathbf{u}\cdot\mathbf{u})}{\partial t} + \nabla\cdot[\mathbf{u}(\frac{1}{2}\rho\mathbf{u}\cdot\mathbf{u})] &= -\nabla\cdot(p\mathbf{u}) + \nabla\cdot(\mu\mathbf{u}\times\boldsymbol{\omega}) \\ &- \mu\boldsymbol{\omega}\cdot\boldsymbol{\omega} + [2(\nabla\mu\times\boldsymbol{\omega}) + \nabla\mu\cdot\nabla\mathbf{u} + (\nabla\mu\cdot\nabla)\mathbf{u}]\cdot\mathbf{u} + \mathbf{S}\cdot\mathbf{u},\end{aligned} \tag{5.43}$$

where $\boldsymbol{\omega} = \nabla\times\mathbf{u}$ is the vorticity. The important characteristic of this equation is that it is conservative except for the non-divergence terms of the second line. Hence, in the absence of external forces and viscosity, the kinetic energy is simply redistributed but not created or destroyed. Similarly, discrete systems will be kinetic energy conservative if convective and pressure operators are shown to be conservative [39].

Collocated kinetic energy conservation

In order to investigate the conservation of kinetic energy for the collocated scheme, the momentum equation, Eq. 5.7, is discretized over the whole domain and multiplied by the velocity vector, $\mathbf{u}$. Then, the resulting equation can be transformed to a summation of surface integrals for each cell c, written as

$$\begin{aligned}&\sum_{c\in\Omega}\mathbf{u}_c\cdot\frac{d(\rho_c\mathbf{u}_c)}{dt}V_c + \sum_{c\in\Omega}\mathbf{u}_c\cdot\sum_{f\in F(c)}\boldsymbol{\phi}_f\hat{M}_f = -\sum_{c\in\Omega}\mathbf{u}_c\cdot\sum_{f\in F(c)}p_f\hat{\mathbf{n}}_f A_f \\ &+\sum_{c\in\Omega}\mathbf{u}_c\cdot\sum_{f\in F(c)}\mu_f\left[(\mathbf{u}_{nb}-\mathbf{u}_c)\frac{A_f}{\delta d_f} + \nabla^T\mathbf{u}_f\cdot\hat{\mathbf{n}}_f A_f\right] + \sum_{c\in\Omega}\mathbf{u}_c\cdot\mathbf{S}_c V_c,\end{aligned} \tag{5.44}$$

where, from left to right, the terms correspond to the values of time derivative, convection, pressure, diffusion and source. At this point, the detailed analysis of Eq. 5.44 is simplified by making use of the two identities presented in the Appendix, these involve combinations of interpolations and differentiation operators.

First, the convective term of Eq. 5.44 is transformed by specializing Eq. 5.64 to $\varphi = \mathbf{u}$, $\psi = \boldsymbol{\phi}$, $\chi = \rho$ and $Q_f = \hat{U}_f A_f$, then, using the continuity equation, Eq. 5.6, and canceling out equal terms, the convective expression can be rewritten as

$$\sum_{c\in\Omega} \mathbf{u}_c \cdot \sum_{f\in F(c)} \boldsymbol{\phi}_f \hat{M}_f = \sum_{c\in\Omega} \sum_{f\in F(c)} \frac{1}{4} \mathbf{u}_c \cdot (4\boldsymbol{\phi}_f \rho_f - \boldsymbol{\phi}_c \rho_c) \hat{U}_f A_f, \tag{5.45}$$

where $\boldsymbol{\phi}_f$ is evaluated as the semi-sum of the velocities of the two adjacent cells, i.e., using the symmetry-preserving convection scheme [11].

Second, if the pressure term in Eq. 5.44 is analyzed in a similar fashion, making use of Eq. 5.62, by taking $\varphi = \mathbf{u}$, $\psi = p$ and $Q_f = \hat{\mathbf{n}}_f A_f$, and Eq. 5.22 is used to simplify the expression, results in the following relation

$$\begin{aligned}\sum_{c\in\Omega} \mathbf{u}_c \cdot \sum_{f\in F(c)} p_f \hat{\mathbf{n}}_f A_f = \sum_{c\in\Omega} \sum_{f\in F(c)} \widehat{\mathbf{u}p} \cdot \hat{\mathbf{n}}_f A_f - \sum_{c\in\Omega} p_c \sum_{f\in F(c)} \frac{\delta t}{\rho_f} \left[(p_{nb} - p_c) \frac{A_f}{\delta d_f} \right] \\ + \sum_{c\in\Omega} p_c \sum_{f\in F(c)} \frac{\delta t}{2} \left[\frac{1}{\rho_c V_c} \sum_{f\in F(c)} p_f \hat{\mathbf{n}}_f A_f + \frac{1}{\rho_{nb} V_{nb}} \sum_{f\in F(nb)} p_f \hat{\mathbf{n}}_f A_f \right] \cdot \hat{\mathbf{n}}_f A_f.\end{aligned} \tag{5.46}$$

Finally, notice that interior fluxes in Eqs. 5.45 and 5.46 cancel out, thus, Eq. 5.44 can be rewritten as

$$\begin{aligned}&\sum_{c\in\Omega} \frac{d(\frac{1}{2}\rho_c \mathbf{u}_c \cdot \mathbf{u}_c)}{dt} V_c + \sum_{f\in F(\partial\Omega)} \frac{1}{4} \mathbf{u}_a \cdot (4\boldsymbol{\phi}_f \rho_f - \boldsymbol{\phi}_a \rho_a) \hat{U}_f A_f = \\ &- \sum_{f\in F(\partial\Omega)} \frac{1}{2} \left(\mathbf{u}_a p_f + \mathbf{u}_f p_a \right) \cdot \hat{\mathbf{n}}_f A_f + \sum_{c\in\Omega} p_c \sum_{f\in F(c)} \frac{\delta t}{\rho_f} \left[(p_{nb} - p_c) \frac{A_f}{\delta d_f} \right] \\ &- \sum_{c\in\Omega} p_c \sum_{f\in F(c)} \frac{\delta t}{2} \left[\frac{1}{\rho_c V_c} \sum_{f\in F(c)} p_f \hat{\mathbf{n}}_f A_f + \frac{1}{\rho_{nb} V_{nb}} \sum_{f\in F(nb)} p_f \hat{\mathbf{n}}_f A_f \right] \cdot \hat{\mathbf{n}}_f A_f \\ &+ \sum_{c\in\Omega} \mathbf{u}_c \cdot \sum_{f\in F(c)} \mu_f \left[(\mathbf{u}_{nb} - \mathbf{u}_c) \frac{A_f}{\delta d_f} + \nabla^T \mathbf{u}_f \cdot \hat{\mathbf{n}}_f A_f \right] + \sum_{c\in\Omega} \mathbf{u}_c \cdot \mathbf{S}_c V_c,\end{aligned} \tag{5.47}$$

which states that, in the absence of viscosity ($\mu = 0$) and source terms, the change in kinetic energy is due to the fluxes through the boundary of the domain and a kinetic energy error from the pressure term. This error term arises from the different pressure gradient evaluations between Eqs. 5.15 and 5.19, necessary to evaluate velocities at centers of cells and mass fluxes at time $n + 1$, respectively. Notice that if first-order interpolations, i.e., semi-summed variables from adjacent cells, and a

symmetry-preserving convection scheme are used, the kinetic energy conservation error is minimized.

It is of great importance to evaluate the scaling order of this kinetic energy pressure error, since it can not be eliminated. Thus, the error is easily analyzed simplifying it for each individual face f, written as

$$\delta t A_f \left[\frac{(p_{nb} - p_c)}{\rho_f \delta d_f} - \frac{1}{2} \left[\sum_{f \in F(c)} \frac{p_f \hat{\mathbf{n}}_f A_f}{\rho_c V_c} + \sum_{f \in F(nb)} \frac{p_f \hat{\mathbf{n}}_f A_f}{\rho_{nb} V_{nb}} \right] \cdot \hat{\mathbf{n}}_f \right], \tag{5.48}$$

resulting that the whole term depends on density and is multiplied by time step, δt, and face surface, A_f. Hence, the pressure error is proportional to $\Delta\rho$, while spatially scaled as $\mathcal{O}(\Delta h^2)$ and temporally scaled as $\mathcal{O}(\Delta t)$, although it can be reduced through the use of different temporal integration schemes, $\mathcal{O}(\Delta t^m)$, as proposed by Felten and Lund [12] and studied by Fishpool and Leschziner [40].

This result can be related to the symmetries of discrete operators in the following way: (1) the convective term in Eq. 5.47 presents no kinetic energy error, since the convection scheme has been chosen to make the convective operator skew-symmetric; (2) the different pressure gradient evaluations between Eqs. 5.15 and 5.19 do not respect the relation $\mathbf{M} = -\mathbf{G}^*$, therefore, a pressure gradient error term arises in Eq. 5.47.

Staggered kinetic energy conservation

The staggered kinetic energy equation starts from the staggered momentum equation, Eq. 5.36. First, Eqs. 5.37 to 5.42 are used to recast the summation over faces as a summation over cells and, second, the resulting equation is multiplied by velocity, $\mathbf{u}$. In this way, the staggered kinetic energy equation is shown to obey the same equation as in the collocated case, Eq. 5.44.

Next, the convective term is converted to flux form as done for the collocated case, Eq. 5.45, while the pressure term is analyzed by specializing Eq. 5.62 to $\varphi = \mathbf{u}$, $\psi = p$, $Q_f = \hat{\mathbf{n}}_f A_f$ and noticing that a special definition for mass fluxes at faces is not needed, giving

$$\sum_{c \in \Omega} \mathbf{u}_c \cdot \sum_{f \in F(c)} p_f \hat{\mathbf{n}}_f A_f = \sum_{c \in \Omega} \sum_{f \in F(c)} \widehat{\mathbf{u}p} \cdot \hat{\mathbf{n}}_f A_f. \tag{5.49}$$

Finally, knowing that interior fluxes cancel out, Eq. 5.44 is rewritten as

$$\begin{aligned}
&\sum_{c\in\Omega}\frac{d(\frac{1}{2}\rho_c\mathbf{u}_c\cdot\mathbf{u}_c)}{dt}V_c+\sum_{f\in F(\partial\Omega)}\frac{1}{4}\mathbf{u}_a\cdot(4\boldsymbol{\phi}_f\rho_f-\boldsymbol{\phi}_a\rho_a)\hat{U}_fA_f=\\
&-\sum_{f\in F(\partial\Omega)}\frac{1}{2}\left(\mathbf{u}_ap_f+\mathbf{u}_fp_a\right)\cdot\hat{\mathbf{n}}_fA_f\\
&+\sum_{c\in\Omega}\mathbf{u}_c\cdot\sum_{f\in F(c)}\mu_f\left[(\mathbf{u}_{nb}-\mathbf{u}_c)\frac{A_f}{\delta d_f}+\nabla^T\mathbf{u}_f\cdot\hat{\mathbf{n}}_fA_f\right]+\sum_{c\in\Omega}\mathbf{u}_c\cdot\mathbf{S}_cV_c,
\end{aligned}\tag{5.50}$$

which states that, in the absence of viscosity ($\mu = 0$) and source terms, the change in kinetic energy is due solely to the fluxes through the boundary of the domain.

In this case, the two discrete operator properties needed to conserve kinetic energy are fulfilled: (1) the convective term is evaluated by a symmetry-preserving convection scheme, thus, making the discrete convective operator skew-symmetric; (2) the mass fluxes at faces do not need a special definition, since it is the primary quantity, then, the divergence-gradient relation respects the $\mathbf{M} = -\mathbf{G}^*$ condition.

5.5 Conservation and accuracy tests

Three different problems will be solved to test the conservation properties and accuracy of the unstructured mesh schemes previously presented. First, the conservation properties will be analyzed by solving a three-dimensional vortex with zero mass flux at the boundaries. Second, an accuracy assessment will be presented using an exact sinusoidal function. Finally, the schemes will be tested by calculating the drag force on a spherical bubble in a turbulent pipe flow.

5.5.1 Three-dimensional vortex

The conservation properties, studied theoretically in Sec. 5.4, are verified numerically by solving a three-dimensional vortex. This problem is chosen since it is inherently unsteady but at the same time has zero net mass flux at the boundaries.

The spatially periodic set of 2×2 three-dimensional vortices shown in Fig. 5.3 are described by

$$\begin{aligned}
u &= -Asin(kx)cos(ky)e^{-2k^2\nu t},\\
v &= Acos(kx)sin(ky)e^{-2k^2\nu t},\\
w &= -A,
\end{aligned}\tag{5.51}$$

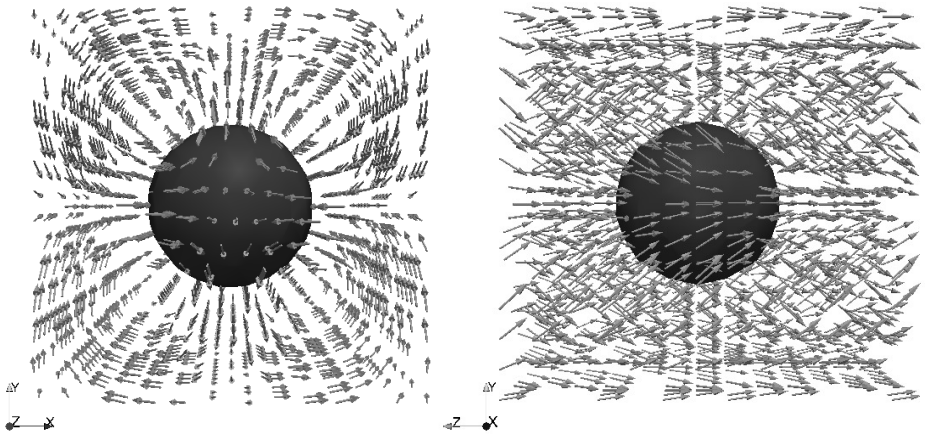


Figure 5.3: Frontal (xy-plane) and lateral (yz-plane) views of the three-dimensional vortex test. The velocity field is displayed in light gray, while the different-density sphere is shown in dark gray.

where $A = 1.0 \times 10^{-3}\ m/s$ is the velocity amplitude, $k = 1$ is the wave number and $\nu = 0$ is the kinematic viscosity, which is set to zero to eliminate the effects of the diffusive term.

The vortex is solved in a box of side $2\pi \times 2\pi \times 2\pi$ meshed by means of 66000 triangular prisms that correspond to a mesh size of $h = 0.2$. In detail, the 3-D mesh is generated by extruding a 2-D grid, discretized in 2200 triangles, 30 times with a constant step. Moreover, the box is filled with two different fluids, one with density $\rho_1 = 1\ kg/m^3$ that occupies the entire cube except for a sphere of radius $R = \pi/2$, fixed in the center, that corresponds to the other fluid, which may present different densities $\rho_2 = 10, 100, 1000\ kg/m^3$. A constant time step of $\Delta t = 1.0 \times 10^3\ s$ is used. Besides, boundaries X and Y are considered slip walls, while periodic conditions are set for Z ones.

First, mass and total momentum for each mesh scheme and sphere's density, ρ_2, are calculated at every time step using Eqs. 5.32 and 5.34. The results corroborate that both collocated and staggered schemes conserve mass and total momentum, as theoretically expected, since there is no net flux across the domain boundaries. Notice that in this problem, the x and y velocity components are symmetric about the axis of the vortex and its z-component is periodic, thus, the inital mass and total momentum are zero and they remain invariable through the test.

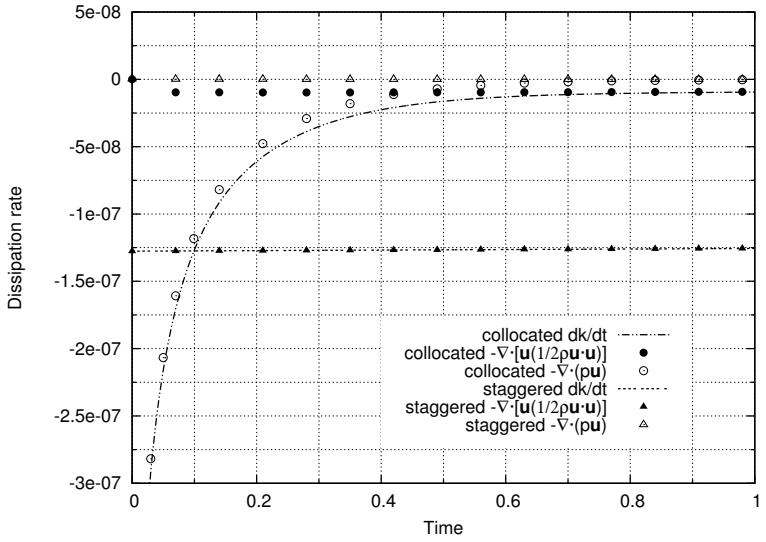


Figure 5.4: Rate of change of kinetic energy, upwind-convection and pressure versus time, using both collocated and staggered mesh schemes with $\Delta\rho = 10$.

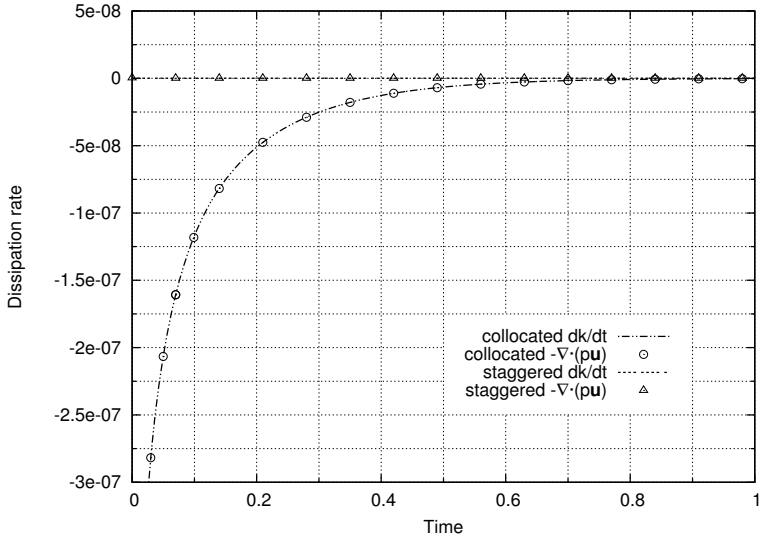


Figure 5.5: Rate of change of kinetic energy and pressure versus time, using the symmetry-preserving collocated and staggered mesh schemes with $\Delta\rho = 10$.

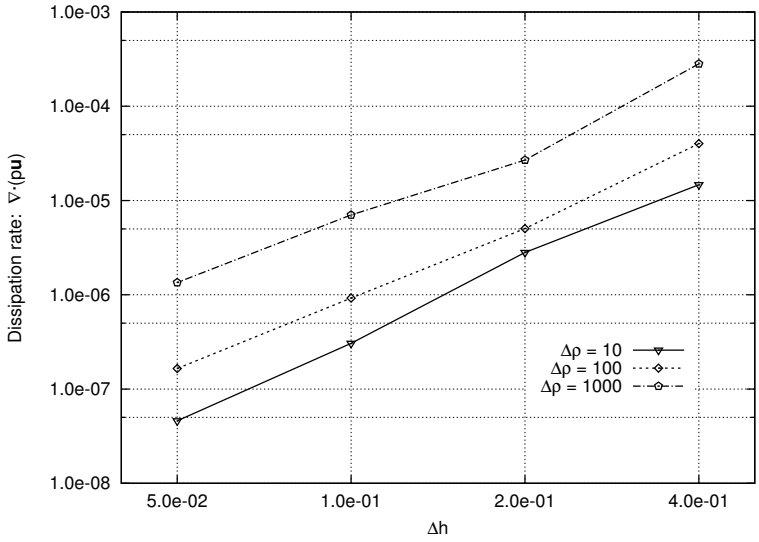


Figure 5.6: Error in kinetic energy for the collocated scheme caused by pressure versus mesh size for different density ratios.

