Politecnico di Milano
Scuola di Ingegneria Industriale e dell'Informazione
Corso di Studio in Ingegneria Energetica

Level Set Methods for Prediction of Two-Phase Incompressible Flow

Author:
Pietro Salvatore Gheorghiu

Advisor:
Prof. Fabio Inzoli

Co-advisor:
Prof. Said Elghobashi

Tutors:
PhD Gaël Guédon
Michele Rosso

Academic year 2014 - 2015

"After everything we've been through, past all the fire and fury,
the one thing I know - is that we can count on each other
to get the job done. Or die trying, if that's what it takes...
...because some things are just worth fighting for."
James Raynor

Acknowledgments

I wish to express my sincere thanks to my advisor professor Fabio Inzoli and to my co-advisor professor Said Elghobashi for the opportunity they gave me to work on this project. Without their support this thesis would not have been possible.

I am grateful to my tutors Gaël and Michele for their constant guidance and encouragement, with the sharing of their expertise they considerably contributed to the final result of this work.

During my permanence in Irvine I was pleased by the presence of a lot of friends. I want to thank the guys of the department, the Italians, the people of the HOPE group and the guys of the Movement. A special thanks goes to Silvio and Cristy for the priceless friendship that we shared. I also express my gratitude to the Cordia family for the hospitality, kindness and generosity that they offered to me during the last five months in California.

Special friendships guided my life towards the completion of this journey: I take this opportunity to say thanks to all my university mates to have been travel companions in the past years. Also my friends from Muggiò are specially thanked, because of their nice disposition that they have always shown to me.

I thank my girlfriend for her unconditional presence, her caring embrace has been often more helpful than thousands of technical advices.

Finally I thank my grandmother, my uncles and aunts, my cousins, my parents, siblings and aunt Adele. Their love and support were the necessary condition that made the enterprise an enjoyable travel.

Abstract

Multi-phase flows have been deeply studied in the last two decades. Such an interest rose from the fact that the number of the possible applications in the engineering field are uncountable. Some examples are the chemical reactions, it is likely that they involve more than one phase of reactants or products at a time. In this framework the numerical simulation of those phenomena acquires an huge importance for predicting the physics of a system and better understanding the small scale evolution of the flow. The only viable way to accomplish this task is to adopt the DNS methodology for resolving the Navier-Stokes equations. Even though in the present day we do not dispose enough speed and memory from the current supercomputers to take advantage from such an approach for engineering applications, the DNS will remain the only numerical method that can provide new physical insights at all scales of turbulence in flows involving complex physical phenomena.

This thesis concerns about the study of the interface evolution between two different phases via the implementation of the Level Set Method (LSM). The Level Set method is an Eulerian method that belongs to the front capturing techniques such as the Volume of Fluid method (VOF). According to these methods the interface is reconstructed from suitable field scalar variables, in the case of the LSM from a distance function. Another category of methods for computing multifluid flows is the Front Tracking (FT). In this case a separate front marks the interface while a fixed grid is adopted for the fluid in every single phase.

We preferred to adopt the Level Set method above the others techniques since it is simple to handle, it manages easily breaks-up and merges of the interface, and it is a good candidate for the exploiting of high-order spatial discretization schemes. The main drawback of the Level Set method is the lack of mass conservation. The task of the current thesis is to compare different techniques to abolish, or at least diminish, this problem.

The work was accomplished with the essential contribution of the Mechanical and Aerospace Department (MAE) of the University of California - Irvine.

The group possesses a DNS code for the resolution of the Navier-Stokes equations written in Fortran 90, with objects and features from Fortran 2003. We took advantage from this code, which represented the starting point of the current project. A numerical model was developed using the environment provided by Octave 3.8.1, whose flexibility and simplicity is the main strength for a dynamic encoding and testing phase. The finite difference discretization was used to reduce the differential equations to algebraic equations. We adopted a structured Cartesian grid and the domain is in two dimensions. Besides the traditional method for the calculation of the interface evolution we implemented and compared three more methods.

The Level Set method theory is deepened in the first part of the thesis. Here the attention is focused on the main features of the method and the techniques to advect and reinitialize the scalar function. The theory also provides us a geometrical toolbox for the calculation of useful properties.

The numerical methods for the discretization of the differential equations were presented. Four Level Set methods were implemented: the first one is the traditional advection method of the Level Set function, the second introduces a modification in the reinitialization step to avoid the shifting of the surface, the third method embeds in the advection equation a source term in order to avoid the reinitialization step and the last one is a mass-conservative method. The verification of the proposed methods was provided. Thus, we compared the methods in term of mass conservation and considering their ability to maintain the exact shape from the analytic solution.

Finally the proposed methods are applied to a incompressible flow case test, that is a falling droplet in air. The Navier-Stokes equations were resolved with the projection method, the surface tensions were treated as source terms localized within the finite thickness of the interface.

Key words: CFD, Direct numerical simulation, Multi-phase flows, Interface capturing, Level set method.

Sintesi della tesi

Problemi che riguardano l'evoluzione di interfacce sono comuni in un elevato numero di applicazioni. Alcuni esempi sono la propagazione di superfici solide come la crescita di cristalli, composti di fluidi immiscibili e sistemi in cui si presentano più fasi di uno stesso composto.

In particolare l'ambito al quale noi siamo interessati riguarda lo studio di sistemi liquido-gas per applicazioni industriali. Descrivere come si comporta l'interfaccia tra due differenti fasi in un sistema turbolento - ad esempio una reazione chimica o una combustione per la propulsione di un velivolo - non è affatto semplice. Eppure la possibilità di poter prevedere questo tipo di fenomeni e poterne capire meglio la loro natura ha spinto la ricerca degli ultimi due decenni a sviluppare tecniche di simulazione numerica di questi fenomeni.

Per raggiungere questo scopo nell'ambito della fluidodinamica computazionale, la tecnica più adeguata è la risoluzione delle equazioni di governo tramite DNS (Direct Numerical Simulation). Si tratta di un approccio diretto che non fa uso di equazioni approssimate, né introduce coefficienti empirici in fase di risoluzione: le equazioni di Navier-Stokes sono discretizzate nello spazio e nel tempo e quindi risolte. Il pregio di questa tecnica risiede nel fatto che permette di riprodurre in maniera accurata, senza limiti teorici associati a modelli semplificatori, ogni fenomeno fisico. Un esempio è lo studio della turbolenza: un metodo DNS in potenza potrebbe risolve tutte le scale di turbolenza senza porre nessun tipo di filtro.

Il principale svantaggio di questa tecnica è legato all'eccessivo onere computazionale richiesto dagli algoritmi risolutivi per ottenere dei risultati. Anche i più moderni super computer, seppur con grandi sforzi di parallelizzazione del codice, non sono in grado di risolvere problemi complessi di utilità industriale in tempi ragionevoli.

La tesi proposta si interessa di studiare l'evoluzione dell'interfaccia in un sistema bifase. La tecnica utilizzata per descrivere l'interfaccia presente tra due fasi differenti è il Level Set Method (LSM). Questa metodologia fa parte della fa-

miglia dei metodi Euleriani, in particolare appartiene alle tecniche di *front capturing*, come ad esempio il Volume Of Fluid (VOF). L'idea chiave di questi metodi è ricostruire l'interfaccia tramite funzioni scalari definite su tutto il dominio di interesse. Nel caso del suddetto metodo adottato, la funzione in questione è la funzione distanza. L'interfaccia quindi viene identificata come una curva di livello della funzione scalare, e in generale il taglio che si effettua coincide con il livello per cui la funzione assume valore nullo.

All'atto pratico gli step da seguire per risolvere un problema numerico che sfrutta il Level Set Method sono:

1. definire la funzione distanza in un dominio $n + 1$ dimensionale, dove n è la dimensione dell'interfaccia tra liquido e fase;

2. trasportare la funzione distanza avanzando di un intervallo di tempo utilizzando una equazione differenziale alle derivate parziali (PDE);

3. re-inizializzare la funzione distanza, partendo da un intorno dell'interfaccia, tramite un processo iterativo;

4. risolvere i campi di velocità e di pressione;

5. ripartire dal punto 1 per risolvere il passo temporale successivo.

I motivi per cui è stato scelto il Level Set Method nella presente tesi riguardano: la facilità di implementazione, la semplicità con cui è possibile gestire il metodo, l'intrinseca abilità della tecnica di gestire casi anomali quali fusioni e rotture di interfacce. Lo svantaggio principale di questo metodo risiede nella sua incapacità di conservare massa, non essendo un metodo conservativo.

L'obiettivo principale di questa tesi è quello di implementare diverse tecniche che sfruttano il Level Set Method ed effettuare un confronto per valutare la loro abilità nella conservazione della massa. In particolare mettiamo a confronto quattro diversi algoritmi.

Il primo è il metodo tradizionale, utilizzato per la prima volta da Osher, Smereka e Sussman [36]. La funzione distanza viene trasportata tramite una PDE in conformità con il campo di velocità. In seguito uno step di re-inizializzazione viene applicato all'interfaccia per ricostruire la funzione distanza, almeno in un intorno del fronte.

Con il secondo metodo si vuole introdurre una modifica nell'equazione di re-inizializzazione con lo scopo di ridurre l'effetto numerico di slittamento dell'interfaccia dalla sua posizione originaria. In questo modo si cerca di andare verso la direzione di una riduzione della perdita di massa per causa dell'equazione di re-inizializzazione.

Il terzo metodo proposto aggiunge un termine di sorgente all'equazione del trasporto della funzione distanza. Questa modifica ha la pretesa di fornire un metodo che non necessita più dell'equazione di re-inizializzazione. Inoltre il termine sorgente aggiunto si annulla per valore nullo della funzione distanza: questo significa che il metodo dovrebbe mantenere inalterata l'evoluzione dell'interfaccia.