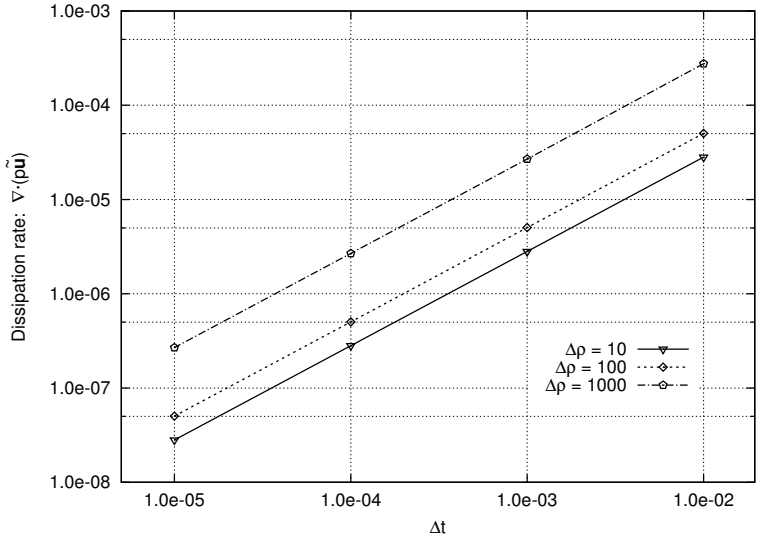


Figure 5.7: Error in kinetic energy for the collocated scheme caused by pressure versus time step for different density ratios.

Furthermore, this test is really appropriate to study the conservation of kinetic energy since viscosity is set to zero, there is no net mass flux at the boundaries and no source terms exist. Under these conditions, the continuous transport equation for kinetic energy, Eq. 5.43, determines that the rate of change of total kinetic energy is zero, $\partial k/\partial t = \partial(\frac{1}{2}\rho \mathbf{u}\cdot\mathbf{u})/\partial t = 0$. Hence, any existing variation of kinetic energy is due to an improper discretization. In this way, the rate of change of kinetic energy, dk/dt, convection using symmetry-preserving [11] and upwind [41] schemes, $\nabla\cdot[\mathbf{u}(\frac{1}{2}\rho\mathbf{u}\cdot\mathbf{u})]$, and pressure, $\nabla\cdot(p\mathbf{u})$, for each mesh scheme and sphere's density, ρ_2, are calculated at every time step using Eq. 5.44.

On the one hand, Fig. 5.4 shows that the use of an upwind convection scheme produces an artificial kinetic energy dissipation. Particularly for case $\Delta\rho = 10$, the collocated discretization produces an upwind-convection error of magnitude 10^{-8}, while the staggered scheme produces one of order 10^{-7}. This difference in the error of convection between mesh schemes is due to the different velocity fields obtained by each one. On the other hand, Fig. 5.5 confirms that the use of the symmetry-preserving convection scheme turns out in a zero contribution to the kinetic energy equation, since if any nonphysical kinetic energy variation exists is solely determined by the pressure term. Furthermore, Fig. 5.5 also demonstrates that the staggered scheme presents a zero pressure contribution to the kinetic energy variation, on the contrary, the collocated scheme presents a nonzero value.

It is interesting to study numerically the scaling order of this error in kinetic energy caused by pressure which, as previously analyzed in Sec. 5.4.3, is intrinsic to the collocated mesh scheme and depends on mesh size and time integration. First, the comparison between the pressure error term and mesh size is evaluated by solving the vortex, for the three different density ratios, on four succesively refined unstructured meshes ($h = 0.4$ to $h = 0.05$) with a fixed time step $\Delta t = 1.0\times10^{-3}\ s$. Second, the relation between the pressure error term and time integration is analyzed by solving the same test on the $h = 0.2$ mesh with four different time steps (1.0×10^{-2} to 5×10^{-5}) and their corresponding relative velocities, $\tilde{\mathbf{u}}$ (1.0×10^{-2} to $5\times10^{-5}\ m/s$); i.e., $\nabla\cdot(p\mathbf{u})$ is the time variation of kinetic energy due to pressure, hence, velocity must be time-proportional, $\tilde{\mathbf{u}} = \mathbf{u}\cdot(\Delta t/1\times10^{-3})$, in order to adequately compare pressure errors between time steps.

Results of the error in kinetic energy caused by pressure, at the first time iteration, depending on the mesh size are plotted in Fig. 5.6. The figure shows that if the mesh is refined, the error in kinetic energy caused by pressure is reduced in a second-order manner independently of the density ratio. This result matches with the theoretical approach introduced in Sec. 5.4.3, which states that the pressure error is spatially scaled as $\mathcal{O}(\Delta h^2)$. Moreover, the error difference between density ratios is explained by the proportional pressure fields obtained from the Poisson's pressure equation, Eq. 5.14; i.e., the pressure field is proportional to the density ratio, since

it is determined from the predictor mass fluxes. Consequently with this result, it is important to notice that when multiphase problems are solved using interface-capturing methods, usually the required mesh size is small enough to make the error in kinetic energy imperceptible for the physics of such type of problems.

The time integration study is plotted in Fig. 5.7. Results of the error in kinetic energy caused by pressure, at the first time iteration, indicate that time steps smaller provide proportionally smaller errors (first-order). Once again, the difference in errors between density ratios is due to the different pressure fields obtained from the Poisson's pressure equation. Moreover, the analysis of the kinetic energy conservation for the collocated mesh scheme, Sec. 5.4.3, has been developed, for simplicity, using a first-order explicit time integration method, but, as proposed by Felten and Lund [12] and studied by Fishpool and Leschziner [40], using other time integration methods may decrease the kinetic energy error. For instance, if using a second-order gear-like time integration scheme, the time step multiplying the pressure error term in Eq. 5.48 is diminished by a scaling factor of $2/3$, therefore, the pressure error term is consequently minimized.

In summary, this test verifies numerically the discrete conservation properties introduced theoretically in Sec. 5.4. On the one hand, the staggered mesh scheme discretely preserves mass, momentum and kinetic energy, if a symmetry-preserving convection scheme is used. On the other hand, the collocated mesh scheme conserves mass and momentum, however, presents an error in the kinetic energy conservation proportional to $\Delta\rho$ of the form $\mathcal{O}(\Delta t^m, \Delta h^2)$, due to the difference in pressure gradient evaluation between Eqs. 5.15 and 5.19.

5.5.2 Exact sinusoidal function

The accuracy of the two mesh schemes presented in this work is studied by means of comparing numerical results to the analytical solution of an exact sinusoidal function. In each case, a sinusoidal function is assigned to the input variables: velocities at centers of cells, $\mathbf{u}$, in the collocated case, while normal face velocities, U, in the staggered case. Then, numerical normal face velocities are obtained from Eq. 5.22 for the collocated case, dividing by face density and considering the ideal situation in which pressure terms vanish, while numerical velocities at the centers of cells are calculated from Eq. 5.27 for the staggered discretization. Finally, the root-square-mean error (rms), x_{rms}, is calculated by comparing analytical and numerical results, its definition is written as

$$x_{rms} = \sqrt{\frac{1}{n}({x_1}^2 + \cdots + {x_n}^2)}, \tag{5.52}$$

where x_i corresponds to each of the n individual errors.

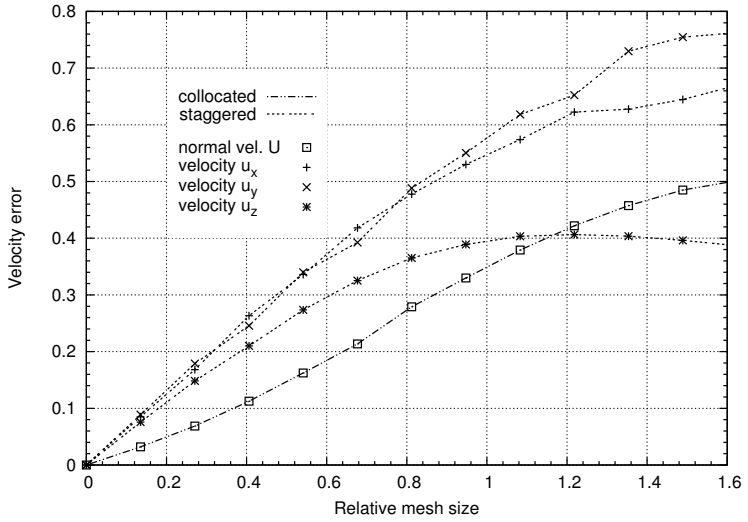


Figure 5.8: Velocity error, x_{rms}, versus relative mesh size with $\Delta\rho = 10$. Normal face velocity, U, is analyzed in the collocated scheme, while the three components of velocity (u_x, u_y and u_z) in the staggered scheme.

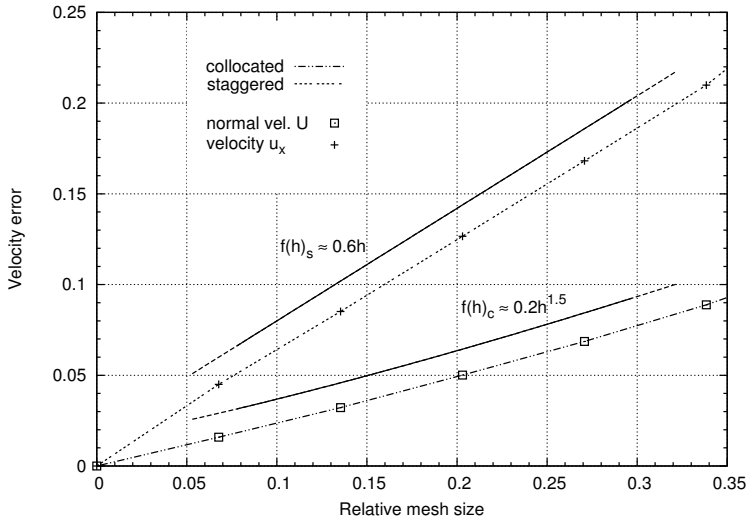


Figure 5.9: Velocity error, x_{rms}, versus relative mesh size with $\Delta\rho = 10$. Normal face velocity, U, is analyzed in the collocated scheme, while the x-component of velocity, u_x, in the staggered scheme. Approximated regression equations are calculated.

A stream function, determined by $\boldsymbol{\psi} = \frac{1}{2\pi N} sin(2\pi Nx)cos(2\pi Ny)\mathbf{k}$, is utilized in order to ensure that the resulting analytical velocity field is divergence-free. In this way, the derivation of $\boldsymbol{\psi}$, defined as $\mathbf{u} = \nabla \times \boldsymbol{\psi}$, gives the following velocity field

$$\begin{aligned} u &= -sin(2\pi Nx)sin(2\pi Ny), \\ v &= -cos(2\pi Nx)cos(2\pi Ny), \\ w &= 0, \end{aligned} \tag{5.53}$$

with a maximum velocity magnitude of one. The test is performed in a cube of side $1.0 \times 1.0 \times 1.0$ meshed by means of 9676 tetrahedral cells. Similarly to the previous test, Sec. 5.5.1, fluid with density $\rho_1 = 1\ kg/m^3$ occupies the entire cube except for a sphere of radius $R = 0.15$, which is fixed in the center of the domain and filled with a fluid that presents different densities $\rho_2 = 10, 100, 1000\ kg/m^3$.

In addition, instead of changing the mesh size, mesh refinement is performed by changing the wavelength of the input sine functions and, consequently, the radius of the centered sphere. In this way, the average mesh volume is calculated as $V_{avg} = \frac{1}{c}\sum_c V_c$, giving an average mesh spacing equal to $\Delta X_{avg} = \sqrt[3]{3V_{avg}} = 0.068$, while the effective length of the domain is defined as $L_{eff} = 1/N$, being N a variable integer value that is increased or decreased in order to enlarge or refine the effective mesh, respectively. In consequence, the relative mesh size is defined as $h = \Delta X_{avg}/L_{eff} = 0.068N$.

Velocity accuracy errors are obtained for relative mesh sizes ranging from 0 to 1.6 and plotted in Fig. 5.8 just for $\Delta\rho = 10$, since results appear to be independent to the density ratio. The figure shows that collocated normal face velocity errors are smaller than staggered cell-centered velocity ones for all relative mesh sizes, considering the ideal situation in which pressure terms in Eq. 5.22 vanish. Going further, Fig. 5.9 zooms errors U and u_x between relative mesh sizes 0 and 0.35, and shows their approximated regression equations. This figure demonstrates that collocated errors are almost second-order, $f(h)_c = 0.2h^{1.5}$, while staggered ones are just first-order (imposed by construction), $f(h)_s = 0.6h$. Consequently, although different quantities have been analyzed to study the accuracy of both schemes, due to their distinct constructions, it is reasonable to conclude that the collocated scheme presents a slightly higher order of accuracy than the staggered one.

5.5.3 Drag force on a spherical bubble in a turbulent pipe flow

The final examination of the collocated and staggered schemes, in terms of conservation and accuracy properties, is performed by calculating the drag force acting on a high-Reynolds-number clean spherical bubble fixed on the axis of a turbulent pipe flow. This test is chosen since it has been found by Merle et al. [42] that the drag force

acting on the bubble is influenced by all the length and time scales down to the Kolmogorov microscales. Hence, the use of kinetic-energy-conserving schemes should result in better calculated solutions respect to the ones obtained by non-conserving ones.

In particular, the problem under consideration falls into the classification of bubbly flow, which differs in three important aspects from bluff body flows. First, when the liquid is pure enough, it has the possibility to slip along the surface of the bubbles, in contrast to the flow over rigid bodies where the no-slip condition prevails. Second, due to the very small relative density of the bubbles compared to that of the liquid, almost all the inertia is contained in the liquid, making inertia induced hydrodynamic forces particularly important in the prediction of bubble motion. Third, the shape of the bubbles may change with the local forces, adding new degrees of freedom to an already complex problem. All these general differences together with other particular characteristics are extensively described in the work by Magnaudet et al. [43], which analyzes the motion of high-Reynolds-number bubbles in inhomogeneous flows. As a note, other studies of the flow around clean spherical bubbles, written by the same authors, may be found in the scientific literature [44–48].

The various forces acting on bubbles moving in fluids are usually named drag, history, added mass and lift. The first one, drag, refers to the slowing-down of the relative motion of a body in a fluid due to its viscosity. The history force, addresses the temporal delay in boundary layer development as the relative velocity changes with time. The added mass force, is the inertia introduced to a system because an accelerating or decelerating body moves its surrounding fluid. Finally, the lift force refers, as its name indicates, to the lift generated by the fluid circulation around an immersed body. However, in this test the added mass force is considered zero since the bubble is fixed, while the history force is neglected when compared to the drag force due to the inertia difference between the bubble and the liquid [42]. Consequently, the analysis of forces should be reduced to the study of drag and lift, but, in this work it is further reduced to just the study of drag.

Statement of the problem and computational domain

The setup of the problem consists of a spherical bubble (abbreviated as bl), with diameter d and density ρ_{bl}, placed fixed at $y = 0$ on the y-axis of a circular pipe, having diameter D and length L, that contains a fluid (abbreviated as fl) of density $\rho_{fl} = 10\rho_{bl}$. The Cartesian coordinate system attached to the bubble is (x, y, z). The physics of the problem depends on the bulk and bubble Reynolds numbers. The first one, bulk Reynolds number, is defined as $Re = \rho_{fl} u_{bk} D / \mu_{fl}$, where u_{bk} refers to the bulk velocity. The second one, bubble Reynolds number, is expressed as $Re_{bl} = \rho_{bl} u_c d / \mu_{bl}$, being u_c the time-averaged y-velocity of the flow at the centerline of the pipe. In particular, this test chooses ρ_{bl}, Re, Re_{bl} and u_{bk} as 100, 6000, 500

and 1, selected in this way so that the size of the bubble is comparable to the Taylor microscale of the flow and is about ten times the Kolmogorov microscale.

The bubble is assumed to be clean, i.e., free of any surfactant or contaminant, and the surface tension to be high enough for its shape to remain spherical. Under these assumptions, the normal velocity and tangential stress are zero at the bubble surface, written as

$$\left.\begin{array}{l} \mathbf{u}\cdot\mathbf{n}_\Gamma = 0 \\ \mathbf{n}_\Gamma \times (\boldsymbol{\tau}\cdot\mathbf{n}_\Gamma) = \mathbf{0} \end{array}\right\} \quad for \quad r = d/2, \tag{5.54}$$

where $\mathbf{n}_\Gamma$ is the unit vector normal to the surface of the bubble and $\boldsymbol{\tau} = \mu(\nabla\mathbf{u} + \nabla^T\mathbf{u})$ refers to the viscous part of the stress tensor. Accordingly, a no-slip boundary condition is imposed at the pipe wall, while a periodic condition connects the inlet and outlet of the pipe. In addition, the flow in the pipe is driven by forcing a pressure difference, ΔP, between the outlet and the inlet. In detail, the averaged momentum balance in the pipe implies that ΔP is directly related to the average shear stress at the pipe wall, $\rho_{fl}u_{\tau_0}^2$ —neglecting the average force acting on the bubble, since it turns out to be small compared to that of the average wall shear stress. In this way, ΔP may be defined as

$$\frac{\Delta P}{L} = \frac{-4\rho_{fl}u_{\tau_0}^2}{D}, \tag{5.55}$$

where u_{τ_0} is the wall shear velocity, which, in the case that $Re > 4000$, the Blasius empirical correlation, taken from the boundary-layer theory by Schlichting et al. [49], evaluates it as

$$u_{\tau_0} = u_{bk}\left(\frac{0.3164Re^{-1/4}}{8}\right)^{1/2}. \tag{5.56}$$

The numerical calculations reported in this section have been carried out with the *TermoFluids* parallel unstructured Computational Fluid Dynamics (CFD) platform [50], in which the collocated and staggered discretizations have been extended to a second-order Adams-Bashforth time integration scheme. In addition, the convection term is evaluated by first-order symmetry-preserving [11] (*sp*) and upwind [41] (*uw*) schemes. This is done because most high-order convective schemes suitable for multiphase flows are based on upwind-type schemes, as for example: QUICK [51], ENO [52] or WENO [53]. In this way, differences in the results between conserving and non-conserving discretizations may be observed.

The relation between pipe length and diameter is $L = 5D$ and the bubble's diameter is chosen as $L = 78d$. In this way, (1) the pipe is long enough to include even the largest-scale structures and (2) the velocity defect in the bubble's wake is significantly decreased before re-entering through the inlet boundary due to the periodic condition. The resulting domain is discretized by rotating 360° a 2-D grid around the y-axis, as exemplified in Fig. 5.10. In particular, the grid spacing must

satisfy requirements for the correct resolution of the boundary layers of the pipe and bubble, as well as the bubble's wake. Hence, the mesh is made up of 5.4M cells, resulting from rotating 128 times the 2-D grid discretized by means of 128 points (concentrated on the pipe wall and the bubble) in the radial direction and 330 points (accumulated at the bubble) in the axial direction. In detail, the 2-D mesh contains the first radial point near the pipe wall at $r^+ = 0.94$ — similarly to the grid spacing used for the direct numerical simulation (DNS) of the turbulent pipe flow at $Re = 5300$ by Eggels et al. [54]. Furthermore, the mesh is generated such that at least three cells lie within the bubble's boundary layer — an estimate of the thickness is $\delta/d \sim Re_{bl}^{-1/2}$ [55] —, which, according to Legendre and Magnaudet [46], is a necessary condition in order to properly solve all the scales in the vicinity of the bubble.