L'ultima tecnica sfrutta al posto di una funzione distanza una funzione di fase. In pratica si tratta di una funzione tangente iperbolica che viene trasportata e mantenuta con uno spessore costante grazie all'introduzione di uno step intermedio. In questo caso non è più presente l'esigenza di re-inizializzare, dato che non si ha più a che fare con una funzione distanza. Questo metodo è di tipo conservativo.

Metodi numerici

Per la riduzione delle equazioni differenziali alle derivate parziali in equazioni algebriche è stato usato il metodo delle differenze finite. La griglia adottata è di tipo ortogonale e le informazioni sono state immagazzinate entro una *staggered grid* cartesiana.

Le derivate parziali per le equazioni di Navier-Stokes sono state discretizzate nello spazio con il metodo WENO 5. Le equazioni per la risoluzione dell'interfaccia uniscono diversi metodi: nella maggior parte dei casi si adotta il metodo WENO 5, seguono metodi upwind, metodi TVD con limitatore *superbee*,il metodo di Lax-Friedrichs locale per smorzare le oscillazioni numeriche.

L'integrazione temporale per il trasporto della Level Set function è stata ottenuta con un metodo esplicito Runke-Kutta al secondo o terzo ordine. Le equazioni di Navier-Stokes sono state trasportate con un metodo di proiezione e l'avanzamento temporale è stato ottenuto con il metodo esplicito Adam-Bashford al secondo ordine.

Per la modellizzazione delle equazioni di Navier-Stokes è stato adottato il *one-fluid approach*, proposto la prima volta da Tryggvason [40]. La tensione superficiale è trattata con l'utilizzo del metodo CSF (Continuum Surface Force method).

Risultati ottenuti

I metodi implementati sono stati simulati sfruttando differenti casi test e sono stati sintetizzati nel capitolo 4. Inizialmente sono stati verificati gli algoritmi, tramite la sola risoluzione delle equazioni del trasporto della funzione Level Set. In questi casi è stato scelto un campo di velocità esterno e costante nel tempo. I test effettuati riguardano:

- rotazione completa di un disco per mezzo di un campo di velocità rotazionale;
- rotazione completa del disco di Zalesak per mezzo di un campo di velocità rotazionale;
- completamento di un intero periodo di tempo di un campo vorticoso, il quale modifica un disco fino a metà periodo per poi riportarlo nella sua posizione originale.

Tutti i test sono stati effettuati per tutti i metodi implementati e con differenti risoluzioni di griglia. I risultati ottenuti mostrano che la discretizzazione spaziale e temporale dei modelli matematici è consistente con il metodo analitico, del quale abbiamo la soluzione esatta. Dai test effettuati il quarto metodo implementato emerge per la sua capacità di conservare la massa e per il basso onere computazionale che richiede. Dal punto di vista del trasporto corretto dell'interfaccia il secondo e il terzo metodo si comportano meglio.

L'accoppiamento con le equazioni di Navier-Stokes è stato affrontato nel capitolo 5. Il calcolo delle normali a seguito della definizione dell'interfaccia è stato il problema cruciale. Difatti se questo passaggio non viene effettuato con sufficiente precisione e cautela il termine di tensione superficiale viene mal risolto causando ingenti problemi al modello complessivo. Il caso di studio considerato è una goccia che cade sotto l'effetto della sola forza di gravità. Viene applicata l'ipotesi di incomprimibilità e il flusso considerato è laminare.

Ogni metodo è stato in grado di simulare correttamente l'evoluzione della goccia che cade senza presentare problemi di stabilità complessiva del codice. Il terzo metodo ha mostrato limiti sulla stabilità che possono essere superati diminuendo la risoluzione temporale per il calcolo della soluzione.

Il lavoro è stato svolto con il contributo decisivo del gruppo di ricerca del professor Elghobashi al Mechanical and Aerospace (MAE) department di University of California - Irvine. Il lavoro è stato svolto a partire da un codice accademico di loro possesso, scritto in Fortran 90 con aggiunta di elementi e oggetti in Fortran 2003. Tutto ciò che verrà mostrato nella presente tesi è stato scritto in codice Octave ed eseguito con il corrispettivo software open-source aggiornato alla versione 8.3.1.

Parole chiave: Fluidodinamica computazionale, Direct numerical simulation, flussi multifase, Interface capturing, Level set method.

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

Problems involving moving boundaries are present in a wide range of applications. Front propagations, crystal growth and multi-phase phenomena are just few examples. The one that interests us the most is the study of turbulent liquid-gas interfaces, which is particularly important in combustion problems with liquid and gas reagents.

To correctly develop a complex engineering system, having a deep understanding of the chemical and thermodynamics processes that rule the transformation of the raw materials to the final products is important. Every single mechanical component must be designed by engineers in a way that ensures its correct operation in a safe and efficient framework. In the field that we concern, developing mechanical products such as chemical reactors and combustors requires to have a complete knowledge of the complex fluid dynamics phenomena that occur inside the channels the fluids passes through. In the past years the trend was to adopt semi-empirical techniques instead of using exact theoretical approaches (DNS, Direct Numerical Simulation) for the simulation of single-phase and multi-phase flows. The semi-empirical models rely on the resolution of approximated numerical Navier-Stokes equations, for example RANS equations (Reynolds Averaged Navier-Stokes equations) that model all the turbulence scales instead of providing their exact solution.

The reason why DNS has not been employed yet for industrial uses is due to the limits imposed by the memory and speed of current supercomputers (even with the massively parallel machines). Since the huge simulation costs of such a method, we are not expecting that the DNS will predict proficiently in reasonable times flows in complex geometries in the near future. On the other

hand DNS is the only numerical method that can really look at the physical phenomenon considering all scales of the turbulence.

This thesis contributes to the development of a DNS model for the simulation of a multi-phase system. In particular we focused our attention on the transport of the interface between the gas and liquid phase.

1.1 Multiphase simulation and recent developments

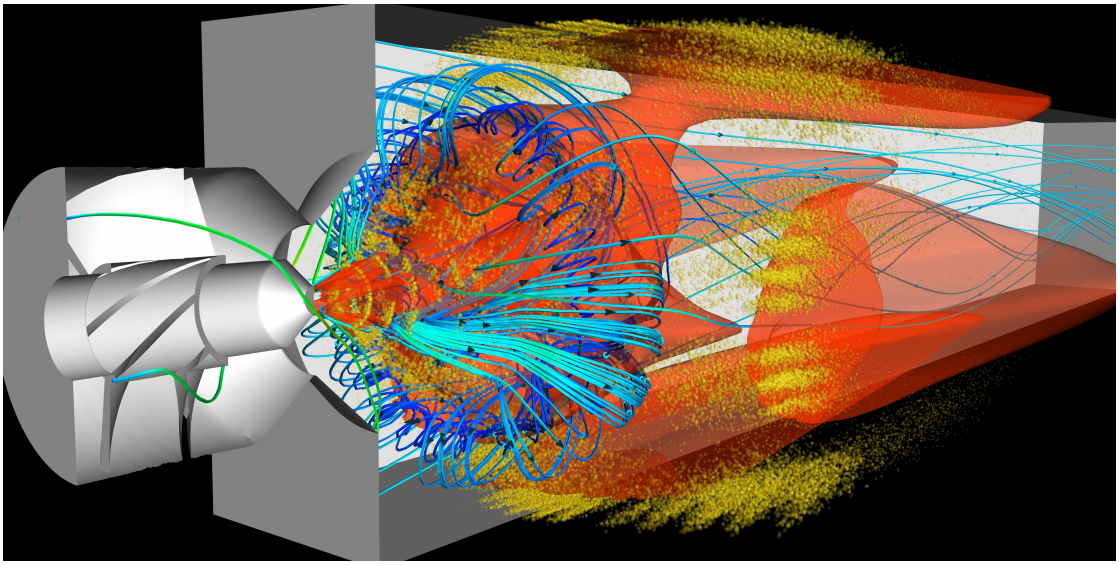


Figure 1.1: *Air flows through the swirler producing a toroidal recirculation zone. Fuel is injected at the center of the venturi. This is a possible application for a multiflow solver. The image is borrowed from NASA website.*

The numerical simulation of multi-phase flows poses three main challenges. The first one lies in the fact that one needs to numerically localize and transport the interface (also called front) separating the phases involved. The second one is due to the numerical treatment of the material properties since they discontinuously change of many order of magnitude between the phases. Finally the third one is related to the discretization of a singular forcing term, i.e. the surface tension, acting only on the front. Among the methods proposed for the numerical transport of the front over the past decades, the Volume Of Fluid method (VOF), the Front-Tracking method (FT) and the Level Set Method (LSM) have become standard numerical tools for multi-phase simulations. Besides those methods also the Constrained Interpolation Profile method (CIP) is worth to be mentioned as a technique to track fronts.

The VOF [32] describes the interface through a volume fraction scalar, thus ensuring, at least theoretically, discrete mass conservation. Nonetheless the scalar function is discontinuous across the interface and thereby a specific advection scheme is required at the cost of putting constrains on the time step size and accuracy of the simulation. Moreover, the calculation of geometric quantities such as interface normals and curvature could be challenging. In fact the surface position is obtained reconstructing the interface on every cell.

The FT, introduced by Unverdi and Tryggvason [40], makes use of an unstructured moving mesh for the discretization of the interface. The advantage of this method is that the transport of the mesh is purely Lagrangian; a problem could be that any topological change or front-to-front interaction has to be handled manually by means of mesh rearrangements, thus causing adverse effects on mass conservation. Furthermore, the numerical implementation of a Lagrangian method is not easy.

The CIP method was devised by Yabe and colleagues [38, 42] and is a scheme for solving hyperbolic equations, for color or density functions. The gradient of the color function is calculated in addition to the value of the color function. The main advantage of the CIP method is its efficiency in reducing the numerical diffusion keeping a good stability. The major drawback is the non-conservativeness. The conservative scheme CIP-CSL (Conservative Semi Lagrangian) [21] was devised with different order of accuracy in space, but do not resolve the problem of controlling the thickness of the interface.

The LSM, pioneered by the work by Osher and Sethian [27], represents the interface as the zero iso-level of a smooth function, typically a signed distance function. The latter is transported in an Eulerian fashion by solving a standard convection PDE and kept smooth through a reinitialization process. Thanks to the smoothness of the Level Set function, normals and curvature can be computed easily. The main drawback of the LSM is its inherent lack of conservation, although many strategies have been proposed to address the issue. In particular Enright et al. [9] used Lagrangian marker particles to correct the front position after the Eulerian advection step, while Sussman and Puckett [35] coupled the LSM with the VOF in order to take advantage of the good conservation properties of the latter. While these techniques alleviate the issue, they also lose the original simplicity of the LSM. A simpler approach was proposed by Olsson and Kreiss [23, 24] (also shown in this thesis), they replaced the signed distance function of the classical LSM with an hyperbolic tangent profile that is advected and re-initialialized using conservative equations. This approach was later improved by Desjardins et al. [8] and used for the simulation of turbulent atomization. An ambitious project came from an idea of Gomes and Faugeras [12], who proposed a method which is implicitly conservative in term of mass and does

not require any adjustment of the interface. Improvements and modifications are introduced by Gorokhovski et al. [31]. Recent works from Nave [22] and Lee [18] exploiting the CIP method, supported the Level Set function with the transport of its gradient, not directly resolving the problem in mass conservation but easing the calculation of geometric quantities.

The LSM simplicity, adaptivity, ease of parallelization and intrinsic ability of handling topological changes naturally are the reasons why we chose it in our study.

1.2 Objectives and contributions of the thesis

This thesis contributed to the development of a CFD code, which investigates the two-way coupling effects of finite-size deformable liquid droplets on decaying isotropic turbulence using direct numerical simulation (DNS). The model is devised for the aerospace propulsion field, where a dispersed phase of liquid fuel is sprayed inside a combustion chamber filled with air.

In the proposed work the transport of the front of a scalar function is investigated. The interface at the beginning is resolved without considering the Navier-Stokes equations, thus the implementation is addressed to resolve the physical problems that involve the evolution of the two fluids.

Within the LSM framework we investigated several techniques in order to manage the mass conservation problem that affects the transportation of the Level Set function. Every time the transport equation is resolved, the total mass is slightly modified - in particular we observed the diminishing of the mass - due to the introduction of mathematical and numerical effects that are non-physical. We compared three recent techniques based on the Level Set method to assess which one ensures the minimum loss or gain in term of mass. Every employed method was verificated, then the coupling with the Navier-Stokes equations is brought to completion.

We implemented the entire experimental code in Octave 3.8.1. This choice is made to simplify as much as possible the development and the simulation of the testing part. The verificated code is then transcribed in Fortran 90, using features and objects from Fortran 2003.

The work was made with the collaboration of the Mechanical and Aerospace department (MAE) at the University of California - Irvine, with the supervision of the professor Elghobashi. The group posses a DNS code for the resolution of the Navier-Stokes equations written in Fortran 90 with objects and features from Fortran 2003. We took advantage from this code, which represented the starting point of the current project.

The thesis is arranged as follows. The chapter 2 introduces the Level Set

technique, its analytic transport and useful geometrical and mathematical tools. The chapter 3 presents the new proposed methods and the numerical schemes adopted. Moreover in the chapter the numerical Navier-Stokes equations are derived. The chapter 4 and 5 show respectively the results of the transport of the Level Set function and the simulation of a falling droplet with the current methodology. In the chapter 6 the results are summarized and possible future works are presented.

CHAPTER 2

Level Set Method and geometrical tools

In this chapter the Level Set method and essential geometrical tools are presented. The Level Set methodology is explained in detail, including the advection equation, the reinitialization step and how to obtain a signed distance function. Then we introduce some important properties and geometrical tools that descend from the Level Set formulation.

2.1 Implicit Functions

In one-dimensional dominion suppose we divide the real line in three distinct subdomains using two points $x = -1$ and $x = 1$. Thus we define three separate subdomains as $(-\infty, -1)$, $(-1, 1)$ and $(1, +\infty)$. We call Ω^- the inner piece bounded by the two points, we call Ω^+ the outer pieces. We call $\partial\Omega$ the interface between the two regions. We are interested in defining the position of the interface. In one-dimensional dominion the interface is defined by the two points on the line, as we did locating the interface as a set of points belonging to the $\mathbb{R}^1$ domain. In that case the interface location is known by definition as $\partial\Omega = \{-1, 1\}$. This is an explicit interface representation: every point is written down in an explicit way.

Alternatively, the implicit interface representation defines the interface as the isocontour of some functions. For example, the zero isocontour of $\phi(x) = x^2 - 1$ is the set of points where $\phi(x) = 0$; i.e. it is exactly $\partial\Omega = \{-1, 1\}$. Since the isocontour corresponds exactly to the interface, in order to know its location it is necessary to calculate the curve value in $\phi = 0$. This is shown in Figure 2.1. Notice that the isocontour is defined in a zero-dimensional domain, while the ϕ

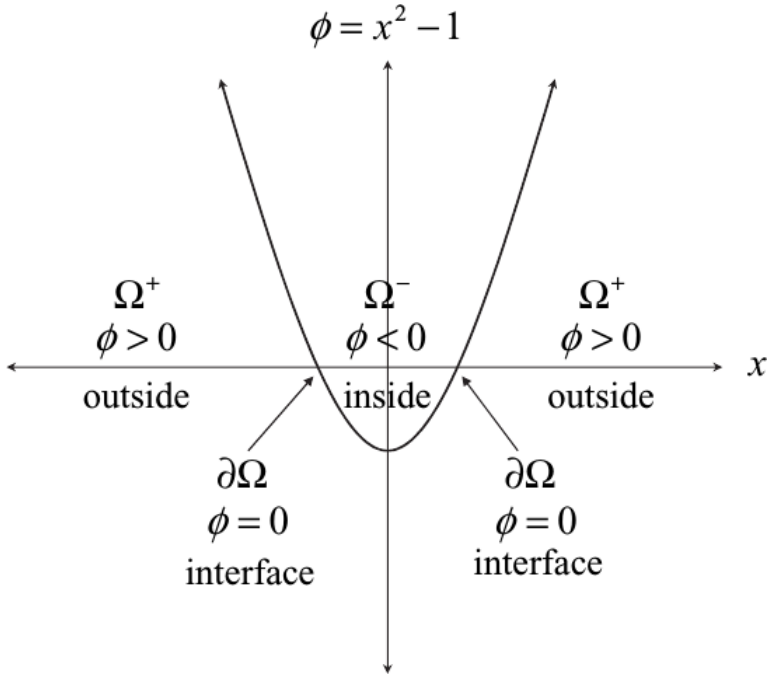


Figure 2.1: *Implicit function $\phi = x^2 - 1$ defining the subdomains Ω^- and Ω^+. The image is borrowed from [25].*

function is defined throughout the one-dimensional domain. More generally, if the implicit function is defined on all $\vec{x} \in \mathbb{R}^n$, the isocontour has always dimension $n - 1$.

Consider the two dimensional case. In two spatial dimensions the interface is a curve that separates a closed inner region from the rest of the domain. For example, let us consider $\phi(\vec{x}) = x^2 + y^2 - 1$, where the interface defined by the $\phi(\vec{x}) = 0$ isocontour is the unit circle centered in $(0, 0)$, see Figure 2.2. Now, the interior region is the unit disk $\Omega^- = \{ \vec{x} \mid |\vec{x}| < 1 \}$ and the exterior region is $\Omega^- = \{ \vec{x} \mid |\vec{x}| > 1 \}$. The interface is defined by $\partial\Omega = \{ \vec{x} \mid |\vec{x}| = 1 \}$. Thus ϕ assumes positive values in Ω^+ and negative values in Ω^-. Above, we chose the $\phi(\vec{x}) = 0$ isocontour. It could be valid any other isocontour as well, and we chose the zero level for the sake of simplicity. Throughout the proposed thesis the interface will be defined most of the time as the zero isocontour of the ϕ function.

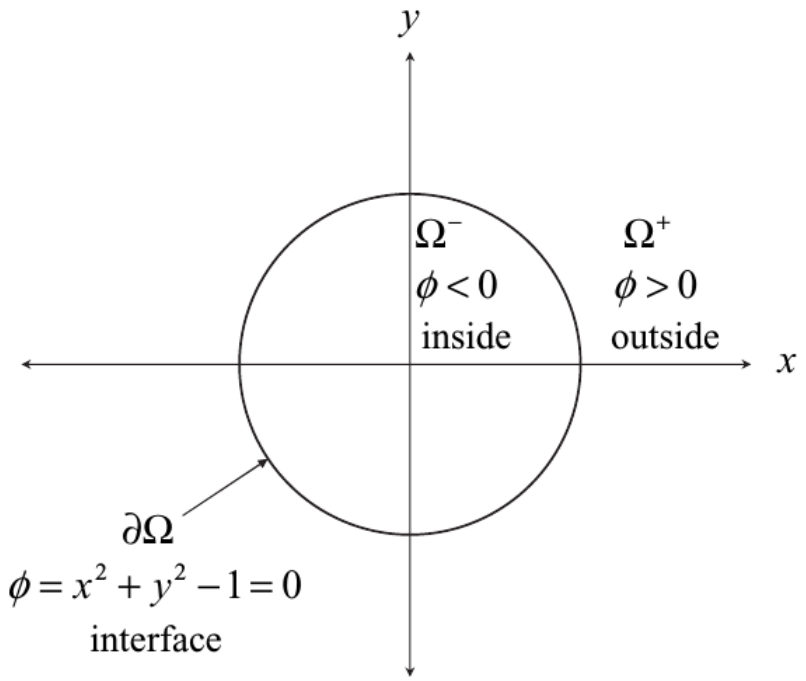


Figure 2.2: *Isocontour of the implicit function $\phi = x^2 + y^2 - 1$ defining the subdomains Ω^- and Ω^+. The image is borrowed from [25].*

2.2 Signed distance functions

A *distance function* $d(\vec{x})$ is defined as

$$d(\vec{x}) = \min(|\vec{x} - \vec{x}_I|) \quad \text{for all} \quad \vec{x}_I \in \partial\Omega, \tag{2.1}$$

implying that $d(\vec{x}) = 0$ on the boundary where $\vec{x}_I \in \partial\Omega$. Otherwise if the point does not belong to the interface, $d(\vec{x})$ is the distance between $\vec{x}$ and its closest point $\vec{x}_c$ on the interface. Furthermore the distance function benefits from the following property

$$|\nabla d| = 1, \tag{2.2}$$

which means that if we consider any point $\vec{x}$ in the dominion the distance function draws the shortest path between $\vec{x}$ and $\vec{x}_c$ to reach the interface. Any other path between these two points will be longer. To better understand what this property involves, let us introduce the signed distance function.