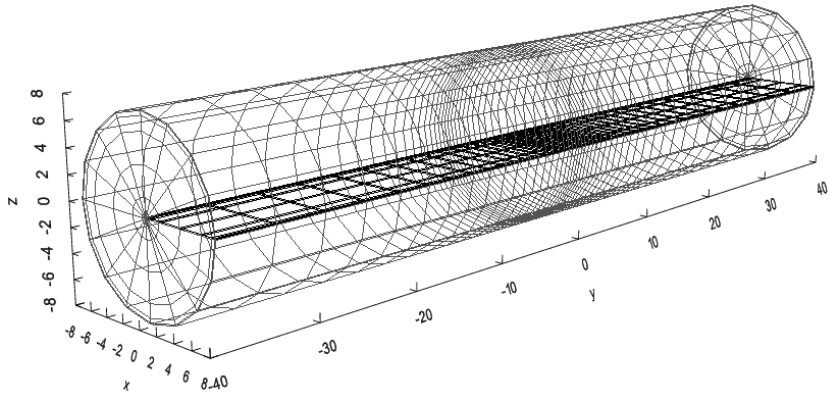


Figure 5.10: Example of a 3-D mesh generated by rotating a 2-D grid around the y-axis. This mesh is a coarse version of the one used for the calculations, however, correctly exemplifies the grid refinement near the pipe wall and bubble surface.

Turbulent pipe flow at Reynolds number 5300

Prior to simulating the flow over the spherical bubble, the solution of the turbulent pipe flow at $Re = 5300$ without the bubble is analyzed. This initial test, aside of being a method to check the numerical model without interfaces, will generate a fully developed turbulent flow useful to start the simulation with the bubble. The calculations are performed in the same domain, which in this case is discretized as in the DNS by Eggels et al. [54]: $96 \times 128 \times 256$ gridpoints equally spaced in the radial, rotational and axial directions, respectively. This mesh configuration produces a rotational coupling of the discrete Poisson's pressure equation, Eq. 5.14, resulting in

circulant submatrices that are diagonalizable in a Fourier space. This allows us to solve the Poisson's pressure equation by means of a Fast Fourier Transform (FFT) method. The algorithm used is a combination of a direct Schur complement based decomposition (DSD) and a Fourier diagonalization. The latter decomposes the original system into a set of mutually independent 2-D systems, which are solved by means of the DSD algorithm. This is detailed in the work by Borrell et al. [56].

The problem solved by means of the collocated and staggered *sp* discretizations is initiated with a random sinusoidal velocity field at dimensionless time $t^* = u_{\tau_0} t/D = 0$, reducing in this way the required time to reach the statistically steady state. In fact, at $t^* = 2.0$ the average turbulent flow can be considered steady, thus, from this point the collection of average data is initialized until $t^* = 4.0$. Differently, the resolution of the problem by means of the collocated and staggered *uw* discretizations is initiated from the respective collocated and staggered *sp* instantaneous velocity fields at $t^* = 2.0$. Once the velocity fields are initialized, the transitory state is considered to last until $t^* = 3.0$, when the collection of average data is performed during 1.0 dimensionless time unit.

The results are compared to the DNS reported by Eggels et al. [54]. In particular, the profile of the axial mean velocity, u_y, normalized by the centerline velocity, u_c, is shown in Fig. 5.11, while the root-mean-square (rms) values of the fluctuating velocities, u_{rms}, normalized by the wall shear velocity, u_{τ_0}, are shown in Fig. 5.12 using wall coordinates, ${u_{rms}}^+$. These two figures demonstrate that, although both convection schemes are first-order accurate, the results obtained by using the *sp* scheme are in good agreement with the DNS, while the ones from the *uw* scheme are really inaccurate. In fact, the use of the *uw* scheme tends to laminarize the flow as it can be observed in Fig. 5.11, where the shape of the *uw* solutions above $r/D = 0.35$ is similar to a laminar profile, and specially in Fig. 5.12, where the *uw* velocity fluctuations are completely different to the DNS results. This enormous difference between the *sp* and *uw* convection schemes is related to the conservation of kinetic energy that is shown in Fig. 5.13. In detail, the figure displays the amount of dissipation rate produced by the convection term, $-\nabla \cdot [\mathbf{u}(\frac{1}{2}\rho\mathbf{u} \cdot \mathbf{u})]$, of the kinetic energy equation, Eq. 5.43, normalized by $\rho {u_{\tau_0}}^3/D$ as function of the dimensionless time. Particularly, the figure demonstrates that, while the *sp* scheme adds kinetic energy (10^3) into the system due to the boundaries of the pipe, the *uw* scheme incorporates an artifical dissipation (-10^4) into the system that results in a laminarization of the flow. Moreover, the results obtained by the collocated and staggered *sp* discretizations show that they perform similarly. Although, Fig. 5.11 reveals that the collocated scheme is slightly more accurate between $r/D = 0.3 - 0.45$, what is justified by the better accuracy of the collocated scheme shown in Fig. 5.9. This demonstrates that with the mesh size and time step (3.7×10^{-4} *s*) used, the collocated scheme's error in kinetic energy conservation is really small and, hence, imperceptible for the physics of this problem.

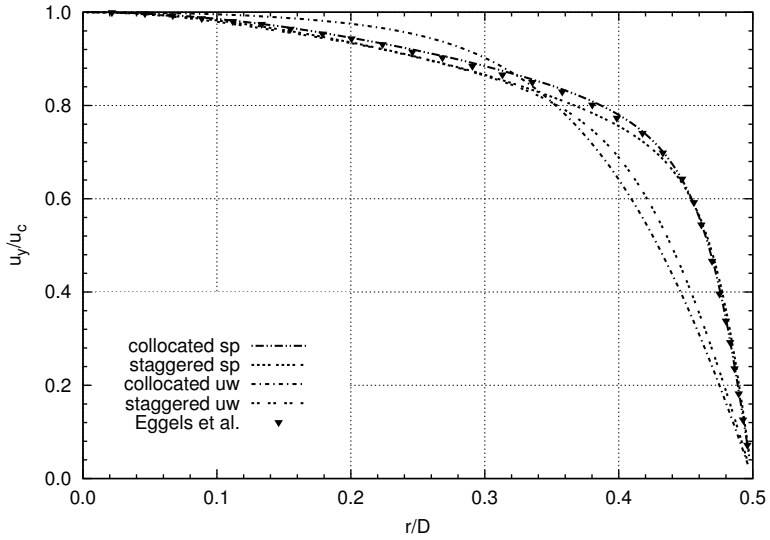


Figure 5.11: Axial mean velocity, u_y, normalized by the centerline velocity, u_c, as function of the distance from the centerline, r/D.

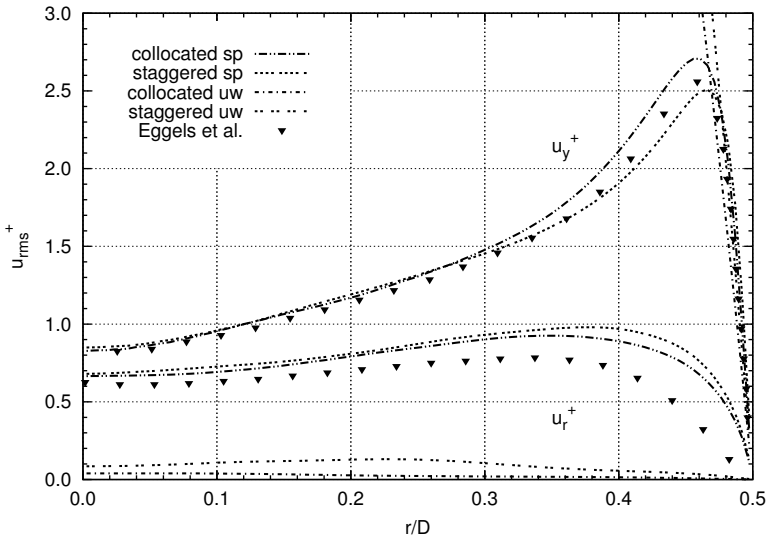


Figure 5.12: Root-mean-square velocities in wall coordinates, ${u_{rms}}^+$, as function of the distance from the centerline, r/D.

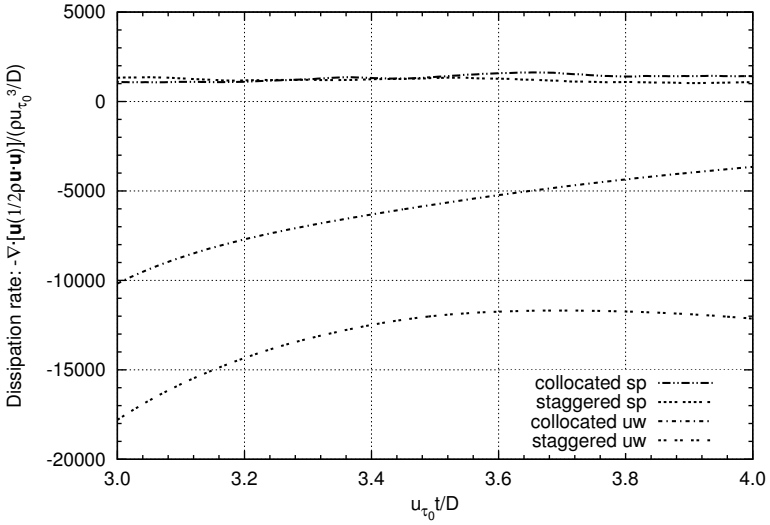


Figure 5.13: Kinetic energy convection term, $-\nabla \cdot [\mathbf{u}(\frac{1}{2}\rho \mathbf{u} \cdot \mathbf{u})]$, normalized by $\rho {u_{\tau_0}}^3 / D$ versus dimensionless time, $u_{\tau_0} t / D$.

Drag force acting on a spherical bubble

At dimensionless time $t^* = 2u_c t/d = 0$, the numerical simulations of the flow over a spherical bubble are started from the velocity fields obtained from the simulations of the turbulent pipe flow at $Re = 5300$. In particular, for each mesh scheme, both *sp* and *uw* cases are initialized from the corresponding instantaneous *sp* velocity field at $Re = 5300$. In this way, it is ensured that all cases are started from fully developed turbulent regimes. Then, independently of the spatial discretization and convection scheme, the initial velocity fields evolve during a transient period in order to reach the new Reynolds number, $Re = 6000$, while at the same time a wake behind the bubble is generated. This wake is similar to the one obtained in the case of a solid sphere. However, it differs in the fact that the fluid slips through the surface of the bubble instead of stopping. Thus, a transfer of momentum from the fluid surrounding the bubble to the fluid inside of it is produced due to viscosity. This generates 3-D oscillating vortices inside the bubble with a predominant negative velocity in the y-axis; see Fig. 5.14. Therefore, in order to arrive to statistically stationary flow conditions everywhere, simulations are advanced in time until $t^* = 20$, the instant when collection of average data is initialized.

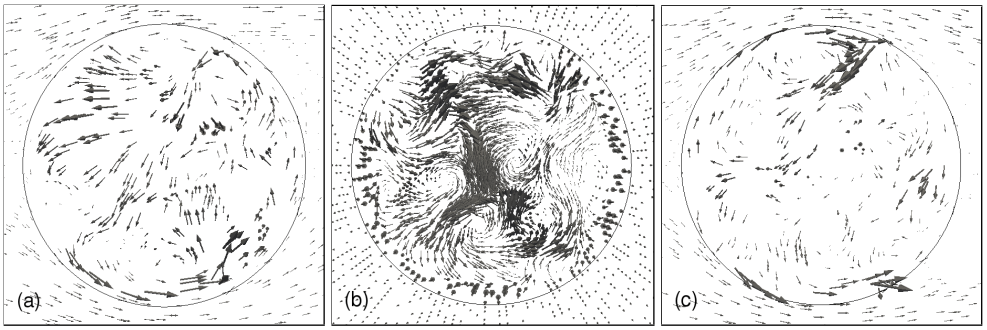


Figure 5.14: Velocity vectors inside the bubble: (a) yz-plane, (b) xz-plane and (c) xy-plane.

In this paper the analysis of forces is reduced to the study of drag, which, as demonstrated in the work by Merle et al. [42], can be fairly well predicted by Moore's expression, resulting in

$$\mathbf{F}_{Drag} = 6\pi\mu_{bl}d(1 - 2.211/Re_{bl}(t)^{1/2})\mathbf{u}_{bl}, \tag{5.57}$$

where $Re_{bl}(t) = \rho_{bl}\|\mathbf{u}_{bl}\|d/\mu_{bl}$ and $\mathbf{u}_{bl}$ are the instantaneous bubble's Reynolds number and velocity at the center of the bubble (absolute value), respectively. Furthermore, in order to analyze the transient evolution of this force, it is compared to the laminar force that would be experienced by the same bubble embedded in a laminar flow

$$\mathbf{F}_{Lam} = 6\pi\mu_{bl}d(1 - 2.211/{Re_{bl}}^{1/2})u_c\mathbf{e}_y, \tag{5.58}$$

where $\mathbf{e}_y$ corresponds to the y-axis unitary vector and time is normalized by the time scale $d/2u_c$.

The instantaneous drag forces resulting from the utilization of the different mesh and convection schemes are shown in Fig. 5.15. The outcome is that, independently of the mesh scheme, the utilization of the uw convection scheme derives in drag forces similar to the one that would be obtained in a laminar flow, since their values are linear and around $F_{Drag}/F_{Lam} = 1.0$. On the contrary, if utilizing the sp convection scheme, the drag forces for the collocated and staggered discretizations randomly oscillate around the F_{Lam} value. As seen in Tab. 5.1, this behavior agrees with the results presented in the work of Merle et al. [42], which state that the drag fluctuations originate in the viscous dissipation induced by the turbulence fluctuations. Hence, this final result demonstrates that: (1) both collocated and staggered discretizations, in the case of using the sp convection scheme, result appropriate for the numerical simulation of turbulence; (2) conservation of kinetic energy in the case of turbulent multiphase immiscible flow is important, since the contrary substantially modifies the physics of the problem.

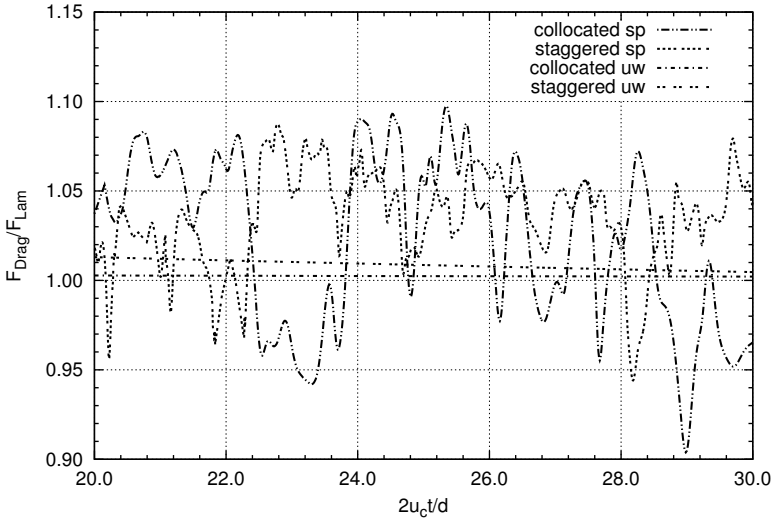


Figure 5.15: Drag force, F_{Drag}, normalized by F_{Lam} versus dimensionless time, $d/2u_c$.

	Merle et al.	collocated *sp*	staggered *sp*	collocated *uw*	staggered *uw*
F_{Drag}/F_{Lam}	∼1.005	∼1.008	∼1.006	∼1.002	∼1.004
F'_{Drag}/F_{Lam}	∼0.043	∼0.046	∼0.039	∼0.0	∼0.0

Table 5.1: Mean values and rms fluctuations of the drag force, F_{Drag}, normalized by F_{Lam} for the different mesh and convection schemes.

5.6 Conclusions

The separated multiphase flow, in which the fluids involved are immiscible, is governed by the continuity, Eq. 5.6, and Navier-Stokes, Eq. 5.7, equations in the variable-density incompressibility limit, where the physical properties are evaluated from the properties of each fluid by means of the fluid-volume fraction values, Eq. 5.8, given by the location of the interface separating them, Eq. 5.3. This model specifically conserves mass and momentum, however, the conservation of secondary derived quantities such as kinetic energy — important for the correct resolution of turbulence — cannot be directly imposed. Hence, this work proposes two unstructured finite-volume mesh discretizations, collocated and staggered, that numerically conserve mass and momentum, while at the same time minimize the errors in the conservation of kinetic energy. On the one hand, the collocated, Sec. 5.3.1, and staggered, Sec. 5.3.2, discretiza-

tions are shown to conserve mass exactly by Eqs. 5.30 and 5.31, respectively, while Eq. 5.34 states that the change in momentum of both discretizations is due to the fluxes through the boundary of the domain and the source terms. On the other hand, the discrete conservation of kinetic energy for the collocated and staggered mesh schemes is shown in Eqs. 5.47 and 5.50, respectively, stating that, if a symmetry-preserving convection scheme is used and in the absence of viscosity ($\mu = 0$) and source terms, the change in kinetic energy is due solely to the fluxes through the boundary of the domain for the staggered discretization, plus a kinetic energy error from the pressure term for the collocated one. This error in the conservation of kinetic energy — intrinsic to the collocated formulation, since it arises from the difference in pressure gradient evaluation between Eqs. 5.15 and 5.19, necessary to exactly conserve mass — is shown by Eq. 5.48 to be proportional to the density ratio and scaled by the mesh size and time step as $\mathcal{O}(\Delta t^m, \Delta h^2)$.

The theoretical conservation properties have been verified numerically by solving a three-dimensional vortex with zero mass flux at the boundaries. The test corroborates that both collocated and staggered discretizations conserve mass and momentum numerically, as theoretically expected, since there is no net flux across the domain boundaries. Moreover, the test demonstrates in Fig. 5.4 that the use of an upwind convection scheme produces an artificial kinetic energy dissipation, while Fig. 5.5 shows that using a symmetry-preserving one turns out in a zero contribution to the kinetic energy equation. Additionally, it is proved numerically that the staggered discretization preserves kinetic energy, while Figs. 5.6 and 5.7 verify that the collocated one presents a kinetic energy error proportional to the density ratio of the form $\mathcal{O}(\Delta t^m, \Delta h^2)$. In this way, it is important to notice that if multiphase problems are to be solved by means of interface-capturing methods, usually the required mesh size is small enough to make the error in kinetic energy imperceptible for the physics of such problems.

The accuracy of the collocated and staggered discretizations has been analyzed by means of comparing their numerical results with the analytical solution of an exact sinusoidal function. The results show that collocated errors are smaller than staggered ones for all relative mesh sizes, considering the ideal situation in which pressure terms in Eq. 5.22 vanish. In particular, Fig. 5.9 demonstrates that collocated errors are almost second-order, $f(h)_c = 0.2h^{1.5}$, while, as imposed by construction, staggered ones are just first-order, $f(h)_s = 0.6h$. Consequently, the test concludes that the collocated scheme presents a slightly higher order of accuracy than the staggered one.