A *signed distance function* [25] is an implicit function ϕ with $|\phi(\vec{x})| = d(\vec{x})$ for all $\vec{x}$. Thus $\phi(\vec{x}) = d(\vec{x}) = 0$ for all $\vec{x} \in \partial\Omega$, $\phi(\vec{x}) = -d(\vec{x})$ for all $\vec{x} \in \Omega^-$ and $\phi(\vec{x}) = d(\vec{x})$ for all $\vec{x} \in \Omega^+$. That is, it determines the distance of a given point from the interface, with positive sign when $\vec{x}$ is inside the isocontour and with

negative sign when $\vec{x}$ is outside the isocontour. The property

$$|\nabla\phi| = 1 \tag{2.3}$$

descends from equation (2.2). Furthermore, given a point $\vec{x}$, since $\phi(\vec{x})$ is the signed distance to the closest point to the interface, we can write

$$\vec{x}_C = \vec{x} - \phi(\vec{x})\vec{n} \tag{2.4}$$

where $\vec{n}$ is the unit normal vector with respect to the interface in $\vec{x}$.

In the last section we used $\phi(x) = x^2 - 1$ as the implicit function for the definition of $\partial\Omega = \{ -1, 1 \}$. Now we chose $\phi(x) = |x| - 1$ whose isocontour is the same of the previous case, but obtained from a signed distance function, as shown in Figure 2.3. The new representation keeps the interface the same, although the implicit curve is different. The new curve satisfies the property (2.3) for every point but $x = 0$. At $x = 0$ there is a kink that could be problematic since the derivative assumes infinite value and thus it is not defined. In order to determine several properties - such as the normal and the curvature - we need to know the derivative value. However we can easily avoid this problem considering that in a Cartesian grid we cannot sample any single real point, thus the kink will be slightly smeared out, making sure that the derivative assumes a finite value. Furthermore there were devised several method to avoid the problem, i.e. the narrow band approach [1]. The main task of this method was to lighten the computational efforts. To achieve that, only a bunch of points of the domain grid in a narrow band built around the interface are resolved. If we exploit the fact that only the region near the zero isocontour is interesting to be resolved, we do not even have the issue to deal with the kink point. Avoiding that point prevent us from numerical problems.

In two spatial dimensions we replaced the implicit function $\phi(\vec{x}) = x^2 + y^2 - 1$ with the signed distance function $\phi(\vec{x}) = \sqrt{x^2 + y^2} - 1$ whose zero isocontour is a unit circle centered in $(0, 0)$. Again, there is a singular point in $\vec{x} = (0, 0)$ and all the considerations made for the 1-D case are still valid.

2.3 Level Set formulation

Above we defined implicit surfaces, now we introduce the Level Set method. *Level Set methods* add dynamics to implicit surfaces and allows to track the interface position in an implicit fashion.

The *Level Set method* was devised by Osher and Sethian [27] in 1988 as a method to implicitly evolve the interface advecting a scalar function (implicit surface) with a partial differential equation (PDE). The key idea is to set the the

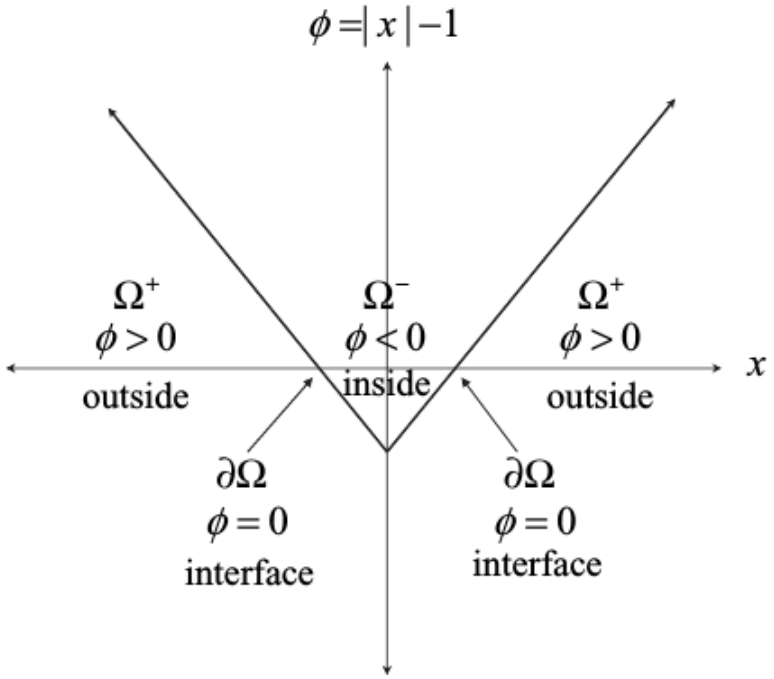


Figure 2.3: *Signed distance function* $\phi(x) = |x| - 1$ *defining the subdomains* Ω^- *and* Ω^+*. The image is borrowed from [25].*

interface as the zero level set of the implicit function. The interface will propagate naturally as the zero level set evolves with the level set function. Some of the advantages of the Level Set method over other techniques are its ability to easily handle topological changes such as merges and breaks, its simplicity and the ease of implementation.

Let us consider a closed interface $\Gamma(t) \in \mathbb{R}^n$ with codimension 1 and $t \in \mathbb{R}^+$. Let it be $\Omega^-(t)$ the region that $\Gamma(t)$ encloses, $\Omega^+(t)$ the region outside $\Gamma(t)$. We define over $\Omega(t)$ the function $\phi(\vec{x}, t)$, the Level Set function, that satisfies the following conditions

$$\begin{cases} \phi(\vec{x}, t) > 0 & \text{in } \Omega^+(t) \\ \phi(\vec{x}, t) = 0 & \text{on } \Gamma(t) \\ \phi(\vec{x}, t) < 0 & \text{in } \Omega^-(t). \end{cases} \tag{2.5}$$

The interior and exterior domain contain opposite signed values of the ϕ function. The exterior partition outside the interface has values $\phi > 0$, the interior partition inside the interface has values $\phi < 0$. The phase location of a generic point $\vec{x}$ is given by the sign of the ϕ function. The above sign convention of the ϕ function will be adopted throughout the text. The reader can refer to 2.3 to better understand how the partition is taken.

Supposed that a velocity field is defined all over the domain. The velocity in each point of the implicit surface is given as $\vec{v}(\vec{x})$. Suppose now $x(t)$ is a particle trajectory on the interface $\Gamma(t)$ moving with velocity $\vec{v} = \dot{x}(t)$. By definition, $\phi(x(t), t) = 0$. Differentiating with respect to t, we get

$$\phi_t + \vec{u} \cdot \nabla\phi = 0 \tag{2.6}$$

where the t subscript denotes the partial derivative of ϕ in the time variable t. ∇ is the gradient operator, so that

$$\vec{u} \cdot \nabla\phi = u\phi_x + v\phi_y + w\phi_z$$

In [27] is shown that the interface is accurately moved by equation (2.6) accordingly with the velocity field $\vec{v}(x)$. For better understanding the equation (2.6) [37] let Γ parameterized by $x(s,t)$ and $y(s,t)$, then the evolution of the curve is determined by the equations:

$$\frac{dx(s,t)}{dt} = u(x(s,t), y(s,t))$$
$$\frac{dy(s,t)}{dt} = v(x(s,t), y(s,t)).$$

We must have

$$\frac{d\phi(x(s,t), y(s,t), t)}{dt} = \frac{d\phi}{dx}\frac{dx}{dy} + \frac{d\phi}{dy}\frac{dy}{dt} + \frac{d\phi}{dt} = \phi_t + u\phi_x + v\phi_y = 0 \tag{2.7}$$

since $\phi(x(s,t), y(s,t), t)$ is defined to be zero for all (s,t). It is now clear that each point at the interface behaves as a material point, indeed the point $\vec{x_0}(t)$ that belongs to the interface is affected by the stream generated by the the velocity field. The point is transported for pure convection.

It is important to notice that the Level Set function is moved precisely *only* at the interface. In the neighborhood the Level Set function is approximately resolved accordingly the velocity field and in the rest of the dominion the solution has no real accordance with the velocity field. This fact in practice means that the Level Set function does not conserve its original slope during the evolution [3]. As it will be explained later, we want the Level Set function to be a signed distance function everywhere, or at least in a wide enough region around the interface, and for every time step. Whereas the zero level set is well resolved and keeps the exact position, the transport equation makes the rest of the function to shift from being a signed distance function. Moreover problems in computation may occur (the solver may not converge) if the function gets a bad trend. The consequence is that not even for divergence free velocity field the advection of ϕ is made in a conservative way, but the method is intrinsically non-conservative.

Actually, we only need the normal component of $\vec{v}$. The gradient of the implicit function in 3-D is defined as

$$\nabla\phi = \Big(\frac{\partial\phi}{\partial x}, \frac{\partial\phi}{\partial y}, \frac{\partial\phi}{\partial z}\Big). \tag{2.8}$$

The gradient is perpendicular to the isocontours of ϕ and points in the direction of increasing ϕ. Therefore, if $\vec{x_0}$ is a point on the zero isocontour of ϕ, then $\nabla\phi$ evaluated at $\vec{x_0}$ is a vector that points outward the interface. Thus, the unit outward normal is

$$\hat{n} = \frac{\nabla\phi}{|\nabla\phi|}, \tag{2.9}$$

for every point at the interface. Thus, the equation (2.6) becomes

$$\phi_t + u_n|\nabla\phi| = 0. \tag{2.10}$$

2.4 Reinitialization equation

It is common acknowledgment that the ϕ function should stay closed to a signed distance function during the calculation process. Thus maintaining the interface close to $|\nabla\phi| = 1$ ensures that the width of the interface is smeared with a constant thickness. However in general solving the equation (2.2) does not keep the Level Set function a signed distance function. The mathematical proof is shown in [3]. In the Figure 2.4 is represented a one-dimension transport problem where the ϕ line is advected. The interface (in this case it is represented by the point on the x-axis) is well transported, but the rest of the function does not keep the initial slope.

The problem can be avoided adding a new reinitialization equation, which is the hyperbolic *Hamilton-Jacobi* equation

$$d_\tau - \text{sgn}(d)(1 - |\nabla\phi|) = 0 \tag{2.11}$$

with the initial condition

$$\phi(\vec{x}, 0) = d(\vec{x}, t),$$

where sgn is the sign function and is defined as

$$\text{sgn}(d) = \frac{d}{|d|} \tag{2.12}$$

and provides the sign of the function. The integration of this equation over a fictitious time τ provides a solution that corresponds to the reconstruction of the ϕ function starting from $\{\phi = 0\}$. Every time step moves ahead the calculation in

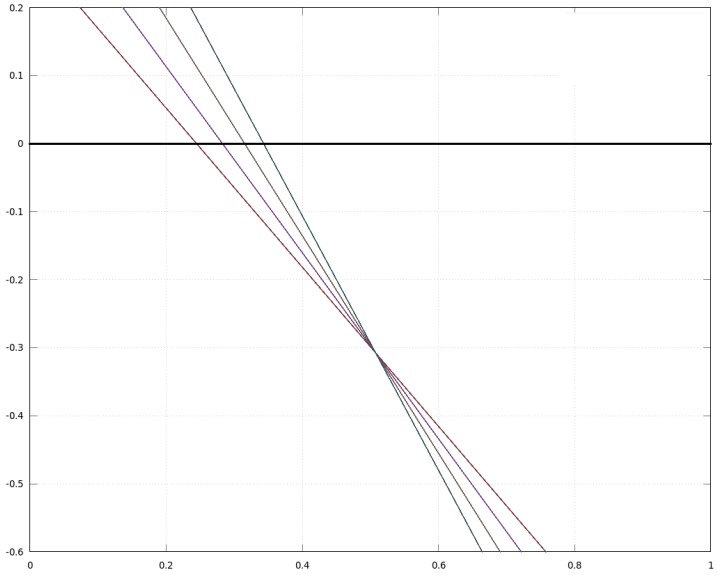


Figure 2.4: *The point interface is transported without the use of the reinitialization equation. While the position of the zero level is exact, the surface changes its slope.*

accordance with the direction of the characteristic. The transporting velocity is represented by the signed function. Usually it is not necessary reinitializing all over the domain, instead we do that just for a limited numbers of cells around the interface. Thus we can save computational time and reduce spurious oscillations. When the reinitialization step comes to convergence we obtain a signed distance function whose zero level is exactly the same as the initial function.

That is not true when the computation is made numerically. Generally to solve the reported above PDE equation upwind methods are used, where the discrete derivatives are computed in accordance to the direction of the characteristic. However this property is violated if the integration is made across the interface, as Russo and Smereka stressed in [30]. In Figure 2.5 is represented the effect of the reconstruction of a signed distance function over a 1-D domain using the equation (2.11). The line that we obtained after the reinitialization is susceptible to a slight shift since the method is inconsistent across the interface. The more the interpolated line is distant from a nodal point, the bigger is the position variation. Thus the final numerical reconstruction provides a function with the zero level set that does not match with the initial one. Such a problem usually causes a shrinking of the interface that affects the area conservation.

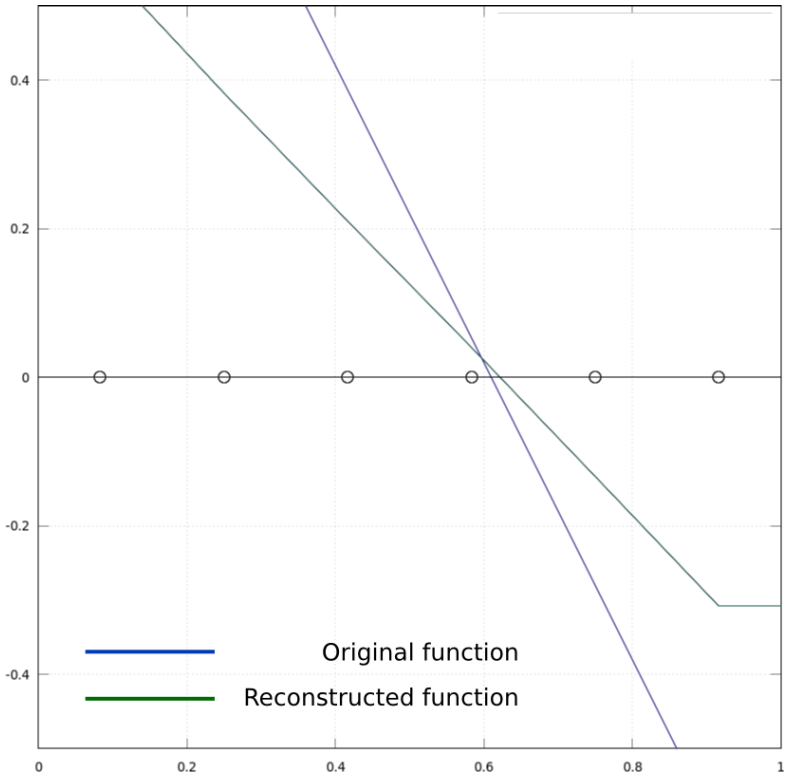


Figure 2.5: *Shifting of the interface because the reinitialization. Even though the line is not transported, it slightly changes its position.*

2.5 Geometric tools

The Level Set formulation provides us several geometric quantities (well exposed in [26]) that can be easily represented in terms of ϕ. In the section 2.3 we defined the unit normal vector to the interface (2.9); now we can introduce the mean curvature of the interface as

$$\kappa = \nabla \cdot \frac{\nabla\phi}{|\nabla\phi|}. \tag{2.13}$$

We adopted a convention that makes κ positive for convex regions, negative for concave regions. For example, let us consider a bubble immersed in water. At the initial point, supposing the bubble a perfect sphere, the curvature will be positive everywhere at the interface. Advancing in time the bubble begins to rise up due to the gravity force and changes its shape since the surface tension acts over it. Once the bubble reaches the steady state (the resultant acting force over the bubble is zero) and it reaches its terminal velocity the final shape will be similar to a "mushroom", flattened on the bottom and elongated on the cross-rise direction. The mean curvature κ on the bottom will become negative since the formation of a concave region. The property κ is necessary to calculate the surface tension.

The *Heaviside function H* is defined as

$$H(\phi) = \begin{cases} 0 & \text{if} \phi \quad \leq 0, \\ 1 & \text{if} \phi \quad > 0. \end{cases} \tag{2.14}$$

Notice that this is a one-dimensional function. It is also clear that ϕ depends on $\vec{x}$; nevertheless is not important to specify this dependence when working with H. We chose arbitrarily to include the boundary with the interior region.

We can use (2.14) to calculate the area or volume integral (it depends whether the integral is in $\mathbb{R}^3$ or $\mathbb{R}^2$) as

$$\int_\Omega f(\vec{x})(1 - H(\phi(\vec{x}))) \, d\vec{x} \tag{2.15}$$

representing the integral of f over the interior region Ω^-. In the same way,

$$\int_\Omega f(\vec{x}) H(\phi(\vec{x})) \, d\vec{x} \tag{2.16}$$

is the integral of f over the exterior region Ω^+.

By definition, the directional derivative of the Heaviside function H in the normal direction $\hat{n}$ is the *Dirac delta function*

$$\hat{\delta}(\vec{x}) = \nabla H(\phi(\vec{x})) \, d\vec{x} \cdot \hat{n}. \tag{2.17}$$

The δ distribution is non-zero only at the interface. Rewriting the above equation as

$$\hat{\delta}(\vec{x}) = H'(\phi(\vec{x}))\nabla\phi(\vec{x}) \cdot \frac{\nabla\phi(\vec{x})}{|\nabla\phi(\vec{x})|} = H'(\phi(\vec{x}))|\nabla\phi(\vec{x})| = \delta(\phi(\vec{x}))|\nabla\phi(\vec{x})|$$

where

$$\delta(\phi) = H'(\phi) \tag{2.18}$$

we obtain the 1-D version of the Dirac delta function which evaluates to 1 where $\phi = 0$ and 0 everywhere else.

We can use (2.17) to calculate the surface or line integral of a function f over the boundary $\partial\Omega$ (it depends whether the integral is over $\mathbb{R}^3$ or $\mathbb{R}^2$)

$$\int_{\Omega} f(\vec{x})\hat{\delta}(\vec{x})\, d\vec{x} \tag{2.19}$$

where the region of integration is all of Ω, since $\hat{\delta}$ cuts out everything except the boundary $\partial\Omega$.