The drag force acting on a high-Reynolds-number clean spherical bubble fixed on the axis of a turbulent pipe flow has been calculated by means of the collocated and staggered discretizations presented, using both the symmetry-preserving and upwind convection schemes, in order to analyze their properties on turbulent cases.

First, the problem has been solved without the bubble and the results have been compared to DNS data. The outcome is that both mesh discretizations, in the case of utilizing the symmetry-preserving convection scheme, are able to properly resolve the turbulent pipe flow, while the use of the upwind scheme produces laminar solutions; see Figs. 5.11 and 5.12. This result can be extrapolated to most high-order convection schemes, e.g., QUICK, ENO or WENO, since they are all based on upwind approximations and, hence, disregard symmetry properties. Moreover, this inital test also demonstrates that, if fine enough meshes and small time steps are used, the collocated kinetic energy error is certainly minimized, consequently, as seen in Fig. 5.11, the collocated scheme is slightly more accurate than the staggered one. Second, the bubble has been introduced in the pipe, the numerical solutions have been obtained and, for each mesh and convection scheme, the drag force has been calculated and plotted in Fig. 5.15. Similar to the case without the bubble, the utilization of the upwind convection scheme, with independence of the mesh discretization, has resulted in drag forces presenting laminar flow behaviors. Contrary, in the case of utilizing the symmetry-preserving convection scheme, both the collocated and staggered discretizations produce oscillating drag forces, agreeing in this way with the benchmark results.

In summary, this work demonstrates that, in the case of multiphase immiscible flow, the use of discretizations that properly conserve mass, momentum and kinetic energy —instead of the conventional idea of prioritizing stability, robustness and accuracy —, turns out in better numerical solutions, especially if turbulence dominates the physics of the problems under consideration. On this regard, this paper proposes two mesh schemes, collocated and staggered, that contemplate these restrictions, although with different properties for the solutions. In particular, the collocated scheme is more accurate and presents no geometric difficulties (no circumcenters are needed), while the staggered scheme numerically preserves kinetic energy and is more stable (do not display spurious pressure modes). Going further, the development of discretizations presenting higher stability and/or accuracy is possible, but always under the constraint of respecting the properties of the continuous equations.

Appendix

The detailed analysis of the discrete kinetic energy equations is simplified if two important identities involving combinations of interpolation and differentiation operators are introduced. The two-variable identity was first presented by Morinishi et al. [9] and restated in finite-volume form by Felten and Lund [12], while the three-variable one is developed in this work. Prior to presenting the identities, some definitions are needed. In particular, φ, ψ and χ represent three general variables and Q_f is a general quantity known on cell faces, i.e., no interpolation is needed. Additionally,

two special interpolator operators for products are defined as

$$\widehat{\varphi\psi} = \frac{1}{2}(\varphi_c\psi_{nb} + \varphi_{nb}\psi_c), \tag{5.59}$$

$$\begin{aligned}\widehat{\varphi\psi\chi} = \frac{1}{4}(&2\varphi_c\psi_c\chi_{nb} + 2\varphi_c\psi_{nb}\chi_c + 2\varphi_{nb}\psi_c\chi_c \\ &+ \varphi_c\psi_{nb}\chi_{nb} + \varphi_{nb}\psi_c\chi_{nb} + \varphi_{nb}\psi_{nb}\chi_c).\end{aligned} \tag{5.60}$$

where subscripts c and nb correspond to any pair of cells sharing a face.

In this way, the two-variable relation, for a given cell c, arises from the combination of the trivial identities

$$\begin{aligned}\varphi_c \sum_{f\in F(c)} \bar{\psi}_f Q_f - \sum_{f\in F(c)} \varphi_c \frac{1}{2}\left(\psi_c + \psi_{nb}\right) Q_f = 0, \\ \psi_c \sum_{f\in F(c)} \bar{\varphi}_f Q_f - \sum_{f\in F(c)} \psi_c \frac{1}{2}\left(\varphi_c + \varphi_{nb}\right) Q_f = 0,\end{aligned} \tag{5.61}$$

where the overbars refer to semi-summed interpolations. Then, if Eq. 5.59 is used to simplify the result, the final relation is written as

$$\varphi_c \sum_{f\in F(c)} \bar{\psi}_f Q_f + \psi_c \sum_{f\in F(c)} \bar{\varphi}_f Q_f = \sum_{f\in F(c)} \widehat{\varphi\psi} Q_f + (\varphi_c\psi_c) \sum_{f\in F(c)} Q_f. \tag{5.62}$$

Analogously, the three-variable identity is obtained from the combination of the following relations

$$\begin{aligned}\varphi_c \sum_{f\in F(c)} \bar{\psi}_f\bar{\chi}_f Q_f - \sum_{f\in F(c)} \varphi_c \frac{1}{2}\left(\psi_c + \psi_{nb}\right)\frac{1}{2}\left(\chi_c + \chi_{nb}\right) Q_f = 0, \\ \psi_c \sum_{f\in F(c)} \bar{\varphi}_f\bar{\chi}_f Q_f - \sum_{f\in F(c)} \psi_c \frac{1}{2}\left(\varphi_c + \varphi_{nb}\right)\frac{1}{2}\left(\chi_c + \chi_{nb}\right) Q_f = 0, \\ \chi_c \sum_{f\in F(c)} \bar{\varphi}_f\bar{\psi}_f Q_f - \sum_{f\in F(c)} \chi_c \frac{1}{2}\left(\varphi_c + \varphi_{nb}\right)\frac{1}{2}\left(\psi_c + \psi_{nb}\right) Q_f = 0,\end{aligned} \tag{5.63}$$

into one equation. Finally, making use of Eq. 5.61, the result is simplified to

$$\begin{aligned}\varphi_c \sum_{f\in F(c)} \bar{\psi}_f\bar{\chi}_f Q_f + \psi_c \sum_{f\in F(c)} \bar{\varphi}_f\bar{\chi}_f Q_f + \chi_c \sum_{f\in F(c)} \bar{\varphi}_f\bar{\psi}_f Q_f \\ = \sum_{f\in F(c)} \widehat{\varphi\psi\chi} Q_f + \frac{3}{4}(\varphi_c\psi_c\chi_c) \sum_{f\in F(c)} Q_f.\end{aligned} \tag{5.64}$$

Acknowledgements

This work has been financially supported by the *Ministerio de Economía y Competitividad, Secretaría de Estado de Investigación, Desarrollo e Innovación*, Spain (ENE-2010-17801), a FPU Grant by the *Ministerio de Educación, Cultura y Deporte*, Spain (AP-2008-03843) and by *Termo Fluids S.L.*

References

[1] G. Tryggvason, S. Dabiri, B. Aboulhasanzadeh, and J. Lu. Multiscale Considerations in Direct Numerical Simulation of Multiphase Flows. *Physics of Fluids*, 25:031302, 2013.

[2] J. M. Tang and C. Vuick. On Deflation and Singular Symmetric Positive Semi-Definite Matrices. *Journal of Computational and Applied Mathematics*, 206:603–614, 2007.

[3] S. P. MacLachlan, J. M. Tang, and C. Vuik. Fast and Robust Solvers for Pressure-Correction in Bubbly Flow Problems. *Journal of Computational Physics*, 227:9742–9761, 2008.

[4] A. N. Marques, J. C. Nave, and R. R. Rosales. A Correction Function Method for Poisson Problems with Interface Jump Conditions. *Journal of Computational Physics*, 230:7567–7597, 2011.

[5] G. H. Miller and E. G. Puckett. A Neumann-Neumann Preconditioned Iterative Substructuring Approach for Computing Solutions to Poisson's Equation with Prescribed Jumps on an Embedded Boundary. *Journal of Computational Physics*, 235:683–700, 2013.

[6] M. Rudman. A Volume-Tracking Method for Incompressible Multifluid Flows with Large Density Variations. *International Journal for Numerical Methods in Fluids*, 28:357–378, 1998.

[7] O. Desjardins and V. Moureau. Methods for Multiphase Flows with High Density Ratio. In *Proceedings of the Summer Program 2010*, pages 313–322, 2010.

[8] M. Raessi and H. Pitsch. Consistent Mass and Momentum Transport for Simulating Incompressible Interfacial Flows with Large Density Ratios Using the Level Set Method. *Computers & Fluids*, 63:70–81, 2012.

[9] Y. Morinishi, T. S. Lund, O. V. Vasilyev, and P. Moin. Fully Conservative Higher Order Finite Difference Schemes for Incompressible Flow. *Journal of Computational Physics*, 143:90–124, 1998.

[10] B. Perot. Conservation Properties of Unstructured Staggered Mesh Schemes. *Journal of Computational Physics*, 159:58–89, 2000.

[11] R. W. C. P. Verstappen and A. E. P. Veldman. Symmetry-Preserving Discretization of Turbulent Flow. *Journal of Computational Physics*, 187:343–368, 2003.

[12] F. N. Felten and T. S. Lund. Kinetic Energy Conservation Issues Associated with the Collocated Mesh Scheme for Incompressible Flow. *Journal of Computational Physics*, 215:465–484, 2006.

[13] L. Jofre, O. Lehmkuhl, J. Ventosa, F. X. Trias, and A. Oliva. Conservation Properties of Unstructured Finite-Volume Mesh Schemes for the Navier-Stokes Equations. *Numerical Heat Transfer, Part B: Fundamentals*, 65:53–79, 2014.

[14] F. X. Trias, A. Gorobets, M. Soria, and A. Oliva. Direct Numerical Simulation of a Differentially Heated Cavity of Aspect Ratio 4 with Rayleigh Numbers up to 10^{11} - Part II: Heat Transfer and Flow Dynamics. *International Journal of Heat and Mass Transfer*, 53:674–683, 2010.

[15] I. Rodríguez, R. Borrell, O. Lehmkuhl, C. D. Pérez-Segarra, and A. Oliva. Direct Numerical Simulation of the Flow over a Sphere at $Re = 3700$. *Journal of Fluid Mechanics*, 679:263–287, 2011.

[16] I. Rodríguez, O. Lehmkuhl, R. Borrell, and A. Oliva. Direct Numerical Simulation of a NACA 0012 in Full Stall. *International Journal of Heat and Fluid Flow*, 43:194–203, 2013.

[17] O. Lehmkuhl, I. Rodríguez, R. Borrell, and A. Oliva. Low-Frequency Unsteadiness in the Vortex Formation Region of a Circular Cylinder. *Physics of Fluids*, 25:085109, 2013.

[18] F. Nicoud. Conservative High-Order Finite-Difference Schemes for Low-Mach Number Flows. *Journal of Computational Physics*, 158:71–97, 2000.

[19] O. Desjardins, G. Blanquart, G. Balarac, and H. Pitsch. High Order Conservative Finite Difference Scheme for Variable Density Low Mach Number Turbulent Flows. *Journal of Computational Physics*, 227:7125–7159, 2008.

[20] Y. Morinishi. Skew-Symmetric form of Convective Terms and Fully Conservative Finite Difference Schemes for Variable Density Low-Mach Number Flows. *Journal of Computational Physics*, 229:276–300, 2010.

[21] D. Fuster. An Energy Preserving Formulation for the Simulation of Multiphase Turbulent Flows. *Journal of Computational Physics*, 235:114–128, 2013.

[22] F. Denner and B. G. M. van Wachem. On the Convolution of Fluid Properties and Surface Force for Interface Capturing Methods. *International Journal of Multiphase Flow*, 54:61–64, 2013.

[23] C. W. Hirt, J. L. Cook, and T. D. Butler. A Lagrangian Method for Calculating the Dynamics of an Incompressible Fluid with Free Surface. *Journal of Computational Physics*, 5:103–124, 1970.

[24] C. W. Hirt, A. A. Amsden, and J. L. Cook. An Arbitrary Lagrangian-Eulerian Computing Method for All Flow Speeds. *Journal of Computational Physics*, 135:203–216, 1997.

[25] H. H. Hu, N. A. Patankar, and M. Y. Zhu. Direct Numerical Simulations of Fluid-Solid Systems Using the Arbitrary Lagrangian-Eulerian Technique. *Journal of Computational Physics*, 169:427–462, 2001.

[26] F. H. Harlow and J. E. Welch. Numerical Calculation of Time-Dependent Viscous Incompressible Flow of Fluid with Free Surface. *Physics of Fluids*, 8:2182–2189, 1965.

[27] C. S. Peskin. Numerical Analysis of Blood Flow in the Heart. *Journal of Computational Physics*, 25:220–252, 1977.

[28] G. Tryggvason, B. Bunner, A. Esmaeeli, D. Juric, N. Al-Rawashi, W. Tauber, J. Han, S. Nas, and Y. J. Jan. A Front-Tracking Method for the Computations of Multiphase Flow. *Journal of Computational Physics*, 169:708–759, 2001.

[29] C. W. Hirt and B. D. Nichols. Volume of fluid (VOF) Method for the Dynamics of Free Boundaries. *Journal of Computational Physics*, 39:201–225, 1981.

[30] W. J. Rider and D. B. Kothe. Reconstructing Volume Tracking. *Journal of Computational Physics*, 141:112–152, 1998.

[31] 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*, 35:1011–1032, 2006.

[32] S. Osher and J. Sethian. Fronts Propagating with Curvature Dependent Speed: Algorithms Based on Hamilton-Jacobi Formulations. *Journal of Computational Physics*, 79:12–49, 1988.

[33] M. Sussman, P. Smereka, and S. Osher. A Level Set Approach for Computing Solutions to Incompressible Two-Phase Flow. *Journal of Computational Physics*, 114:146–159, 1994.

[34] D. Enright, R. Fedkiw, J. Ferziger, and I. Mitchell. A Hybrid Particle Level Set Method for Improved Interface Capturing. *Journal of Computational Physics*, 183:83–116, 2002.

[35] E. Olsson and G. Kreiss. A Conservative Level Set Method for Two Phase Flow. *Journal of Computational Physics*, 210:225–246, 2005.

[36] O. Desjardins, V. Moureau, and H. Pitsch. An Accurate Conservative Level Set/Ghost Fluid Method for Simulating Turbulent Atomization. *Journal of Computational Physics*, 227:8395–8416, 2008.

[37] N. Balcázar, L. Jofre, O. Lehmkuhl, J. Castro, and J. Rigola. A Finite-Volume/Level-Set Method for Simulating Two-Phase Flows on Unstructured Grids. *International Journal of Multiphase Flow*, 64:55–72, 2014.

[38] 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.

[39] J. B. Perot. Discrete Conservation Properties of Unstructured Mesh Schemes. *Annual Review of Fluid Mechanics*, 43:299–318, 2011.

[40] G. M. Fishpool and M. A. Leschziner. Stability Bounds for Fractional-Step Schemes for the Navier-Stokes Equations at High Reynolds Number. *Computers & Fluids*, 38:1289–1298, 2009.

[41] A. E. P. Veldman and K. Lam. Symmetry-Preserving Upwind Discretization of Convection on Non-Uniform Grids. *Applied Numerical Mathematics*, 58:1881–1891, 2008.

[42] A. Merle, D. Legendre, and J. Magnaudet. Forces on a High-Reynolds-Number Spherical Bubble in a Turbulent Flow. *Journal of Fluid Mechanics*, 532:53–62, 2005.

[43] J. Magnaudet and I. Eames. The Motion of High-Reynolds-Number Bubbles in Inhomogeneous Flows. *Annual Review of Fluid Mechanics*, 32:659–708, 2000.

[44] J. Magnaudet, M. Rivero, and J. Fabre. Accelerated Flows Past a Rigid Sphere or a Spherical Bubble. Part 1. Steady Straining Flow. *Journal of Fluid Mechanics*, 284:97–135, 1995.

[45] A. Blanco and J. Magnaudet. The Structure of the Axisymmetric High-Reynolds Number Flow Around an Ellipsoidal Bubble of Fixed Shape. *Physics of Fluids*, 7:1265–1274, 1995.

[46] D. Legendre and J. Magnaudet. The Lift Force on a Spherical Bubble in a Viscous Linear Shear Flow. *Journal of Fluid Mechanics*, 368:81–126, 1998.

[47] J. Magnaudet and D. Legendre. The Viscous Drag Force on a Spherical Bubble with a Time-Dependent Radius. *Physics of Fluids*, 10:550–555, 1998.

[48] J. Magnaudet and D. Legendre. Some Aspects of the Lift Force on a Spherical Bubble. *Applied Scientific Research*, 58:41–61, 1998.

[49] H. Schlichting, K. Gersten, E. Krause, H. J. Oertel, and C. Mayes. *Boundary-Layer Theory*. Springer, 2004.

[50] O. Lehmkuhl, C. D. Pérez-Segarra, R. Borrell, M. Soria, and A. Oliva. TERMOFLUIDS: A New Parallel Unstructured CFD Code for the Simulation of Turbulent Industrial Problems on Low Cost PC Cluster. In *Proceedings of the Parallel CFD Conference*, pages 1–8, 2007.

[51] B. P. Leonard. A Stable and Accurate Convective Modelling Procedure Based on Quadratic Upstream Interpolation. *Computer Methods in Applied Mechanics and Engineering*, 19:59–98, 1979.

[52] A. Harten, B. Engquist, S. Osher, and S. R. Chakravarthy. Uniformly High Order Accurate Essentially Non-Oscillatory Schemes, III. *Journal of Computational Physics*, 71:231–303, 1987.

[53] X. D. Liu, S. Osher, and T. Chan. Weighted Essentially Non-Oscillatory Schemes. *Journal of Computational Physics*, 115:200–212, 1994.

[54] J. G. M. Eggels, F. Unger, M. H. Weiss, J. Westerweel, R. J. Adrian, R. Friedrich, and F. T. M. Nieuwstadt. Fully Developed Turbulent Pipe Flow: a Comparison between Direct Numerical Simulation and Experiment. *Journal of Fluid Mechanics*, 268:175–209, 1994.

[55] D. W. Moore. The Boundary Layer on a Spherical Gas Bubble. *Journal of Fluid Mechanics*, 16:161–176, 1963.

[56] R. Borrell, O. Lehmkuhl, F. X. Trias, and A. Oliva. Parallel Direct Poisson Solver for Discretizations with one Fourier Diagonalizable Direction. *Journal of Computational Physics*, 230:4723–4741, 2011.

6

Numerical simulation of the Richtmyer-Meshkov instability

Main contents of this chapter have been published in:

L. Jofre, N. Balcázar, O. Lehmkuhl, J. Castro, and A. Oliva. Numerical Study of the Incompressible Richtmyer-Meshkov Instability. Interface-Capturing Methods on General Meshes. In Proceedings of the *15th International Conference on Fluid Flow Technologies*, Budapest (Hungary), September 2012.

Abstract. The Richtmyer-Meshkov instability occurs at a nearly planar interface separating two fluids that are impulsively accelerated in the direction normal to the interface. This impulsive acceleration can be the result of an impulsive body force or a passing shock wave. The initial development of the instability creates small amplitude perturbations which initially grow linearly with time. This is followed by a nonlinear regime with bubbles appearing in the case of a light fluid penetrating a heavy fluid, and with spikes appearing in the case of a heavy fluid penetrating a light fluid. This instability is important in astrophysical phenomena and technological applications, such as: inertial confinement fusion and processes involving explosions. In this work, the incompressible Richtmyer-Meshkov instability is numerically simulated by means of a Volume-of-Fluid method. In addition, the numerical outcome is compared to experimental data.