Special care must be taken when resolving discontinuities in the Navier-Stokes equations. Consider the surface integral in equation (2.19). Since $\delta(\phi) = 0$ almost everywhere it is unlikely that a numerical approximation based on sampling will give a good approximation to the integral. Otherwise is likely that over the grid the function will be irregularly caught. Moreover in the Navier-Stokes equations the δ function is required to calculate properties such as surface tension. Fluid dynamics applications of the Level Set method raise problems about steep gradient at the interface: indeed we deal with a density ratio of about 1000 considering water and air. A such high discontinuity yields a non stable solution of the Navier-Stokes equations.

Thus, we decided to smear out the H function and the δ function. First, we define the smeared-out *Heaviside function* as

$$H_\varepsilon(\phi) = \begin{cases} 0 & \phi < -\varepsilon, \\ \frac{1}{2} + \frac{\phi}{2\varepsilon} + \frac{1}{2\pi}\sin\left(\frac{\pi\phi}{\varepsilon}\right) & -\varepsilon \le \phi \le \varepsilon, \\ 1 & \phi > \varepsilon, \end{cases} \tag{2.20}$$

where ε is a parameter that determines the size of the bandwidth of numerical smearing. We adopt $\varepsilon = 1.5\Delta x$ where Δx is the distance between two adjacent points of the grid (see chapter 3). That makes the width of the interface equal to three grid cells when ϕ is normalized as a signed distance function. Then the delta function is defined according to equation (2.18) as the derivate of the

Heaviside function

$$\delta_\varepsilon(\phi) = \begin{cases} 0 & \phi < -\varepsilon, \\ \frac{1}{2\varepsilon} + \frac{1}{2\varepsilon}\cos\left(\frac{\pi\phi}{\varepsilon}\right) & -\varepsilon \leq \phi \leq \varepsilon, \\ 0 & \phi > \varepsilon. \end{cases} \tag{2.21}$$

The smeared-out functions approach to the calculus of implicit functions leads to first-order accurate methods. Thus the error in the calculation is always of order $O(\Delta x)$, regardless the integration method used. Thus, no matter how high is the order method to calculate the equation of the surface (2.19), the error will always be first-order accurate.

2.6 Error estimation

In the previous chapter we have shown how to calculate volume and surface of the region enclosed by the zero level of a Level Set function. The first task of this thesis is to assess the capability of several Level Set approaches to preserve the mass. Indeed, as explained in the chapter 1, we must deal with the fact that one of the most crucial drawbacks of the Level Set method is its incapability to preserve the mass. The method is not conservative since the advection of ϕ shifts the Level Set function from the distance function and the numerical reconstruction is not exact (see section 2.3). Therefore it is crucial to understand how well each method is able to keep the mass unchanged. Furthermore, the numerical implementation of each method introduces numerical diffusion (for more information about numerical diffusion, see [41]). Numerical diffusion makes the solution of the discretized transport equation to be different from the exact solution. For example, the upwind discretization method is a first order method and the solution of the modified PDE generates an additional term called diffusive term. The new term is proportional to the grid size: the error grows with spatial discretization.

Thus, to assess the behaviour of the Level Set methods proposed in this text, the following measures of error will be used:

$$e_m\% = \frac{M - M_0}{M_0} \cdot 100 \tag{2.22}$$

where

$$M = \int_\Omega \rho(1 - H(\phi))\, dxdy \tag{2.23}$$

that is exactly the equation (2.15) valid for a two-dimensional case. The equation (2.22) represents the percentage mass loss (or gain) with respect to the initial

value M_0 of the mass. We chose to keep track of the entire region of the domain for the mass calculation. Let us consider a rectangular dominion of length l_x and width l_y. We define $\Delta x = \frac{l_x}{N_x}$ and $\Delta y = \frac{l_x}{N_x}$ as the length and the width of the cell, where N_x and N_y are number of the grid points in the x and y direction respectively. Over the discretized domain the value of the ϕ function is stored at the center point of each grid cell. Thus we can calculate the mass as a discrete summation of the cell values throughout the domain:

$$M = \sum_{i=0}^{l_x} \sum_{j=0}^{l_y} \rho_{ij}(1 - H(\phi_{i,j}))\, \Delta x \Delta y. \tag{2.24}$$

We calculate the error length as the percentage variation of length with respect to the initial length

$$e_l\% = \frac{L - L_0}{L_0} \tag{2.25}$$

where

$$L = \int_\Omega \delta(\phi)\, dxdy \tag{2.26}$$

and L_0 is the initial length of the interface. This equation is valid in two-dimensional cases. The discrete form of the equation (2.26) is

$$L = \sum_{i=0}^{l_x} \sum_{j=0}^{l_y} \delta(\phi_{i,j})\, \Delta x \Delta y. \tag{2.27}$$

We introduce the L_{inf} error, defined as

$$L_{inf} = \max(|\phi_{i,j} - d_{i,j}|), \tag{2.28}$$

and the L_1 error, defined as

$$L_1 = \sum_{i=0}^{N_x} \sum_{j=0}^{N_y} |\phi_{i,j} - d_{i,j}| / N_x N_y. \tag{2.29}$$

Where d is the exact Level Set function equation provided by the analytic solution, N_x and N_y are the number of the intervals on the domain with respect to the x and y coordinates. The value of a point of a signed distance function is the spatial coordinate position of that point. Hence, these errors provide informations about the position of the computed interface with respect to the exact interface and will be referred to as the "distance errors".

Those measures of error are useless if the Level set is not reasonably resolved by the mesh. Thus we chose cases where the feature sizes of the Level Set are not smaller than the grid resolution.

CHAPTER 3

Numerical methods

The equations and the resolution algorithm are implemented using the software Octave for the verification step, then the results are transferred in Fortran 90 as part of the main code for the resolution of the 3-D turbulent problem and the validation of the code. Every partial differential equation is reduced to an algebraic equation system, where the derivatives are treated as finite differences. In this chapter the proposed new models to find the solution of the Level Set equation are presented in detail. furthermore we explain the discretization procedure adopted for their numerical resolution. Thus, we introduce the mass conservation law, the governing conservation law and the detailed sequence of the algorithm to solve the velocity and pressure field.

3.1 Grid discretization

The domain is discretized using a regular orthogonal grid. In this case the position of the grid nodes is uniquely defined by a set of indices. In particular a Cartesian grid is used, this means that all the cells within the domain are squares or cubes of the same dimension. We always consider a 2-D domain, the 3-D complication is not necessary since we do not simulate problems that involve turbulence. After the validation step the code will be implemented over a 3-D domain. Anyway, the extension to the 3-D case is trivial thanks to the ease in handling the level set approach, thus for the sake of simplicity in this thesis is presented only the 2-D problem implementation.

Let us consider a rectangular domain whose length is l_x and the height is l_y. We embed the domain in a Cartesian coordinate system where the length of the

rectangle lays on the x coordinate, the height of the rectangle on the y coordinate, and the south-west corner of the rectangle coincides with the point $(0, 0)$. Now we can divide the domain in a finite number of grid points, N_x for the x-axis and N_y for the y-axis. Consider that the interval between two different grid points is fixed and it is the same in the two directions. Thus the domain is split in squares whose corners are the grid points (from now we call them nodal points). For the sake of simplicity it is common use to adopt the indices convention instead using the Cartesian coordinates convention. Therefore, the convention we use to define the nodal points is the following:

$$u_{i,j}^{n} = u(x + i\Delta x, y + j\Delta y, t_0 + n\Delta t). \tag{3.1}$$

Usually the subscript identifies the spatial position and the superscript the position in time. Thus, the Cartesian grid is arranged by the following set of elements:

$$\{\, (x_i, y_j) \mid 0 < i < N_x\,,\, 0 < j < N_y \,\}\,;$$

therefore every single point over the grid is univocally defined by a two elements vector. The set of points is ordered, $x_0 < \cdots < x_{i-1} < x_i < \cdots < x_{N_x}$ and $y_0 < \cdots < y_{j-1} < y_j < \cdots < y_{N_y}$. The distance between a point and the next one is fixed, e.g. $\frac{l_x}{N_x} = x_{i+1,j} - x_{i,j} = \Delta x$ and $\frac{l_y}{N_y} = x_{i,j+i} - x_{i,j} = \Delta y$. Notice that using a Cartesian grid implies the use of a rectangular domain.

After the definition of the mesh the finite difference method demands to discretize the values of the scalar and vector properties over the grid. In particular the vector properties are the velocity fields, while the scalar properties are the ϕ function, pressure, etc. It is not convenient to store the velocity in the same cells of the scalar properties, mainly because the solution of the pressure field in the Navier-Stokes equations could oscillate. For a detailed description of the problem see [41]. To avoid this problem a staggered grid was implemented. The traditional grid allocates all the variables in the nodal points. In a staggered grid (Figure 3.1), only the scalar values are stored in the nodal points, each other variable is stored in its own sub-grid shifted half a cell distance from the nodal points. In a 2-D domain there are three different sub-grids, one for the pressure, one for the horizontal component u of the velocity and the third one for the vertical component v of the velocity. Thus, in the staggered grid arrangement the velocities are stored in the cell faces of the control volume, while the scalar properties are stored in the nodal points. The notation that we use is the following: the scalar properties are stored in the $x_{i,j}$ points, the u-component of the velocity is stored in $x_{i+\frac{1}{2},j}$ and the v-component of the velocity is stored in $x_{i,j+\frac{1}{2}}$. The grid is not staggered in time. We call the $x_{i+\frac{1}{2},j}$ and $x_{i,j+\frac{1}{2}}$ positions cell faces.

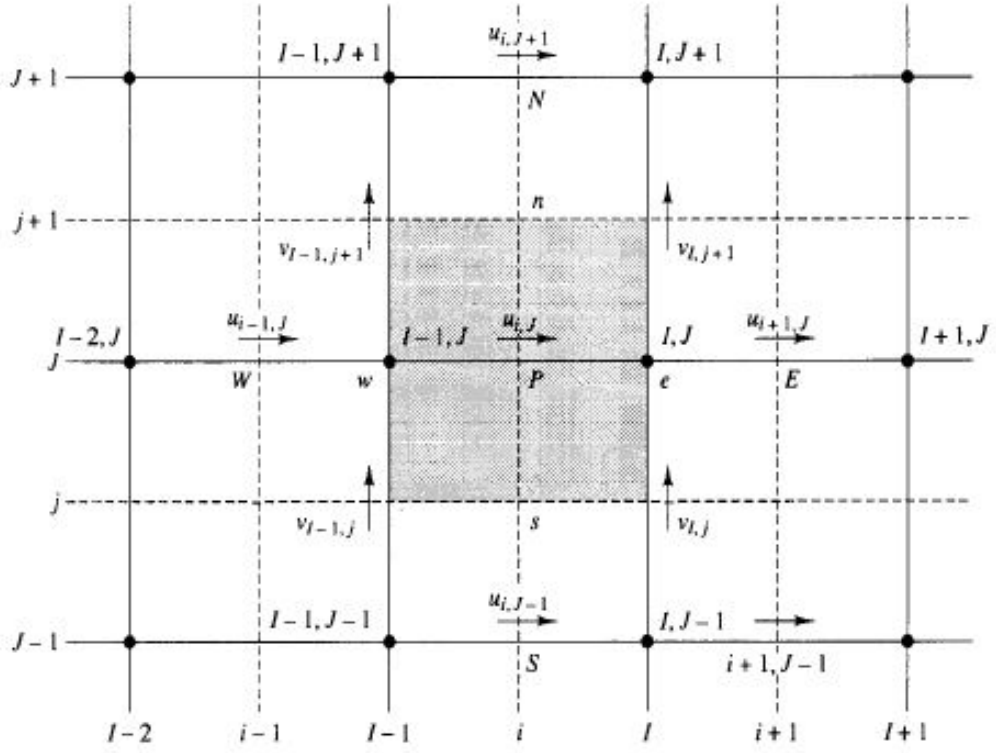


Figure 3.1: *Staggered grid representation. The scalar properties are stored in a different position from the other properties. The image is borrowed from [41].*

Around the domain a band of ghost cells is added, as it is represented in Figure 3.2. They are not part of the real domain, but they are necessary to ease the calculation at boundaries and for parallel communications. Applying a stencil for the calculation of the derivatives requires the presence of the ghost cells to gain all the values for the calculations of the differences. Moreover the ghost cells help in the definition of the boundary conditions.

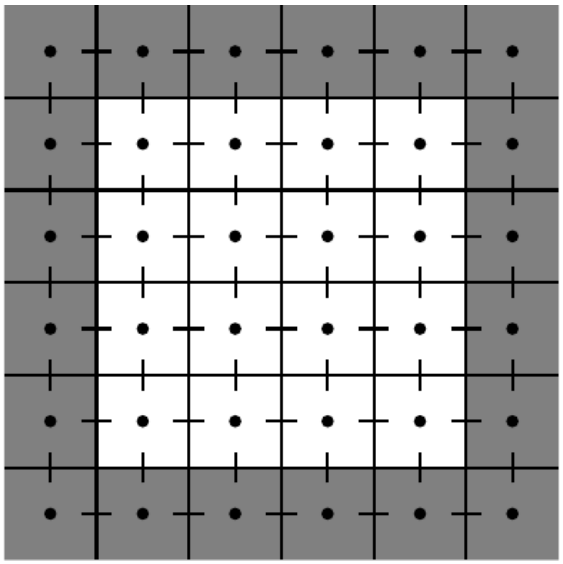


Figure 3.2: *The gray band is the ghost cells strip around the real boundary of the domain.*

3.2 Finite difference discretization

The finite difference method is used to convert the analytic equations and the governing equations into a set of algebraic equations. It consists in approximating the differential operator by replacing the derivatives in the equation using differential quotients. The difference between the exact solution and the numerical solution provides the error that is committed going from a differential equation to an algebraic equation. It is usually called truncation error because it has the magnitude of the truncation term of the Taylor expansion.

In this section we listed the schemes that we used to reduce the differential operators to algebraic terms.

3.2.1 Forward, backward and centered difference

Let us consider a mono-dimensional case. From the definition of the derivative we write the forward difference of u with respect to the spatial coordinate as

$$\frac{\partial u}{\partial x} = \lim_{\Delta x \to 0} \frac{u(x+h) - u(x)}{\Delta x},$$

removing the limit we obtain the forward difference

$$\frac{\partial u}{\partial x} \approx \frac{u(x+\Delta x) - u(x)}{\Delta x} = \frac{u_{i+1} - u_i}{\Delta x}. \tag{3.2}$$

In the same way we can write the backward difference of u as

$$\frac{\partial u}{\partial x} \approx \frac{u(x) - u(x-\Delta x)}{\Delta x} = \frac{u_i - u_{i-1}}{\Delta x}. \tag{3.3}$$

Finally, the central difference formula is the following:

$$\frac{\partial u}{\partial x} \approx \frac{u(x+\Delta x) - u(x-\Delta x)}{2\Delta x} = \frac{u_{i+1} - u_{i-1}}{2\Delta x}. \tag{3.4}$$

To obtain the second order central difference let us consider the Taylor expansion of the u_{i+1} and u_{i-1} around u_i:

$$u_{i+1} = u_i + \Delta x \frac{\partial u}{\partial x}\bigg|_i + \frac{\Delta x^2}{2}\frac{\partial^2 u}{\partial x^2}\bigg|_i + \frac{\Delta x^3}{3}\frac{\partial^3 u}{\partial x^3}\bigg|_i + O(\Delta x^4)$$
$$u_{i-1} = u_i - \Delta x \frac{\partial u}{\partial x}\bigg|_i + \frac{\Delta x^2}{2}\frac{\partial^2 u}{\partial x^2}\bigg|_i - \frac{\Delta x^3}{3}\frac{\partial^3 u}{\partial x^3}\bigg|_i + O(\Delta x^4).$$

Adding these two expansions the odd terms cancel each other out. Moreover we remove the terms of $O(\Delta x^4)$ and higher since they are negligible. From the sum we obtain

$$u_{i+1} + u_{i-1} = 2u_i + \Delta x^2 \frac{\partial^2 u}{\partial x^2}\bigg|_i$$

and if we rearrange to solve for $\frac{\partial^2 u}{\partial x^2}\Big|_i$ the final result is:

$$\frac{\partial^2 u}{\partial x^2} = \frac{u_{i+1} - 2u_i + u_{i-1}}{\Delta x^2}. \tag{3.5}$$

The extension to the two-dimensional case is straightforward. I just show the result of the mixed term of the second order difference scheme:

$$\frac{\partial^2 u}{\partial x \partial y} = \frac{u_{i+1,j+1} - u_{i+1,j-1} - u_{i-1,j+1} + u_{i-1,j-1}}{4\Delta x \Delta y}. \tag{3.6}$$

Notice that usually the properties values are stored in the cell nodes. The derivatives in the finite difference discretization framework must be stored in the face nodes.

3.2.2 Upwind method

The upwind scheme is an adaptive finite difference stencil that can simulate the direction of the propagation of the information in a flow field. The upwind scheme takes into account the direction of the flow determining the sign of the the convection field. Then it applies the biased difference according to the flow direction; the values at the cell faces are always taken to be equal to the value at the upstream node. The Figure 3.3 represents the upwind principle. If the

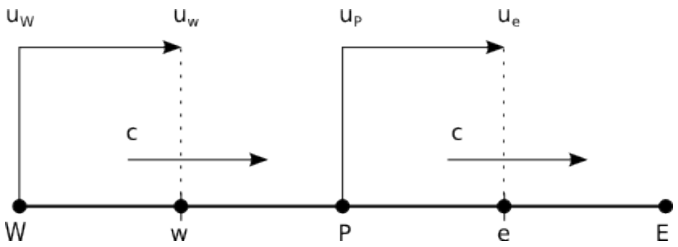


Figure 3.3: *Upwind scheme representation. The u property is transported in accordance with the velocity direction.*

convective flux courses towards the east side ($c > 0$) the the value of u_e becomes equal to u_P, otherwise if the convective flux courses towards the west side ($c < 0$) the value of u_e becomes equal to u_E. Considering the following one-dimensional linear advection equation

$$\frac{\partial u}{\partial t} + c\frac{\partial u}{\partial x} = 0$$

the first order upwind scheme is given by

$$\begin{aligned} \frac{u_i^{n+1} - u_i^n}{\Delta t} + c\frac{u_i^n - u_{i-1}^n}{\Delta x} = 0 \quad \text{for} \quad c > 0, \\ \frac{u_i^{n+1} - u_i^n}{\Delta t} + c\frac{u_{i+1}^n - u_i^n}{\Delta x} = 0 \quad \text{for} \quad c < 0. \end{aligned}$$

In this thesis the upwind scheme is usually written using the Godunov's scheme [2] that assumes the following form (for the two dimensional case):

$$u_{i,j}^{n+1} = u_{i,j}^n - \Delta t G(u)_{i,j},$$

and

$$G(u)_{i,j} = \begin{cases} \sqrt{\max(a_+^2, b_-^2) + \max(c_+^2, d_-^2)} & \text{if} \quad c_{i,j} > 0 \\ \sqrt{\max(a_-^2, b_+^2) + \max(c_-^2, d_+^2)} & \text{if} \quad c_{i,j} < 0, \end{cases} \tag{3.7}$$

where for any real number d, it is $d_+ = \max(d, 0)$, $d_- = \min(d, 0)$, with

$$a = \frac{u_{i,j} - u_{i-1,j}}{\Delta x},$$
$$b = \frac{u_{i+1,j} - u_{i,j}}{\Delta x},$$
$$c = \frac{u_{i,j} - u_{i,j-1}}{\Delta y},$$
$$d = \frac{u_{i,j+1} - u_{i,j}}{\Delta y}.$$