6.1 Introduction

6.1.1 Richtmyer-Meshkov instability

The Richtmyer-Meshkov (RM) instability —named after the pioneering works of Richtmyer [1] and Meshkov [2]— occurs at a nearly planar interface separating two fluids that are impulsively accelerated in the direction normal to the interface, as a result of a impulsive body force or a passing shock wave. The initial development of the instability creates small amplitude perturbations which initially grow linearly with time. This initial evolution is followed by a nonlinear regime with bubbles appearing in the case of a light fluid penetrating a heavy fluid, and with spikes appearing in the case of a heavy fluid penetrating a light fluid.

Recent experiments of the RM instability initiated with two- (2-D) and three-dimensional (3-D) single-mode perturbations [3, 4] have verified the early time linear growth predicted by Richtmyer. However, no nonlinear solution capable of predicting the behavior from the early linear stages into the far nonlinear regime is available at the moment. Many researchers have developed nonlinear analyses [5], heuristic models [6] and analytical approaches [7], which capture some of the physics of the late-time asymptotic flow, but they all necessarily must incorporate empirical constants that limit their generalization.

Therefore, this work aims at numerically simulating the RM instability in order to demonstrate the capacity of the computational techniques to study physical phenomena. In particular, this paper is focused on the case of the RM instability comprised of two incompressible immiscible liquids with two- (2-D) and three-dimensional (3-D) single-mode initial perturbations. The interface of the instability is captured by a Volume-of-Fluid method and the momentum equations are discretized by means of a staggered mesh scheme. The numerical results of amplitude, velocity and vorticity of the instability are analyzed and compared to the experimental data provided by Niederhaus, Chapman and Jacobs [3, 4].

6.1.2 Method of interface-capturing

The contact of different fluids or phases in motion produces a thin region, named interface, that separates them. This kind of flows are usually classified as interfacial flows and are found in multiple fields, such as: engineering, fundamental physics and geophysics. Typical examples of this phenomena are bubbles, drops, sprays, jets, waves, clouds and, in particular, the RM instability.

There are many different methods to follow the motion of the interface between fluids —a general list of them is found in the work by Scardovelli and Zaleski [8]—, but in general these may be classified in two large groups: interface-tracking and interface-capturing. On the one hand, the interface-tracking approaches chase the

interface as it moves by defining it as a boundary between two subdomains of a moving grid, or by following the Lagrangian trajectories of massless particles. On the other hand, the interface-capturing approaches describe the motion of the interface by embedding the different fluids into a static grid with the help of scalar values. In particular, this work chooses the interface-capturing Volume-of-Fluid (VOF) method, since its formulation preserves volume, large changes in interface's topology are properly handled and interfaces between fluids are maintained in a sharp manner. In detail, the first VOF implementations were presented in the 1970s for 2-D Cartesian meshes, being the method proposed by Hirt and Nichols [9] the reference one. In recent years, the method has been improved and adapted for 3-D meshes in a Cartesian approach by Liovic et al. [10] and, more generally, on 3-D Cartesian and unstructured meshes by Jofre et al. [11].

6.1.3 Discretization of the Navier-Stokes equations

One of the decisions to make regarding the discretization of the Navier-Stokes equations is the placement of the velocity and pressure nodes on the grid, since an inappropiate selection may result in a checkerboard solution caused by the decoupling of velocity and pressure. This issue is more critical when sharp discontinuities are present in the domain, as in the case of multiphase flow. In order to solve this problem, there are two main mesh arrangements for the calculation of the Navier-Stokes equations: the collocated and staggered schemes.

One of the first collocated schemes was presented by Rhie and Chow [12] for body-fitted meshes in the 1980s. In recent years, the scheme has been extended to unstructured meshes and improved to diminish the kinetic energy conservation error by means of: (1) using a least-squares procedure to calculate the pressure-gradient term [13]; (2) utilizing vectors that span the null space of the discrete pressure Laplacian to obtain a smooth pressure field [14]; or (3) proposing a special definition for the face mass fluxes that exactly conserves mass [15,16]. The main characteristic of this scheme is that the velocity and pressure nodes are located at the same grid points, what may result in a checkerboard pressure problem when solving discontinuous flows like the RM instability.

In order to avoid this problem, a staggered mesh arrangement is used in this work. This type of scheme is a numerical strategy where variables are located at different points within the mesh. Many different staggering schemes are possible. However, in this work we are interested in the scheme presented by Perot [17], since it is a generalization to unstructured meshes of the one originally presented by Harlow and Welch [18]. This scheme locates pressure at cell centers and normal velocities at cell faces. The main variable is the face mass flux, from which velocity vectors at cell centers are interpolated in such a way that momentum and kinetic energy are conserved.

6.2 Governing equations

A single set of mass and momentum equations may be utilized to describe the flow in a domain composed of different fluids. In order to do so, the Navier-Stokes equations in the variable-density incompressibility limit, together with the advection equation that captures the motion of the interface between fluids, need to be considered.

In detail, the fluid k volume fraction color function, $C_k(\mathbf{x}, t)$, used to capture the motion of the interface, is defined by an identity function as

$$C_k(\mathbf{x}, t) = \begin{cases} 1 & \text{if there is fluid } k \\ 0 & \text{otherwise,} \end{cases} \tag{6.1}$$

where $\mathbf{x}$ is a position in space and t refers to time instant. Therefore, for each cell c, with volume V_c, its k'th fluid volume fraction at time t is evaluated as

$$C_k[c, t] = \frac{\int C_k(\mathbf{x}, t) dV_c}{V_c}. \tag{6.2}$$

Particularly, in the absence of phase change the volume fraction advection equation results in

$$\frac{\partial C_k}{\partial t} + \nabla \cdot (C_k \mathbf{u}) = 0. \tag{6.3}$$

Moreover, under the hypothesis of incompressible flow and negligible surface tension, the conservation equations of mass and momentum are defined as

$$\nabla \cdot \mathbf{u} = 0, \tag{6.4}$$

$$\frac{\partial (\rho \mathbf{u})}{\partial t} + \nabla \cdot (\rho \mathbf{u}\mathbf{u}) = -\nabla p + \nabla \cdot (\mu [\nabla \mathbf{u} + \nabla^T \mathbf{u}]) + \rho \mathbf{g}, \tag{6.5}$$

where $\mathbf{u}$ and p represent velocity and pressure, respectively, and $\mathbf{g}$ is the gravitational acceleration. Additionally, density, ρ, and dynamic viscosity, μ, are interpolated from the properties of each fluid k by means of the fluid-volume fraction values, written as

$$\rho = \sum_k C_k \rho_k \quad \text{and} \quad \mu = \sum_k C_k \mu_k. \tag{6.6}$$

Therefore, the solution of the momentum equation, Eq. 6.5, provides the velocity field used in the volume fraction advection equation, Eq. 6.3, to calculate the new volume fraction scalar field.

6.3 Numerical model

6.3.1 Volume-of-Fluid method

The VOF method discretizes, applying the divergence theorem and using a first-order explicit time scheme, the volume fraction advection equation, Eq. 6.3, for each cell c as

$$C_k^{n+1} - C_k^n + \frac{1}{V_c} \sum_{f \in F(c)} V_{k,f}^n = 0, \tag{6.7}$$

where the superscript n refers to the discrete time level and $V_{k,f}$ is the volumetric flow of fluid k across face f, calculated geometrically from the total volume flux given by

$$V_f = |\mathbf{u}_f \cdot \mathbf{n}_f| A_f \Delta t = \sum_k V_{k,f}, \tag{6.8}$$

where Δt is the time step and $\mathbf{u}_f$, $\mathbf{n}_f$ and A_f correspond, respectively, to the velocity, the unit-outward normal and the area of face f.

In order to calculate $V_{k,f}$, two consecutive steps are required: interface reconstruction and advection. First, the interface is reconstructed by approximating its form to a geometric surface. In particular, this work reconstructs interfaces by planes using a Least Square Gradient (LSG) approach of the Youngs method [19]. Second, once the interface has been reconstructed, the advection step geometrically constructs volumetric flows (polyhedrons) at mesh cell faces and, later, cuts them by the reconstructed interface in order to compute the amount of fluid k across the faces, $V_{k,f}$. These two steps are fully explained in the work by Jofre et al. [11].

6.3.2 Unstructured staggered mesh scheme

The multiphase flow of immiscible fluids presents sharp discontinuities in the domain due to the difference in physical properties between fluids. Therefore, in order to avoid possible spurious pressure modes, the Navier-Stokes equations, Eq. 6.5, are discretized by means of the unstructured staggered mesh scheme presented by Perot [17]. This mesh scheme, instead of evaluating velocities at cell centers, evolves face mass fluxes, $M_f = \rho_f \mathbf{u}_f \cdot \mathbf{n}_f A_f$, in time. Hence, some preliminary remarks are needed. First, face-centered control volumes are defined for each face f as $V_f = (W_f^a + W_f^b) A_f$, where W_f is the distance between the circumcenter of face f and each of the circumcenters of the neighbor cells that contain this face, while A_f is the surface of face f; see Fig. 6.1. Second, convective, diffusive and source terms are calculated at the centers of cells as non-volumetric quantities, later, they are interpolated to faces using W_f. In detail, the velocity-pressure coupling is solved by

means of a fractional step procedure [20], written in staggered discrete form as

$$M_f^{n+1} = M_f^p - \Delta t(p_b^{n+1} - p_a^{n+1})\frac{A_f}{(W_f^a + W_f^b)}, \tag{6.9}$$

$$M_f^p = M_f^n - \Delta t\left[W_f^a(\mathbf{c}_a^n - \mathbf{d}_a^n - \mathbf{s}_a^{n+1}) + W_f^b(\mathbf{c}_b^n - \mathbf{d}_b^n - \mathbf{s}_b^{n+1})\right]\cdot\mathbf{n}_f\frac{A_f}{(W_f^a + W_f^b)}, \tag{6.10}$$

where subscripts a and b refer to the two cells adjacent to face f and $\mathbf{c}$, $\mathbf{d}$ and $\mathbf{s}$ are the non-volumetric cell-centered discretizations of the convective, diffusive and source terms, evaluated for each cell c as

$$\mathbf{c}_c^n = \frac{1}{V_c}\sum_{f\in F(c)}\boldsymbol{\phi}_f^n\hat{M}_f^n, \quad \mathbf{s}_c^{n+1} = \rho_c^{n+1}\mathbf{g}, \tag{6.11}$$

$$\mathbf{d}_c^n = \frac{1}{V_c}\sum_{f\in F(c)}\mu_f^n\left[(\mathbf{u}_{nb}^n - \mathbf{u}_c^n)\frac{A_f}{\delta d_f} + \nabla^T\mathbf{u}_f^n\cdot\hat{\mathbf{n}}_f A_f\right],$$

where $\boldsymbol{\phi}_f$ is the convected velocity at face f and the length δd_f is the normal-projected distance between the centroids of cells a and b.

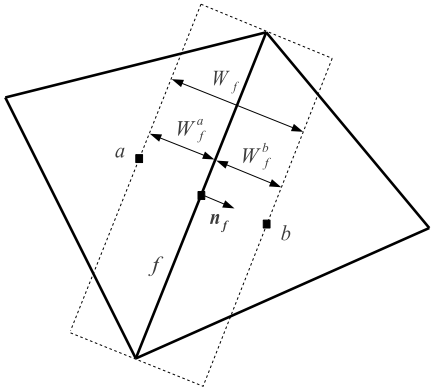


Figure 6.1: Notation for the staggered mesh scheme on a 2-D unstructured mesh.

Next, dividing Eq. 6.9 by face density, $\rho_f^{n+1} = \frac{1}{2}(\rho_c^{n+1} + \rho_{nb}^{n+1})$, summing over the bordering faces of cell c and making use of the incompressibility constraint, results in the discrete Poisson's pressure equation

$$\sum_{f\in F(c)}\frac{\hat{M}_f^p}{\rho_f^{n+1}} = \Delta t\sum_{f\in F(c)}\frac{1}{\rho_f^{n+1}}(p_{nb}^{n+1} - p_c^{n+1})\frac{A_f}{\delta d_f}, \tag{6.12}$$

which, by means of a preconditioned conjugate gradient solver [21], solves the pressure field at time instant $n+1$. Following the obtention of this p^{n+1} field, M_f^{n+1} is calculated from Eq. 6.9.

Finally, the staggered mesh scheme discretizes mass fluxes in time, thus, velocities at centers of cells need to be interpolated from face normal values. Hence, if a first-order approximation of the momentum field (constant $\rho\mathbf{u}$) is assumed, the cell-centered velocities are interpolated from the face mass fluxes as

$$\mathbf{u}_c = \frac{1}{\rho_c V_c} \sum_{f \in F(c)} \mathbf{r}_f^c \hat{M}_f, \tag{6.13}$$

where $\mathbf{r}_f^c = \mathbf{x}_f^{CG} - \mathbf{x}_c^{CC}$ is the vector from the circumcenter of cell c, $\mathbf{x}_c^{CC}$, to the centroid of face f, $\mathbf{x}_f^{CG}$.

6.4 Numerical results

6.4.1 Statement of the problem

The numerical simulations of the RM instability are based on the 2-D and 3-D experiments of Niederhaus and Jacobs [3] and Chapman and Jacobs [4], respectively. In detail, the 2-D tank is 119.9 mm in width and 254.4 mm in height, while the dimensions for the 3-D case are 72.6 mm in width and depth and 250 mm in height. The lighter upper fluid, ρ_1, and the heavier bottom fluid, ρ_2, result in an Atwood number equal to $A = (\rho_2 - \rho_1)/(\rho_2 + \rho_1) = 0.1587$. Similarly to the experiments, the initial shape of the interface between fluids is set equal to a small periodical disturbance in order to make the system unstable. In the 2-D case the amplitude is $a_0 = 0.23/k$, where $k = 2\pi/\lambda$, and the wavelength is $\lambda = 82.6$, while in the 3-D case the amplitude is $a_0 = 0.38/k$ and the wavelength is $\lambda = 48.4$. In this way, the initial disturbances are approximated as $\eta = a_0 \cdot \sin(kx)$ for the 2-D case, and $\eta = a_0[sin(kx) + sin(ky)]$ for the 3-D one. Moreover, the acceleration pulse imparted to the fluids is numerically approximated to a triangular shape with a duration of 26 ms, a peak magnitude of $50\mathbf{g}$, and an integrated impulse of 6.4 m/s.

The variables used to compare the numerical results to the experimental ones are defined in Fig. 6.2, where a, a_b and a_s are the total, bubble and spike amplitudes and $\dot{a}$, $\dot{a}_b$ and $\dot{a}_s$ represent the total, bubble and spike velocities. The 2-D and 3-D cases are numerically solved on Cartesian and unstructured meshes with average grid sizes of $h = 0.005$, resulting in a mesh with 1250 cells for the 2-D test and a mesh with 11250 cells for the 3-D one, and $h = 0.0025$, resulting in a mesh with 4900 cells for the 2-D test and a mesh with 84000 cells for the 3-D one. A fixed time step of 5.0×10^{-4} s is chosen to evolve the discrete equations in time.

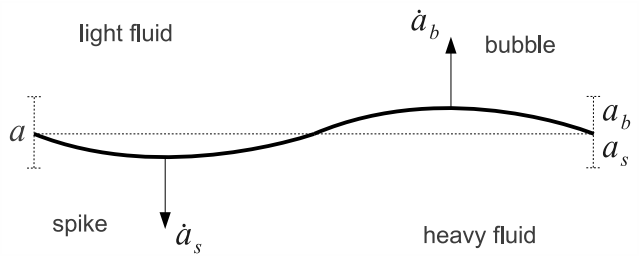


Figure 6.2: Schematic representation of the interface variables used to analyze the RM instability.

6.4.2 Development of the instability

Figs. 6.3 and 6.4 are a sequence of images showing the evolution of the 2-D and 3-D RM instabilities in comparison to the Planar Laser-Induced Fluorescence (PLIF) images from the experiments. In detail, Fig. 6.3 contains two blocks of images showing the evolution of the light (black) and heavy (gray) fluids of the 2-D RM instability at times (relative to the midpoint of spring impact): (first) -14 ms, (second) 102 ms, (third) 353 ms and (fourth) 686 ms. Similarly, Fig. 6.4 shows the evolution of the 3-D RM instability at times (relative to the midpoint of spring impact): (first) -33 ms, (second) 50 ms, (third) 300 ms and (fourth) 633 ms.

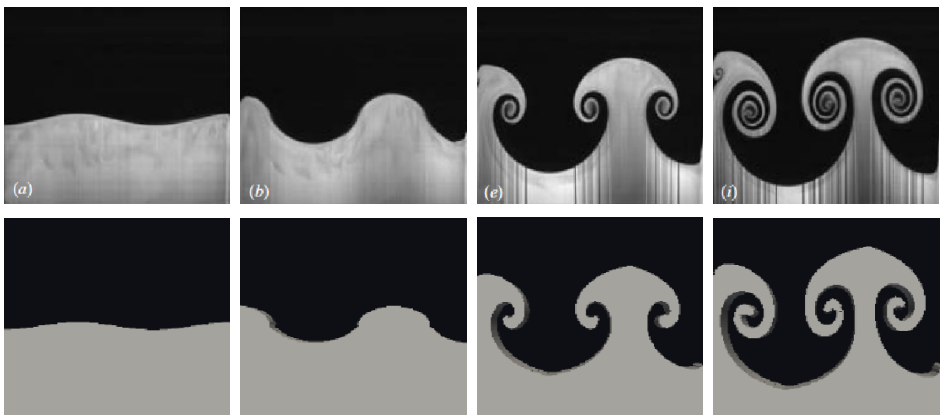


Figure 6.3: Two blocks of images showing the evolution of the light (black) and heavy (gray) fluids of the 2-D RM instability. Top: PLIF images of the experiment by Niederhaus and Jacobs [3]. Bottom: interface reconstruction planes from the numerical simulation.

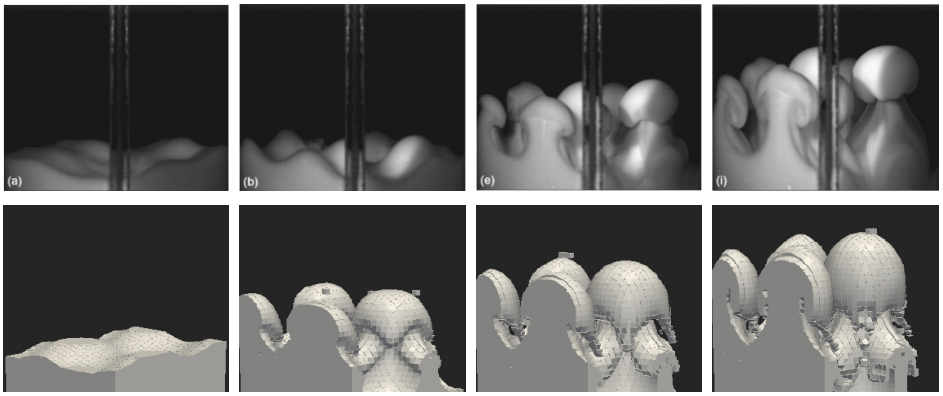


Figure 6.4: Two blocks of images showing the evolution of the light (black) and heavy (gray) fluids of the 3-D RM instability. Top: PLIF images of the experiment by Chapman and Jacobs [4]. Bottom: interface reconstruction planes from the numerical simulation.

The impulsive acceleration in these experiments is directed from the heavier fluid into the lighter one. Thus, the amplitude of the instability changes sign before growing and, immediately after inversion, retains a sinusoidal shape. Though, with time, vortices begin to form, producing the typical mushroom pattern of the RM instability.

6.4.3 Amplitude measurements

In the following figures, Figs. 6.5 and 6.6, the amplitudes, a, of the 2-D and 3-D RM instabilities along time are plotted for the experimental and numerical results. Similarly to the experiments of Niederhaus, Chapman and Jacobs [3, 4], these measurements are made dimensionless by scaling the amplitude with the wave number, k, and time with the wave number and the theoretical initial growth rate, $\dot{a}_0$.

The linear theory that describes the early stages of the RM instability, developed by Richtmyer [1], is shown to be satisfied until nondimensional time $k\dot{a}_0 t = 2 - 3$, both by the experimental results as by the numerical solutions. In contrast, the numerical results of the late time instability's amplitude are not accurate enough. However, it is believed that the numerical results would tend to the experimental ones if the meshes were densified, since this is the pattern shown when meshes are densified from $h = 0.005$ to $h = 0.0025$.

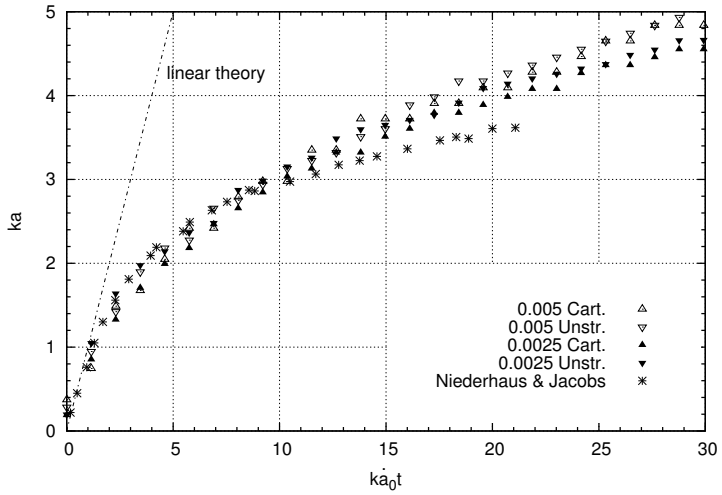


Figure 6.5: Nondimensional amplitude of the instability versus nondimensional time for the 2-D tests.

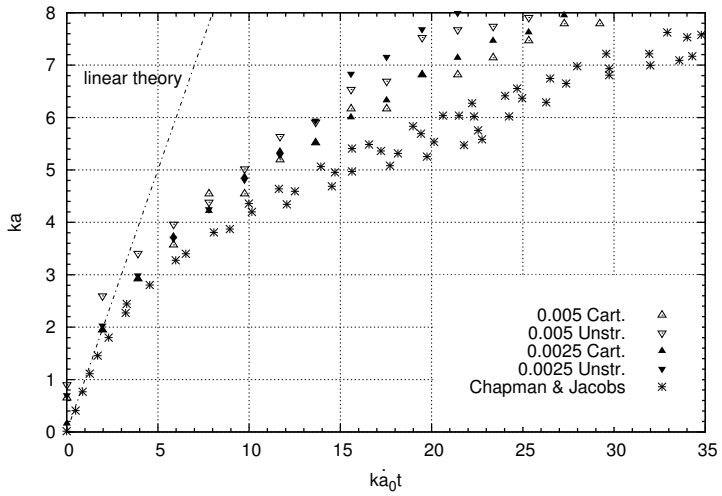


Figure 6.6: Nondimensional amplitude of the instability versus nondimensional time for the 3-D tests.

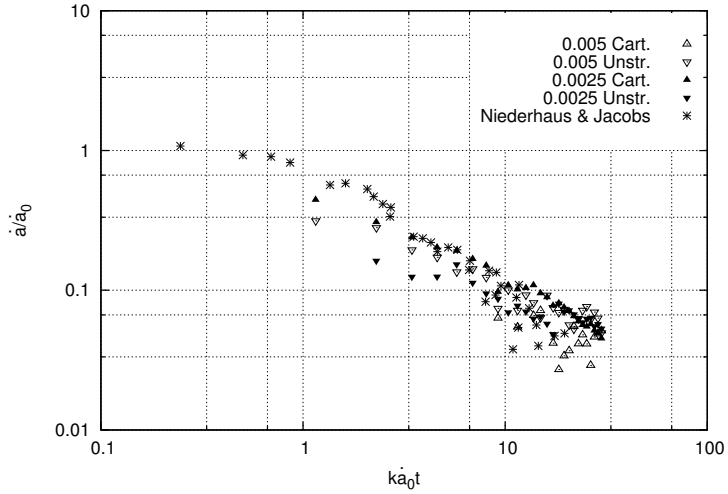


Figure 6.7: Nondimensional velocity versus nondimensional time for the 2-D tests.

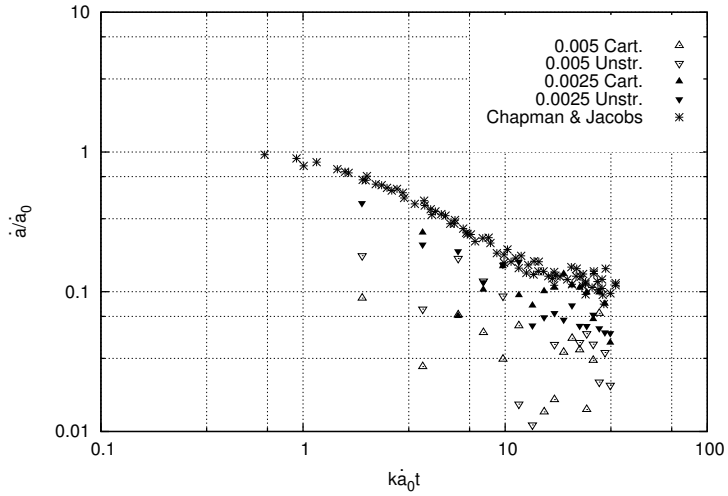


Figure 6.8: Nondimensional velocity versus nondimensional time for the 3-D tests.

6.4.4 Velocity measurements

The results of velocity, $\dot{a}$, for the 2-D and 3-D RM instabilities are plotted along time in Figs. 6.7 and 6.8. The velocity, defined as the average velocity of the bubbles and spikes, is nondimensionalized by dividing it by the theoretical initial growth rate, $\dot{a}_0$, while time is nondimensionalized by multiplying it by the wave number, k, and the theoretical initial growth rate.

The figures show that the numerical results reproduce, in general, the experimental ones, except in the 3-D case with coarse meshes, $h = 0.005$. However, when meshes are densified, $h = 0.0025$, these numerical results tend to the experimental ones; see Fig. 6.8. Moreover, it is observed, from experimental and numerical results, that 3-D velocities present slightly faster nonlinear growth than the 2-D ones. This difference in velocity behavior between 2-D and 3-D cases is due to the configurations of the vorticity fields. In particular, the 2-D vortexs are stationary while the 3-D vortex rings move alternately upward and downward, as respectively shown in Figs. 6.9 and 6.10. As a result, the interface's velocity in the 2-D case decays with time as it is pushed away from the vortex centers. On the contrary, in the 3-D flow the interface's velocity is the sum of a decaying component similar to the 2-D flow and the vortex ring velocity associated to the vortex stretching mechanism. Thus, with time, the interface's velocity approaches the speed of the vortex rings.

6.4.5 Vorticity distributions

The vorticity equation, simplified for incompressible flows of inviscid fluids and restricted to cases with conservative body forces, is written as

$$\frac{D\boldsymbol{\omega}}{Dt} = (\boldsymbol{\omega} \cdot \nabla)\mathbf{u} + \frac{1}{\rho^2}\nabla\rho \times \nabla p, \tag{6.14}$$

where vorticity is defined as $\boldsymbol{\omega} = \nabla \times \mathbf{u}$, the term on the left-hand side is the material derivative of the vorticity vector, the first term on the right-hand side describes the stretching or tilting of vorticity due to the velocity gradients, and the second term is the baroclinic mechanism accounting for the changes in vorticity due to the intersection of density and pressure isosurfaces.

In the RM instabilities, vorticity is created during the impulsive acceleration by the baroclinic term of Eq. 6.14. In detail, the pressure gradient during the impulsive acceleration is hydrostatic and, thus, oriented in the direction of the acceleration, while the density gradient is perpendicular to the fluids interface. Consequently, in the 2-D instability case the distribution of these gradients result in the formation of vortices oriented perpendicular to the viewing plane, as shown in Fig. 6.9. However, in the 3-D case the vorticity results in the distribution of Fig. 6.10, which consists of an array of vortex rings.

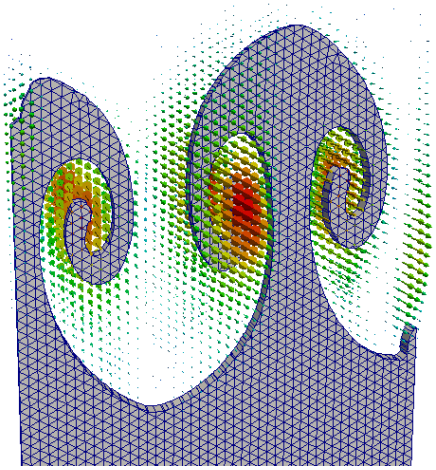


Figure 6.9: Vorticity vectors, VOF reconstruction planes and mesh of the 2-D RM instability test.

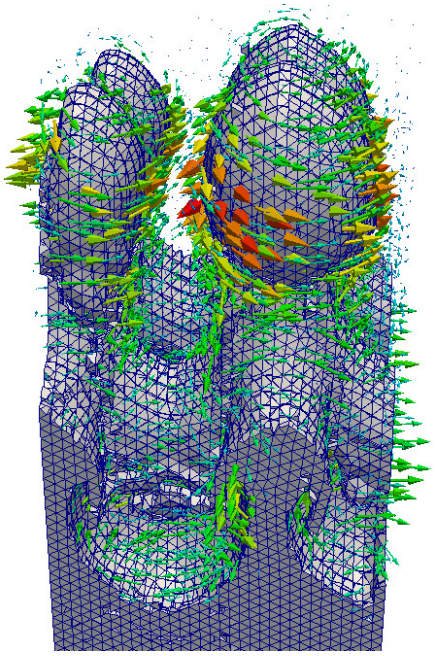


Figure 6.10: Vorticity vectors, VOF reconstruction planes and mesh of the 3-D RM instability test.

The difference in velocity behavior between the 2-D and 3-D cases, which is due to the vortex stretching mechanism, can be visualized by calculating the first term on the right-hand side of Eq. 6.14 by means of Figs. 6.9 and 6.10. In particular, the 2-D case results in a null vector space, since $\boldsymbol{\omega} = (0, 0, \omega_z)$ and the z'th derivatives of the velocity tensor are equal to zero. On the contrary, the 3-D case presents two main vectors with opposite senses at the crests of the bubbles and spikes, indicating that the interface of the instability is being stretched and its amplitude increased by the vortex stretching mechanism.

6.5 Conclusions

In this work, the incompressible RM instability has been simulated by means of a VOF method and a staggered mesh scheme suitable for 3-D unstructured meshes. The numerical simulation has shown the capacity of the discrete system to obtain accurate results of the RM instability initiated with 2-D and 3-D single-mode perturbations. Therefore, this work encourages the authors to test their numerical model on more complex multiphase problems.

The interface-capturing method presents good overall results on Cartesian and unstructured meshes; see Figs. 6.3 and 6.4. In particular, when meshes are densified the numerical results of the instability's amplitude and velocity agree with the experimental data, as shown from Fig. 6.5 to Fig. 6.8.

The analysis of the vorticity distributions, which has been carried out by combining the vorticity equation and the vorticity fields obtained from the numerical results, reveals a main physical difference between the 2-D and 3-D cases. In the 2-D case the stretching term of the vorticity equation due to the velocity gradients is zero, while in the 3-D case the term presents two main vectors with opposite senses at the crests of the bubbles and spikes. As a result, the 2-D interface's velocity decays with time as it is pushed away from the vortex centers, on the contrary, in the 3-D case the interface's velocity approaches that of the vortex ring associated to the vortex stretching mechanism.

Acknowledgements

This work has been financially supported by a FPU Grant from the *Ministerio de Educación, Cultura y Deporte*, Spain (AP-2008-03843) and by *Termo Fluids S.L.*

The authors would like to acknowledge sincerely Jeffrey W. Jacobs, *Journal of Fluid Mechanics* and *Physics of Fluids* for their permission to partially reproduce the PLIF images of the 2-D and 3-D RM instabilities in Figs. 6.3 and 6.4, respectively.

References

[1] R. D. Richtmyer. Taylor Instability in Shock Acceleration of Compressible Fluids. *Communications on Pure and Applied Mathematics*, 13:297–319, 1960.

[2] E. E. Meshkov. Instability of the Interface of Two Gases Accelerated by a Shock Wave. *Fluid Dynamics*, 4:101–104, 1969.

[3] C. E. Niederhaus and J. W. Jacobs. Experimental Study of the Richtmyer-Meshkov Instability of Incompressible Fluids. *Journal of Fluid Mechanics*, 485:243–277, 2003.

[4] P. R. Chapman and J. W. Jacobs. Experiments on the Three-Dimensional Incompressible Richtmyer-Meshkov Instability. *Physics of Fluids*, 18:074101, 2006.

[5] Q. Zhang and S. Sohn. Quantitative Theory of Richtmyer-Meshkov Instability in Three Dimensions. *Zeitschrift für angewandte Mathematik und Physik*, 50:1–46, 1999.

[6] D. Oron, L. Arazi, D. Kartoon, A. Rikanati, U. Alon, and D. Shvarts. Dimensionality Dependence of the Rayleigh-Taylor and Richtmyer-Meshkov Instability Late-Time Scaling Laws. *Physics of Plasmas*, 8:2883–2890, 2001.

[7] V. N. Goncharov. Analytical Model of Nonlinear, Single-Mode, Classical Rayleigh-Taylor Instability at Arbitrary Atwood Numbers. *Physical Review Letters*, 88:134502, 2002.

[8] R. Scardovelli and S. Zaleski. Direct Numerical Simulation of Free-Surface and Interfacial Flow. *Annual Review of Fluid Mechanics*, 31:567–603, 1999.

[9] C. W. Hirt and B. D. Nichols. Volume of fluid (VOF) Method for the Dynamics of Free Boundaries. *Journal of Computational Physics*, 39:201–225, 1981.

[10] 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*, 35:1011–1032, 2006.

[11] 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.

[12] C. M. Rhie and W. L. Chow. Numerical Study of the Turbulent Flow Past an Airfol with Trailing Edge Separation. *AIAA Journal*, 21:1525–1532, 1983.

[13] K. Mahesh, G. Constantinescu, and P. Moin. A Numerical Method for Large-Eddy Simulation in Complex Geometries. *Journal of Computational Physics*, 197:215–240, 2004.

[14] Shashank, J. Larsson, and G. Iaccarino. A Co-located Incompressible Navier-Stokes Solver with Exact Mass, Momentum and Kinetic Energy Conservation in the Inviscid Limit. *Journal of Computational Physics*, 229:4425–4430, 2010.

[15] F. N. Felten and T. S. Lund. Kinetic Energy Conservation Issues Associated with the Collocated Mesh Scheme for Incompressible Flow. *Journal of Computational Physics*, 215:465–484, 2006.

[16] O. Lehmkuhl, I. Rodríguez, R. Borrell, and A. Oliva. Low-Frequency Unsteadiness in the Vortex Formation Region of a Circular Cylinder. *Physics of Fluids*, 25:085109, 2013.

[17] B. Perot. Conservation Properties of Unstructured Staggered Mesh Schemes. *Journal of Computational Physics*, 159:58–89, 2000.

[18] F. H. Harlow and J. E. Welch. Numerical Calculation of Time-Dependent Viscous Incompressible Flow of Fluid with Free Surface. *Physics of Fluids*, 8:2182–2189, 1965.

[19] D. L. Youngs. Time-Dependent Multi-Material Flow with Large Fluid Distortion. In *Numerical Methods for Fluid Dynamics*, pages 273–285. Academic Press, New York, 1982.

[20] A. J. Chorin. Numerical Solution of the Navier-Stokes Equations. *Journal of Computational Physics*, 22:745–762, 1968.

[21] J. R. Shewchuk. An Introduction to the Conjugate Gradient Method without the Agonizing Pain. Carnegie Mellon University, 1994.

7

Conclusions and further research

As written in the abstract, the main objective of this thesis has been to develop a basis for the numerical simulation of multiphase flows of immiscible fluids. In addition, these numerical methods have been designed to be suitable for three-dimensional (3-D) unstructured meshes, as well for Cartesian grids. Therefore, in order to finalize this dissertation, the concluding remarks and future work regarding these issues are presented in this chapter.

7.1 Conclusions

Within the *Heat and Mass Transfer Technological Center* (CTTC) research group, this work is the first approach to the discretization of multiphase flow of immiscible fluids by means of 3-D unstructured meshes. Therefore, the focus is placed on the development of the numerical methods necessary to simulate these flows, rather than on the numerical results corresponding to the physics under consideration. For this purpose, the physical characteristics and the mathematical formulation of the multiphase immiscible flow are presented in Chapter 1. In particular, this chapter demonstrates that, through the use of jump conditions at the interfaces, a single set of mass and momentum conservation equations can be used to describe the flow in a domain composed of different immiscible phases. In consequence, the following chapters develop numerical techniques, first, to locate interfaces as they move and change topology, and second, to correctly discretize the momentum equations that describe the configuration of flows.

In order to capture interfaces on 3-D unstructured meshes, as well on Cartesian grids, a geometrical Volume-of-Fluid (VOF) method, based on a new approach for the multidimensional advection, has been proposed in Chapter 2. In particular, the

method reconstructs interfaces by means of the first-order Parker and Youngs [1] and second-order LVIRA [2] interface reconstruction methods implemented on 3-D unstructured meshes, while the proposed advection step constructs flux polyhedrons by using the Lagrangian trajectories of the cell-vertex velocities. This procedure minimizes the situation of over/underlapping between flux polyhedrons, however, the volume of the polyhedrons needs to be adjusted in order to correctly solve the advection equation. The reconstruction and advection steps of the method have been analyzed by solving various tests on Cartesian and unstructured meshes. In detail, the reconstruction tests show that the proposed methods produce results similar to the ones found in the scientific literature, e.g., Liovic et al. [3] and Ahn and Shashkov [4]. In addition, the tests demonstrate that the Youngs algorithm is first-order accurate and exhibits better results on coarse grids, while LVIRA is second-order accurate and performs better when the grid is refined, but requires more computational time since it performs a 2-D minimization. Moreover, results of the advection tests on Cartesian grids are similar to the ones presented by Hernández et al. [5] and Liovic et al. [3], while results on unstructured meshes are of same order of magnitude as the Cartesian ones. Consequently, this demonstrates that the proposed unsplit advection algorithm solves correctly the advection equation both on Cartesian as on unstructured meshes. Furthermore, independently of the type of mesh used, the tests show that the use of the Youngs reconstruction method turns out in a first-order advection algorithm, while the use of the LVIRA method tends to produce a second-order one.

Complementing the VOF method, Chapter 3 develops a new parallelization strategy. It has been developed with the aim of overcoming the workload imbalance obtained with the standard domain decomposition (DD) when the fluids interface is not homogeneously distributed throughout the domain. Basically, it consists in a load balancing (LB) process, complementary to the underlying DD, that reassigns tasks from processes with higher workload to processes with lower workload. This process is applied separately to the reconstruction and advection steps of the VOF algorithm. Moreover, since the initial DD is surpassed and the algorithm is applied to general unstructured discretizations, all the geometric and algebraic data required to perform any reassigned task need to be transmitted with it. Several tests have been performed in the MareNostrum-III supercomputer [6], engaging up to 1024 CPU-cores. The results show that the parallel efficiency of the DD strategy depends only on the interface distribution within the domain. On the contrary, our LB strategy overcomes the imbalance, but the redistribution cost cancels part of the gains achieved from it. However, when directly comparing both strategies, the result is that the larger the initial imbalance, the larger the speedup achieved by the LB algorithm respect to the DD one. In detail, gains up to $\sim 12\times$ for the most ill-conditioned situations are observed, but even in situations where the interface is spread throughout the domain, the gain achieved does not drop below $1\times$.

The finite-volume discretization of the continuity and Navier-Stokes equations in the case of single-phase flow is dealt in Chapter 4. The discretization is presented by means of a collocated and two staggered mesh schemes suitable for 3-D unstructured meshes. In particular, the collocated discretization corresponds to the scheme extensively used by Lehmkuhl et al. [7,8] and Rodríguez et al. [9,10], while the staggered discretizations are the one introduced by Perot et al. [11,12] (*a*) and a self-developed one that proposes a least-squares cell-centered velocity interpolation (*b*). The theoretical analysis of their conservation properties demonstrates that, given that the continuity and Navier-Stokes equations are specifically derived to conserve mass and momentum, the collocated and staggered schemes presented also conserve discretely these properties. On the other hand, the analysis of the kinetic energy conservation — which is a really important property when solving turbulent flows, since the energy is convected from the large eddies to the small dissipative scales — results in two different behaviors depending upon the mesh scheme. In detail, the analysis demonstrates that, in the absence of viscosity ($\mu = 0$) and utilizing a symmetry-preserving convection scheme [13], the change in kinetic energy is due to the fluxes through the boundaries of the domain for the staggered schemes, plus an error from the pressure term for the collocated scheme. This pressure error term arises from the special definition of the normal face velocity needed to exactly conserve mass. These theoretical results are numerically proved by solving a Rankine vortex. This test shows that the staggered mesh schemes preserve numerically mass, momentum and kinetic energy, while the collocated scheme conserves mass and momentum, but presents a kinetic energy error of the form $\mathcal{O}(\Delta t^m, \Delta h^2)$. Thus, densifying meshes and using small time steps or high-order temporal schemes decreases the collocated kinetic energy error. Moreover, an accuracy study for the different mesh schemes is performed by comparing numerical results to the analytical solution of an exact sinusoidal function. The results show that collocated and staggered *b* accuracy errors are nearly second-order, while the staggered *a* scheme presents first-order ones. Therefore, the authors conclude that if incompressible turbulent flow is to be solved, using time-explicit algorithms with fine unstructured meshes and small time steps, the collocated scheme is a better option over the staggered ones: (1) the pressure kinetic energy error is unnoticeable in such situations; (2) presents good accuracy; (3) it is a fast scheme that does not need the calculation of circumcenters. However, the use of the collocated scheme to solve problems regarding other fluid or flow characteristics, e.g., multiphase flow, combustion problems, or others, may produce spurious pressure modes (checkerboard). In these situations the staggered schemes presented in this study are a good alternative, especially the staggered *b* mesh discretization, since it presents better accuracy than the staggered *a* one, although, it requires a more complicated and computationally demanding cell-centered velocity reconstruction.

After analyzing the case of single-phase flow, in Chapter 5 the study is extended to the case of different fluids separated by interfaces. Similar to the previous case, two unstructured finite-volume mesh discretizations are proposed, collocated and staggered, that numerically conserve mass and momentum, while at the same time minimize the errors in the conservation of kinetic energy. On the one hand, the collocated and staggered discretizations are shown to conserve mass exactly, while the equation of momentum conservation states that the change in momentum of both discretizations is due to the fluxes through the boundary of the domain and the source terms. On the other hand, the discrete conservations of kinetic energy for the collocated and staggered mesh schemes have been derived, stating that, if a symmetry-preserving convection scheme [13] is used and in the absence of viscosity ($\mu = 0$) and source terms, the change in kinetic energy is due solely to the fluxes through the boundary of the domain for the staggered discretization, plus a kinetic energy error from the pressure term for the collocated one. This error in the conservation of kinetic energy — intrinsic to the collocated formulation, since it arises from the difference in pressure gradient evaluations between the calculations of cell-centered velocities and face mass fluxes — is shown to be proportional to the density ratio and scaled by the mesh size and time step as $\mathcal{O}(\Delta t^m, \Delta h^2)$. In order to numerically verify the theoretical conservation properties, a three-dimensional vortex is solved. The test corroborates that both collocated and staggered discretizations conserve mass and momentum numerically. Moreover, the test demonstrates that the use of an upwind convection scheme [14] produces an artificial kinetic energy dissipation, while using a symmetry-preserving one turns out in a zero contribution to the kinetic energy equation. This result can be extrapolated to most high-order convection schemes, e.g., QUICK [15], ENO [16] or WENO [17], since they are all based on upwind approximations and, hence, disregard symmetry properties. Additionally, it is proved numerically that the staggered discretization preserves kinetic energy, while the collocated one presents a kinetic energy error that, as theoretically expected, is proportional to density ratio and decreases with mesh size and time step. Once again, the accuracy of both mesh discretizations is analyzed by means of comparing numerical results to the analytical solution of an exact sinusoidal function. The test concludes that collocated errors are almost second-order, while, as imposed by construction, staggered ones are just first-order. Therefore, summarizing, the collocated scheme is more accurate and presents no geometric difficulties (no circumcenters are needed), while the staggered scheme numerically preserves kinetic energy and is more stable (do not display spurious pressure modes).

Finally, Chapter 6 performs a general assessment of the numerical methods developed in order to capture interfaces and resolve the momentum equations on 3-D unstructured meshes. The methods are tested by numerically simulating the Richtmyer-Meshkov (RM) instability [18,19] of two incompressible immiscible liquids.

In particular, the instability has been simulated by means of the VOF method and the staggered mesh scheme. The numerical simulation has shown the capacity of the discrete system to obtain accurate results of the RM instability. Therefore, the results of this work are encouraging in terms of testing the numerical model on more complex multiphase problems. In detail, the interface-capturing method presents good overall results on Cartesian and unstructured meshes, since, when meshes are densified, the numerical results of the instability's amplitude and velocity agree with the experimental data provided by Niederhaus, Chapman and Jacobs [20, 21]. Moreover, the analysis of the vorticity distributions, which has been carried out by combining the vorticity equation and the vorticity fields obtained from the numerical results, reveals a main physical difference between the 2-D and 3-D cases. In the 2-D case the stretching term of the vorticity equation due to the velocity gradients is zero, while in the 3-D case the term presents two main vectors with opposite senses at the crests of the bubbles and spikes. As a result, the 2-D interface's velocity decays with time as it is pushed away from the vortex centers, on the contrary, in the 3-D case the interface's velocity approaches that of the vortex ring associated to the vortex stretching mechanism.

7.2 Further research

The main objective of this thesis is to set the basis for the numerical simulation of multiphase flow of immiscible fluids on complex geometries. Although this objective has been accomplished, there are still issues to be considered. For instance, the development of other interface-capturing methods, the detailed treatment of surface tension forces, the analysis of hybrid convection schemes suitable for fluids with interfaces, the implementation of faster and more robust Poisson's pressure solvers, the necessity to incorporate adaptive mesh refinement (AMR) methods or the consideration of phase change phenomena. Some of these issues are being currently investigated by other researchers of the CTTC together with my collaboration, while others are still in a very initial state. In detail, a conservative Level-Set (CLS) method [22], capable of capturing interfaces on 3-D unstructured meshes, and an accurate surface tension calculation [23] have been implemented and succesfully tested within the *TermoFluids* (TF) CFD platform [24]. Moreover, in order to perform direct numerical simulations (DNS) of multiphase immiscible flow, the focus is being placed on developing: (1) hybrid convective schemes on the basis of symmetry-preserving upwind discretizations suitable for unstructured meshes [14]; (2) conjugate-gradient (CG) multigrid Poisson's pressure solvers [25, 26]; (3) efficient parallel AMR methods for improving the accuracy on capturing interfaces between fluids [27]. Finally, if considering the discretization of the phenomena related to phase change and energy, a new whole world opens which has not been explored much yet.

References

[1] B. Parker and D. Youngs. Two and Three Dimensional Eulerian Simulation of Fluid Flow with Material Interfaces. Technical Report 01/92, UK Atomic Weapons Establishment, 1992.

[2] J. E. Pilliod and E. G. Puckett. Second-Order Volume-of-Fluid Algorithms for Tracking Material Interfaces. Technical Report LBNL-40744, Lawrence Berkeley National Laboratory, 1997.

[3] 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*, 35:1011–1032, 2006.

[4] H. T. Ahn and M. Shashkov. Multi-Material Interface Reconstruction on Generalized Polyhedral Meshes. *Journal of Computational Physics*, 226:2096–2132, 2007.

[5] 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*, 58:897–921, 2008.

[6] Barcelona Supercomputing Center. Webpage: http://www.bsc.es.

[7] O. Lehmkuhl, I. Rodríguez, A. Báez, A. Oliva, and C. D. Pérez-Segarra. On the Large-Eddy Simulations for the Flow Around Aerodynamic Profiles Using Unstructured Grids. *Computers & Fluids*, 84:176–189, 2013.

[8] O. Lehmkuhl, I. Rodríguez, R. Borrell, and A. Oliva. Low-Frequency Unsteadiness in the Vortex Formation Region of a Circular Cylinder. *Physics of Fluids*, 25:085109, 2013.

[9] I. Rodríguez, R. Borrell, O. Lehmkuhl, C. D. Pérez-Segarra, and A. Oliva. Direct Numerical Simulation of the Flow over a Sphere at $Re = 3700$. *Journal of Fluid Mechanics*, 679:263–287, 2011.

[10] I. Rodríguez, O. Lehmkuhl, R. Borrell, and A. Oliva. Direct Numerical Simulation of a NACA 0012 in Full Stall. *International Journal of Heat and Fluid Flow*, 43:194–203, 2013.

[11] B. Perot. Conservation Properties of Unstructured Staggered Mesh Schemes. *Journal of Computational Physics*, 159:58–89, 2000.

[12] X. Zhang, D. Schmidt, and B. Perot. Accuracy and Conservation Properties of a Three-Dimensional Unstructured Staggered Mesh Scheme for Fluid Dynamics. *Journal of Computational Physics*, 176:764–791, 2002.

[13] R. W. C. P. Verstappen and A. E. P. Veldman. Symmetry-Preserving Discretization of Turbulent Flow. *Journal of Computational Physics*, 187:343–368, 2003.

[14] A. E. P. Veldman and K. Lam. Symmetry-Preserving Upwind Discretization of Convection on Non-Uniform Grids. *Applied Numerical Mathematics*, 58:1881–1891, 2008.

[15] B. P. Leonard. A Stable and Accurate Convective Modelling Procedure Based on Quadratic Upstream Interpolation. *Computer Methods in Applied Mechanics and Engineering*, 19:59–98, 1979.

[16] A. Harten, B. Engquist, S. Osher, and S. R. Chakravarthy. Uniformly High Order Accurate Essentially Non-Oscillatory Schemes, III. *Journal of Computational Physics*, 71:231–303, 1987.

[17] X. D. Liu, S. Osher, and T. Chan. Weighted Essentially Non-Oscillatory Schemes. *Journal of Computational Physics*, 115:200–212, 1994.

[18] R. D. Richtmyer. Taylor Instability in Shock Acceleration of Compressible Fluids. *Communications on Pure and Applied Mathematics*, 13:297–319, 1960.

[19] E. E. Meshkov. Instability of the Interface of Two Gases Accelerated by a Shock Wave. *Fluid Dynamics*, 4:101–104, 1969.

[20] C. E. Niederhaus and J. W. Jacobs. Experimental Study of the Richtmyer-Meshkov Instability of Incompressible Fluids. *Journal of Fluid Mechanics*, 485:243–277, 2003.

[21] P. R. Chapman and J. W. Jacobs. Experiments on the Three-Dimensional Incompressible Richtmyer-Meshkov Instability. *Physics of Fluids*, 18:074101, 2006.

[22] N. Balcázar, L. Jofre, O. Lehmkuhl, J. Castro, and J. Rigola. A Finite-Volume/Level-Set Method for Simulating Two-Phase Flows on Unstructured Grids. *International Journal of Multiphase Flow*, 64:55–72, 2014.

[23] N. Balcázar, L. Jofre, O. Lehmkuhl, and A. Oliva. A Combined Volume-of-Fluid/Level-Set Method for the Simulation of Surface-Tension-Driven Interfacial Flows. *International Journal of Multiphase Flow*, To be submitted, 2014.

[24] O. Lehmkuhl, C. D. Pérez-Segarra, R. Borrell, M. Soria, and A. Oliva. TERMOFLUIDS: A New Parallel Unstructured CFD Code for the Simulation of Turbulent Industrial Problems on Low Cost PC Cluster. In *Proceedings of the Parallel CFD Conference*, pages 1–8, 2007.

[25] M.W. Gee, C.M. Siefert, J.J. Hu, R.S. Tuminaro, and M.G. Sala. ML 5.0 Smoothed Aggregation User's Guide. Technical Report SAND2006-2649, Sandia National Laboratories, 2006.

[26] M.A. Heroux. Aztec00 User Guide. Technical Report SAND2004-3796, Sandia National Laboratories, 2007.

[27] O. Antepara, O. Lehmkuhl, R. Borrell, J. Chiva, and A. Oliva. Parallel Adaptive Mesh Refinement for Large-Eddy Simulations of Turbulent Flows. *Computers & Fluids*, Under Review, 2014.

Appendix A

Discretization of the convection-diffusion equation

This appendix presents the discretization of the general convection-diffusion differential equation, based on the finite-volume formulation and suitable for three-dimensional unstructured meshes composed of arbitrary convex polyhedra, as well for Cartesian grids.

A.1 Convection-diffusion equation

The transport equations of Computational Fluid Dynamics (CFD) and Heat Transfer (HT) can all be expressed by the general convection-diffusion equation [1]. In detail, if the dependent variable is denoted by ϕ, the general differential equation is written as

$$\frac{\partial(\rho\phi)}{\partial t} + \nabla \cdot (\rho \mathbf{u}\phi) = \nabla \cdot (\Gamma \nabla\phi) + S, \tag{A.1}$$

where ρ, t and $\mathbf{u}$ correspond to density, time and velocity, respectively, while Γ is the diffusion coefficient and S the source term. It may be observed, from the above equation, that the general convection-diffusion equation is composed of four terms, which are named correspondingly from left to right as: unsteady, convection, diffusion and source.

A.2 Finite-volume unstructured discretization

First, integrating Eq. A.1 over the volume of a cell c and using a first-order explicit time scheme gives

$$\int_{\Omega_c} \frac{(\rho\phi)^{n+1} - (\rho\phi)^n}{\Delta t}\, dV + \int_{\Omega_c} \nabla \cdot (\rho \mathbf{u}\phi)^n\, dV = \int_{\Omega_c} \nabla \cdot (\Gamma \nabla \phi)^n\, dV + \int_{\Omega_c} S^n\, dV, \tag{A.2}$$

where superscript n refers to time instant and Δt is the time step.

Second, considering the finite-volume hypothesis and applying the divergence theorem to the bordering faces of cell c, $f \in F(c)$, results in Eq. A.2 discretized as

$$\frac{(\rho\phi)^{n+1} - (\rho\phi)^n}{\Delta t} V_c + \sum_{f \in F(c)} (\rho_f \mathbf{u}_f \phi_f)^n \cdot \hat{\mathbf{n}}_f A_f = \sum_{f \in F(c)} (\Gamma_f \nabla \phi_f)^n \cdot \hat{\mathbf{n}}_f A_f + S^n\, V_c, \tag{A.3}$$

where $\hat{\mathbf{n}}_f$ and A_f correspond to the normal outward unit vector and surface of face f, respectively, and V_c is the volume of the cell under consideration.

A.3 Evaluation of the convection term

The finite-volume discretization of the convection term, see Eq. A.3, is

$$C = \sum_{f \in F(c)} \rho_f \mathbf{u}_f \phi_f \cdot \hat{\mathbf{n}}_f A_f = \sum_{f \in F(c)} (\rho_f \mathbf{u}_f \cdot \hat{\mathbf{n}}_f A_f) \phi_f = \sum_{f \in F(c)} \hat{M}_f \phi_f, \tag{A.4}$$

being $\hat{M}_f$ the mass flow and ϕ_f the value of ϕ at face f evaluated by a convective numerical scheme.

There are many different convective numerical schemes, some of them listed in the work by Pérez-Segarra et al. [2], for example (see Fig. A.1):

- Upwind:

$$\begin{aligned} \phi_f &= \phi_P, \quad if \quad \hat{M}_f \geq 0 \\ \phi_f &= \phi_F, \quad if \quad \hat{M}_f < 0 \end{aligned} \tag{A.5}$$

- Symmetry-preserving:

$$\phi_f = \frac{1}{2}(\phi_P + \phi_F) \tag{A.6}$$

A.4 Evaluation of the diffusion term

The finite-volume discretization of the diffusion term, see Eq. A.3, is

$$D = \sum_{f \in F(c)} \Gamma_f \nabla \phi_f \cdot \hat{\mathbf{n}}_f A_f. \tag{A.7}$$

On of the most common approaches to calculate this term is the Direct Gradient Evaluation (DGE) described in the work by Pérez-Segarra et al. [2], which evaluates the gradient directly at the cell face as (see Fig. A.1)

$$D = \sum_{f \in F(c)} \Gamma_f \nabla \phi_f \cdot \hat{\mathbf{n}}_f A_f = \sum_{f \in F(c)} \Gamma_f \left(\frac{\partial \phi}{\partial n} \right)_f A_f \approx \sum_{f \in F(c)} \Gamma_f \frac{\phi_{F'} - \phi_{P'}}{\mathbf{PF} \cdot \hat{\mathbf{n}}_f} A_f, \tag{A.8}$$

where **PF** is the vector between nodes P and F, and $\phi_{P'}$ and $\phi_{F'}$ are the projections of the nodal values on the normal surface vector direction, $\mathbf{n}_f$, estimated by the gradient at the nodal position as

$$\phi_{P'} \approx \phi_P + \nabla \phi_P \cdot \mathbf{PP'} \qquad \text{and} \qquad \phi_{F'} \approx \phi_F + \nabla \phi_F \cdot \mathbf{FF'}. \tag{A.9}$$

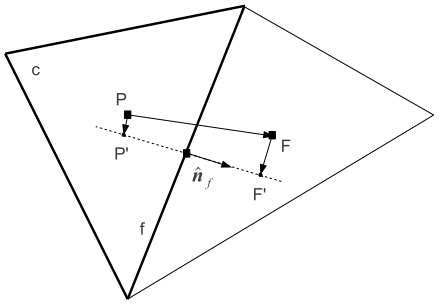


Figure A.1: Schematic representation of the geometric parameters needed for the evaluation of the convection and diffusion terms.

References

[1] S. V. Patankar. *Numerical Heat Transfer and Fluid Flow*. Hemisphere Publishing Corporation, 1980.

[2] C. D. Pérez-Segarra, C. Farré, J. Cadafalch, and A. Oliva. Analysis of Different Numerical Schemes for the Resolution of Convection-Diffussion Equations Using Finite-Volume Methods on Three-Dimensional Unstructured Grids. Part I: Discretization Schemes. *Numerical Heat Transfer, Part B*, 49:333–350, 2006.

Appendix B

Vector calculus identities

This appendix presents a list of important vector calculus identities.

B.1 Operator notation

Notation of the symbols and differential operators used in vector calculus.

B.1.1 Nabla

The nabla symbol, ∇, can be interpreted as a vector of partial derivative operators, and its three possible meanings — gradient, divergence and curl — can be formally viewed as the scalar, dot and cross products, respectively, of the ∇ operator with the field. In the three-dimensional Cartesian coordinate system $\mathbb{R}^3$ with coordinates (x, y, z), ∇ is defined in terms of partial derivative operators as

$$\nabla = \hat{\mathbf{x}}\frac{\partial}{\partial x} + \hat{\mathbf{y}}\frac{\partial}{\partial y} + \hat{\mathbf{z}}\frac{\partial}{\partial z}, \tag{B.1}$$

where $\{\hat{\mathbf{x}}, \hat{\mathbf{y}}, \hat{\mathbf{z}}\}$ are the unit vectors in their respective directions.

B.1.2 Gradient

The gradient of a tensor field, T, of order n, is generally written as

$$grad(\mathsf{T}) = \nabla\mathsf{T}, \tag{B.2}$$

and is a tensor field of order $n + 1$. In particular, if the tensor field has order 0, i.e., a scalar, the resulting gradient is a vector field.

B.1.3 Divergence

The divergence of a tensor field, T, of nonzero order n, is generally written as

$$div(\mathsf{T}) = \nabla \cdot \mathsf{T}, \tag{B.3}$$

and is a contraction to a tensor field of order $n-1$. For instance, the divergence of a vector is a scalar. The divergence of a higher order tensor field may be found by decomposing the tensor field into a sum of outer products, thereby allowing the use of the identity

$$\nabla \cdot (\mathbf{a} \otimes \mathsf{T}) = \mathsf{T}(\nabla \cdot \mathbf{a}) + (\mathbf{a} \cdot \nabla)\mathsf{T}, \tag{B.4}$$

where $\mathbf{a} \cdot \nabla$ is the directional derivative in the direction of $\mathbf{a}$ multiplied by its magnitude. Specifically, for the outer product of two vectors, $\mathbf{a}$ and $\mathbf{b}$,

$$\nabla \cdot (\mathbf{a} \otimes \mathbf{b}) = \nabla \cdot (\mathbf{a}\mathbf{b}^T) = \mathbf{b}(\nabla \cdot \mathbf{a}) + (\mathbf{a} \cdot \nabla)\mathbf{b}. \tag{B.5}$$

B.1.4 Curl

The curl of a three-dimensional vector field, $\mathbf{a}$, is generally written as

$$curl(\mathbf{a}) = \nabla \times \mathbf{a}, \tag{B.6}$$

and is also a three-dimensional vector field.

B.1.5 Laplacian

The laplacian of a tensor field, T, is generally written as

$$\Delta\mathsf{T} = \nabla^2\mathsf{T} = (\nabla \cdot \nabla)\mathsf{T}, \tag{B.7}$$

and is a tensor field of the same order.

B.2 Operator identities

In the remainder of this section, ψ and ϕ are scalars while $\mathbf{a}$ and $\mathbf{b}$ are vectors.

B.2.1 Distributive properties

$$\nabla(\psi + \phi) = \nabla\psi + \nabla\phi \tag{B.8}$$
$$\nabla \cdot (\mathbf{a} + \mathbf{b}) = \nabla \cdot \mathbf{a} + \nabla \cdot \mathbf{b} \tag{B.9}$$
$$\nabla \times (\mathbf{a} + \mathbf{b}) = \nabla \times \mathbf{a} + \nabla \times \mathbf{b} \tag{B.10}$$

B.2.2 Product rules

$$\nabla(\psi\phi) = \phi\nabla\psi + \psi\nabla\phi \tag{B.11}$$
$$\nabla(\mathbf{a}\cdot\mathbf{b}) = (\mathbf{a}\cdot\nabla)\mathbf{b} + (\mathbf{b}\cdot\nabla)\mathbf{a} + \mathbf{a}\times(\nabla\times\mathbf{b}) + \mathbf{b}\times(\nabla\times\mathbf{a}) \tag{B.12}$$
$$\nabla\cdot(\psi\mathbf{a}) = (\nabla\psi)\cdot\mathbf{a} + \psi(\nabla\cdot\mathbf{a}) \tag{B.13}$$
$$\nabla\cdot(\mathbf{a}\times\mathbf{b}) = \mathbf{b}\cdot(\nabla\times\mathbf{a}) - \mathbf{a}\cdot(\nabla\times\mathbf{b}) \tag{B.14}$$
$$\nabla\times(\psi\mathbf{a}) = (\nabla\psi)\times\mathbf{a} + \psi(\nabla\times\mathbf{a}) \tag{B.15}$$
$$\nabla\times(\mathbf{a}\times\mathbf{b}) = \mathbf{a}\,(\nabla\cdot\mathbf{b}) - \mathbf{b}\,(\nabla\cdot\mathbf{a}) + (\mathbf{b}\cdot\nabla)\,\mathbf{a} - (\mathbf{a}\cdot\nabla)\,\mathbf{b} \tag{B.16}$$

B.2.3 Second derivatives

$$\nabla\times(\nabla\psi) = 0 \tag{B.17}$$
$$\nabla\cdot(\nabla\times\mathbf{a}) = 0 \tag{B.18}$$
$$\nabla\times(\nabla\times\mathbf{a}) = \nabla(\nabla\cdot\mathbf{a}) - \Delta\mathbf{a} \tag{B.19}$$
$$\Delta(\nabla\cdot\mathbf{a}) = \nabla\cdot(\Delta\mathbf{a}) \tag{B.20}$$
$$\nabla\cdot(\psi\nabla\phi) = \psi\Delta\phi + \nabla\psi\cdot\nabla\phi \tag{B.21}$$
$$\psi\nabla^2\phi - \phi\nabla^2\psi = \nabla\cdot(\psi\nabla\phi - \phi\nabla\psi) \tag{B.22}$$
$$\Delta(\psi\phi) = \psi\Delta\phi + 2\nabla\psi\cdot\nabla\phi + \phi\Delta\psi \tag{B.23}$$

B.3 Vector identities

In this section, letters **a**, **b**, **c** and **d** represent vectors.

$$\mathbf{a} + \mathbf{b} = \mathbf{b} + \mathbf{a} \tag{B.24}$$
$$\mathbf{a}\cdot\mathbf{b} = \mathbf{b}\cdot\mathbf{a} \tag{B.25}$$
$$\mathbf{a}\times\mathbf{b} = \text{-}\mathbf{b}\times\mathbf{a} \tag{B.26}$$
$$(\mathbf{a}+\mathbf{b})\cdot\mathbf{c} = \mathbf{a}\cdot\mathbf{c} + \mathbf{b}\cdot\mathbf{c} \tag{B.27}$$
$$(\mathbf{a}+\mathbf{b})\times\mathbf{c} = \mathbf{a}\times\mathbf{c} + \mathbf{b}\times\mathbf{c} \tag{B.28}$$
$$\mathbf{a}\cdot(\mathbf{b}\times\mathbf{c}) = \mathbf{b}\cdot(\mathbf{c}\times\mathbf{a}) = \mathbf{c}\cdot(\mathbf{a}\times\mathbf{b}) \tag{B.29}$$
$$\mathbf{a}\times(\mathbf{b}\times\mathbf{c}) = (\mathbf{a}\cdot\mathbf{c})\mathbf{b} - (\mathbf{a}\cdot\mathbf{b})\mathbf{c} \tag{B.30}$$
$$(\mathbf{a}\times\mathbf{b})\cdot(\mathbf{c}\times\mathbf{d}) = (\mathbf{a}\cdot\mathbf{c})(\mathbf{b}\cdot\mathbf{d}) - (\mathbf{b}\cdot\mathbf{c})(\mathbf{a}\cdot\mathbf{d}) \tag{B.31}$$
$$(\mathbf{a}\cdot(\mathbf{b}\times\mathbf{c}))\mathbf{d} = (\mathbf{a}\cdot\mathbf{d})(\mathbf{b}\times\mathbf{c}) + (\mathbf{b}\cdot\mathbf{d})(\mathbf{c}\times\mathbf{a}) + (\mathbf{c}\cdot\mathbf{d})(\mathbf{a}\times\mathbf{b}) \tag{B.32}$$
$$(\mathbf{a}\times\mathbf{b})\times(\mathbf{c}\times\mathbf{d}) = (\mathbf{a}\cdot(\mathbf{b}\times\mathbf{d}))\mathbf{c} - (\mathbf{a}\cdot(\mathbf{b}\times\mathbf{c}))\mathbf{d} \tag{B.33}$$

B.4 Integration identities

In the remainder of this section, ψ and ϕ are scalars while $\mathbf{a}$ is a vector.

B.4.1 Volume-surface integrals

In the following volume-surface integral theorems, V denotes a three-dimensional volume with a corresponding two-dimensional closed boundary, $S = \partial V$, and normal outward unit vector, $\hat{\mathbf{n}}$.

- Divergence theorem:

$$\int_V (\nabla \cdot \mathbf{a})\, dV = \oint_{\partial V} (\mathbf{a} \cdot \hat{\mathbf{n}})\, dS \tag{B.34}$$

$$\int_V \nabla \psi dV = \oint_{\partial V} (\psi \hat{\mathbf{n}})\, dS \tag{B.35}$$

$$\int_V (\nabla \times \mathbf{a})\, dV = \oint_{\partial V} (\hat{\mathbf{n}} \times \mathbf{a})\, dS \tag{B.36}$$

- Green's first identity:

$$\int_V (\psi \Delta \phi + \nabla \phi \cdot \nabla \psi)\ dV = \oint_{\partial V} \psi\, (\nabla \phi \cdot \hat{\mathbf{n}})\ dS \tag{B.37}$$

- Green's second identity:

$$\int_V (\psi \Delta \phi - \phi \Delta \psi)\, dV = \oint_{\partial V} [(\psi \nabla \phi - \phi \nabla \psi) \cdot \hat{\mathbf{n}}]\, dS \tag{B.38}$$

B.4.2 Surface-curve integrals

In the following surface-curve integral theorems, S denotes a two-dimensional surface with a corresponding one-dimensional closed boundary, $C = \partial S$, and tangential counterclockwise unit vector, $\hat{\mathbf{l}}$.

- Stokes' theorem:

$$\int_S ((\nabla \times \mathbf{a}) \cdot \hat{\mathbf{n}})\, dS = \oint_{\partial S} (\mathbf{a} \cdot \hat{\mathbf{l}})\, dC \tag{B.39}$$

$$\int_S (\hat{\mathbf{n}} \times \nabla \psi)\, dS = \oint_{\partial S} (\psi \hat{\mathbf{l}})\, dC \tag{B.40}$$

Appendix C

Parallel computing resources

This appendix lists the different parallel computing systems where the numerical codes developed in the context of this thesis have been executed. In order to gain acces to some of these equipments, it is required to submit state-of-the-art science projects to competitive international calls, in which both the scientific relevance of the presented project and the parallel performance of the code are evaluated.

C.1 JFF supercomputer, Terrassa

The JFF supercomputer, from the *Heat and Mass Transfer Technological Center* (CTTC), is a HPC Beowulf cluster consisting in 40 cluster nodes, each one containing 2 AMD Opteron with 16 Cores for each CPU, linked with 64 Gigabytes of RAM memory and an infiniband QDR 4X network interconnection between nodes with latencies of 1.07 microseconds with a 40Gbits/s bandwith. For detailed information, visit the website of the center: http://www.cttc.upc.edu

C.2 MareNostrum supercomputer, Barcelona

The MareNostrum supercomputer, located in the *Barcelona Supercomputing Center* (BSC), is based on Intel SandyBridge 8-core processors at 2.6 GHz (2 per node), iDataPlex Compute Racks, a Linux Operating System and an Infiniband FDR10 interconnection network, resulting in a peak performance of 1.1 Petaflops and 100.8 TB of main memory. Visit the website for more information: http://www.bsc.es

Figure C.1: JFF supercomputer.

Figure C.2: MareNostrum supercomputer.

C.3 Curie supercomputer, Paris

The Curie supercomputer, owned by the *Grand Equipement National de Calcul Intensif* (GENCI) and operated into the *Très Grand Centre de Calcul* (TGCC) by the *Commissariat à l'Énergie Atomique* (CEA), is a Tier0 system open to scientists through the French participation into the *Partnership for Advanced Computing in Europe* (PRACE) research infrastructure. Curie offers three different fractions of x86-64 computing resources, from which the **Curie Fat Nodes** is used in this thesis. The **Curie Fat Nodes** cluster is composed of 360 S6010 bullx nodes, each one with 4 8-core x86-64 CPU-cores, 128 GB of memory and 1 local disk of 2TB. For detailed information of this supercomputer, see the website: http://www-hpc.cea.fr/en/complexe/tgcc-curie.htm

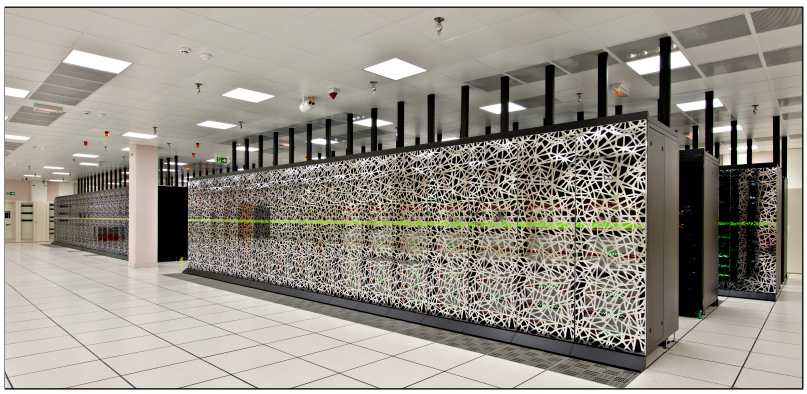


Figure C.3: Curie supercomputer.

Appendix D

Main publications in the context of this thesis

This is an exhaustive list of the publications, on international journals and conferences, carried out within the framework of this thesis.

D.1 Journal Papers

- 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.
- L. Jofre, R. Borrell, O. Lehmkuhl, and A. Oliva. Parallel Load Balancing Strategy for Volume-of-Fluid Methods on 3-D Unstructured Meshes. Under review in *Journal of Computational Physics*, 2014.
- L. Jofre, O. Lehmkuhl, J. Ventosa, F. X. Trias, and A. Oliva. Conservation Properties of Unstructured Finite-Volume Mesh Schemes for the Navier-Stokes Equations. *Numerical Heat Transfer, Part B*, 65:53–79, 2014.
- L. Jofre, O. Lehmkuhl, and A. Oliva. Conservation Properties of Finite-Volume Mesh Schemes for the Simulation of Multiphase Immiscible Flow. To be submitted to *International Journal of Multiphase Flow*, 2014.
- N. Balcázar, L. Jofre, O. Lehmkuhl, J. Castro, and J. Rigola. A Finite-Volume/Level-Set Method for Simulating Two-Phase Flows on Unstructured Grids. *International Journal of Multiphase Flow*, 64:55-72, 2014.
- N. Balcázar, O. Lehmkuhl, L. Jofre, J. Castro, and J. Rigola. Numerical Investigation of Bubble Flow Using a Conservative Level-Set Method. Submitted to *Chemical Engineering Science*, 2014.

- N. Balcázar, L. Jofre, O. Lehmkuhl, and A. Oliva. A Combined Volume-of-Fluid/Level-Set Method for the Simulation of Surface-Tension-Driven Interfacial Flows. To be submitted to *International Journal of Multiphase Flow*, 2014.

D.2 Conference Proceedings

- L. Jofre, N. Balcázar, O. Lehmkuhl, R. Borrell, and J. Castro. Direct Numerical Simulation of the Flow Over a Spherical Bubble in a Turbulent Pipe Flow. In Proceedings of the *6th European Conference on Computational Fluid Dynamics*, Barcelona (Spain), July 2014.

- N. Balcázar, L. Jofre, O. Lehmkuhl, J. Castro, and A. Oliva. A Multiple Marker Level-Set Method for the Simulation of Bubbly Flows. In Proceedings of the *6th European Conference on Computational Fluid Dynamics*, Barcelona (Spain), July 2014.

- E. Schillaci, N. Balcázar, L. Jofre, O. Lehmkuhl and J. Castro. A Free Surface Model for the Numerical Simulation of Oscillating Water Column Systems. In Proceedings of the *6th European Conference on Computational Fluid Dynamics*, Barcelona (Spain), July 2014.

- L. Jofre, O. Lehmkuhl, N. Balcázar, J. Castro, J. Rigola, and A. Oliva. Conservative discretization of multiphase flow with high density ratios. In Proceedings of the *7th International Conference on Computational and Experimental Methods in Multiphase and Complex Flow*, A Coruña (Spain), July 2013.

- N. Balcázar, L. Jofre, O. Lehmkuhl, J. Castro, J. Rigola, and A. Oliva. A Finite-Volume/Level-Set Interface Capturing Method for Unstructured Grids: Simulation of Bubbles Rising Through Viscous Liquids. In Proceedings of the *7th International Conference on Computational and Experimental Methods in Multiphase and Complex Flow*, A Coruña (Spain), July 2013.

- R. Borrell, L. Jofre, O. Lehmkuhl, and J. Castro. Parallelization Strategy for the Volume-of-Fluid Method on Unstructured Meshes. In Proceedings of the *25th International Conference on Parallel Computational Fluid Dynamics*, Changsha (P.R. China), May 2013.

- L. Jofre, O. Lehmkuhl, J. Ventosa, and A. Oliva. Conservation Properties and Accuracy of Unstructured Mesh Schemes for the Navier-Stokes Equations. In Proceedings of the *7th International Symposium on Turbulence, Heat and Mass Transfer*, Palermo (Italy), September 2012.

- L. Jofre, N. Balcázar, O. Lehmkuhl, J. Castro, and A. Oliva. Numerical Study of the Incompressible Richtmyer-Meshkov Instability. Interface-Capturing Methods on General Meshes. In Proceedings of the *15th International Conference on Fluid Flow Technologies*, Budapest (Hungary), September 2012.

- N. Balcázar, L. Jofre, O. Lehmkuhl, J. Castro, and A. Oliva. Numerical Simulation of Incompressible Two-Phase Flows by Conservative Level-Set Method. In Proceedings of the *15th International Conference on Fluid Flow Technologies*, Budapest (Hungary), September 2012.

- L. Jofre, R. Borrell, O. Lehmkuhl, and A. Oliva. Parallelization of the Volume-of-Fluid Method for 3-D Unstructured Meshes. In Proceedings of the *24th International Conference on Parallel Computational Fluid Dynamics*, Atlanta (U.S. America), May 2012.

- L. Jofre, O. Lehmkuhl, J. Castro, and A. Oliva. VOF/Navier-Stokes Implementation on 3-D Unstructured Staggered Meshes. Application to the Richtmyer-Meshkov Instability. In Proceedings of the *7th International Conference on Computational Heat and Mass Transfer*, Istanbul (Turkey), July 2011.

- L. Jofre, R. Borrell, O. Lehmkuhl, J. Castro, and A. Oliva. Parallelization Study of a VOF/Navier-Stokes Model for 3-D Unstructured Staggered Meshes. In Proceedings of the *23th International Conference on Parallel Computational Fluid Dynamics*, Barcelona (Spain), May 2011.

- L. Jofre, O. Lehmkuhl, J. Castro, and A. Oliva. A PLIC-VOF Implementation on Parallel 3-D Unstructured Meshes. In Proceedings of the *5th European Conference on Computational Fluid Dynamics*, Lisbon (Portugal), June 2010.