3.2.3 Total Variation Diminishing methods: TVD

The main drawback of high-order schemes is that they produce artificial oscillations near discontinuities. That is true in particular when the Péclet number is high. The TVD schemes are designed to avoid this problem introducing an artificial diffusion or a weighting that counteracts the formation of oscillations. TVD methods exploit the idea that the total variation of a function must never grow. As soon as this condition is satisfied we obtain a stable and non-oscillatory method. Let us define the total variation as

$$TV = \sum_i |u_{i+1} - u_i|$$

and a method is TVD if the total variation does not increase in time. That means that the property u_i monotonically increases or decreases in space at every time step. A method that satisfies this condition is said to be *monotonically preserving*, respecting the following disequation:

$$TV(u^{n+1}) \leq TV(u^n)$$

Basically the advection of u_i is made using an upwind scheme with the addition of a limiter that smears the function out. There are several limiters in literature [19], we chose to use the *Superbee* limiter mainly because is known to be one of the least diffusive. The general advecting function for u_i, considering the cell i to be upstream, is

$$u_{i+\frac{1}{2}} = u_i + \frac{\Delta x}{2}\psi_i(r)$$

where $\psi(r)$ is the limiter function

$$\psi(r) = \max[0, \min(2r, 1), \min(r, 2)].$$

Here

$$r = \frac{u_{i-1} - u_i}{\Delta x}$$

If $\psi(r) = 0$ the scheme is reduced to the first order upwind scheme.

3.2.4 Weighted ENO scheme: WENO 5 scheme

The essentially non-oscillatory schemes (ENO) are conceived by Harten and al. [15] in order to provide an useful tool to resolve problems with smooth solutions that contain discontinuities. They are high order interpolation schemes that are substantially non-oscillatory. In 1997 Jiang and Shu [16] created third and fifth order schemes WENO designing non linear weights for the smoothness management. The WENO schemes calculate the fluxes over different low interpolation order stencils to combine them and obtaining an higher-order scheme. They can avoid discontinuities problems automatically choosing the smoothest stencil. In particular in this work a fifth order WENO scheme is implemented.

We introduce the following convention:

$$D^+\phi_k = \phi_{k+1} - \phi_k, \qquad D^-\phi_k = \phi_k - \phi_{k-1}.$$

Now we can write the left and right biased derivatives of the stencil in Figure 3.4 for $\phi^-_{x,i}$ as

$$d_1 = \frac{D^+\phi_{i-3}}{\Delta x}, \quad d_2 = \frac{D^+\phi_{i-2}}{\Delta x}, \quad d_3 = \frac{D^+\phi_{i-1}}{\Delta x}$$
$$d_4 = \frac{D^+\phi_i}{\Delta x}, \quad d_5 = \frac{D^+\phi_{i+1}}{\Delta x},$$

and for $\phi^+_{x,i}$ as

$$d_1 = \frac{D^-\phi_{i+3}}{\Delta x}, \quad d_2 = \frac{D^-\phi_{i+2}}{\Delta x}, \quad d_3 = \frac{D^-\phi_{i+1}}{\Delta x},$$
$$d_4 = \frac{D^-\phi_i}{\Delta x}, \quad d_5 = \frac{D^-\phi_{i-1}}{\Delta x},$$

The third-order accurate ENO chooses one from the following schemes for the right and left flux calculation:

$$\phi^{\pm,0}_{x,i} = \frac{1}{3}d_1 - \frac{7}{6}d_2 + \frac{11}{6}d_3,$$
$$\phi^{\pm,1}_{x,i} = -\frac{1}{6}d_2 + \frac{5}{6}d_3 + \frac{1}{3}d_4,$$
$$\phi^{\pm,2}_{x,i} = \frac{1}{3}d_3 + \frac{5}{6}d_4 - \frac{1}{6}d_5.$$

Conversely, the WENO approximation of ϕ_x is a weighted combination of $\phi^{\pm,s}_{x,i}$ with $(s = 0, 1, 2)$, thus we get

$$\phi^{\pm}_{x,i} = \omega_0\phi^{\pm,0}_{x,i} + \omega_1\phi^{\pm,1}_{x,i} + \omega_2\phi^{\pm,2}_{x,i}. \tag{3.8}$$

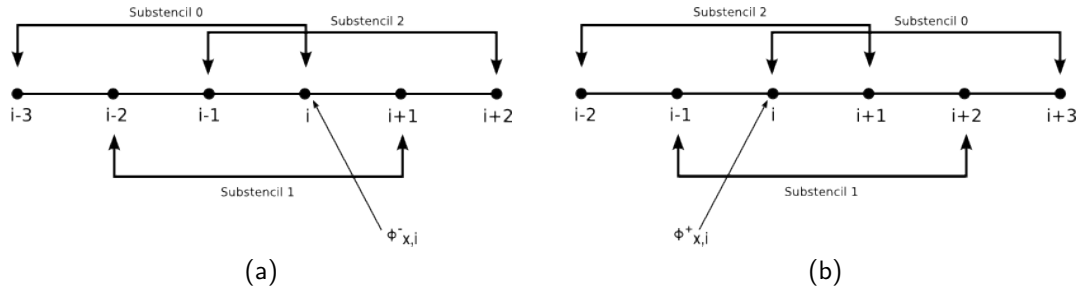
The weights are defined as

$$
\begin{aligned}
a_0 &= \frac{1}{10}\frac{1}{(\varepsilon + S_0)^2}, \qquad & \omega_0 &= \frac{a_0}{a_0 + a_1 + a_2}, \\
a_1 &= \frac{1}{10}\frac{1}{(\varepsilon + S_1)^2}, \qquad & \omega_1 &= \frac{a_1}{a_0 + a_1 + a_2}, \\
a_2 &= \frac{1}{10}\frac{1}{(\varepsilon + S_2)^2}, \qquad & \omega_2 &= \frac{a_2}{a_0 + a_1 + a_2}.
\end{aligned}
$$

Here

$$
\begin{aligned}
S_0 &= \frac{13}{12}(d_1 - 2d_2 + d_3)^2 + \frac{1}{4}(d_1 - 4d_2 + 3d_3)^2 \\
S_1 &= \frac{13}{12}(d_2 - 2d_3 + d_4)^2 + \frac{1}{4}(d_2 - d_4)^2 \\
S_2 &= \frac{13}{12}(d_3 - 2d_4 + d_5)^2 + \frac{1}{4}(3d_3 - 4d_4 + d_5)^2.
\end{aligned}
$$

We are using $\varepsilon = 10^{-6}$ to avoid the denominator becomes zero.



(a) (b)

Figure 3.4: *From the left, the left biased and the right biased WENO 5 schemes.*

Notice that we cannot always take advantage from the method to be fifth order, the actual order depends on the weights values. Indeed when there is a discontinuityin the field values the method prefers to choose the smoothest stencil for the calculation since the weights over the steepest stencils become negligible. We can exploit this feature to manage the calculation of the derivatives over the boundaries. Suppose we must calculate the value of the flux at i_m that is the nearest nodal point to the right boundary of the domain. We need to calculate the derivatives avoiding the ghost cells that physically have no meaning. In order to do that we store in the ghost cells high numbers (e.g. 10^8). Thus, considering the left biased flux, the derivative across the interface becomes orders of magnitude bigger in module than the others. Therefore the second and third weight become negligible since the denominators increase while the first weight

turns almost to one. The first weight evaluates only the first three differences of the stencil (Figure 3.4) that belong to the real domain. Notice that, since we store so high values in the ghost cells, it is *necessary* to ensure that at the boundaries the fluxes are directed outflow. In fact in our example the calculation of $\phi^+_{x,i}$ loses any meaning and would provide senseless results.

The WENO 5 method requires to add three ghost cell stripes all around the domain to well define the real boundary cells. That is clear if we pay attention to the way the stencil is built: the right side flux calculation requires three nodes on the right of the nodal point i while the left side flux calculation requires three nodes on the left of the nodal point i.

Remark. Since we are considering a solenoidal field, the equation (2.6) can be recast in the following form:

$$\frac{\partial \phi}{\partial t} + \nabla \cdot (\vec{u}\phi) = 0. \tag{3.9}$$

We obtain that simply considering that

$$\nabla \cdot (\vec{u}\phi) = \vec{u} \cdot \nabla\phi + \phi \nabla \cdot \phi \tag{3.10}$$

has the last term on the right-hand side equal to zero. Analytically using one equation or the other one makes no difference in the incompressible framework, otherwise if we analyze the numerical implementation the equations are not the same. Let us pay attention to the fact that the velocities values are stored on the interfaces of the cells while the values of ϕ are stored in the nodal points since we are adopting the staggered grid method. Furthermore, the fluxes of ϕ are stored on the cells interfaces. In the eq. (2.6) the calculation of the flux gradient is made before the divergence calculation and it consists in differentiating the flux over the cell. Thus the velocities are not considered in the physic flux calculation. This method is not written numerically in a conservative way. On the other hand the eq. (3.9) calculates the flux making the right property balance over the boundaries of the cell. The method is automatically conservative and keeps the value of the ϕ integration over the domain constant. In our specific case until the simulations are conducted with a velocity field that is independent from the physics of the system no problem emerges, however when we introduce the Navier-Stokes equations some troubles can occur. Nevertheless for the sake of simplicity we numerically discretized the equation (2.6) with a non implicitly conservative WENO 5 scheme in the "test code", then we implemented the eq. (3.9) for the main code.

3.3 Capturing and calculation of the interface

The Level Set methods belong to the interface-capturing methods since the key idea relies on the reconstruction of the interface handling an auxiliary field function. The field function is a scalar function defined in an Eulerian coordinate system. Therefore all the equations and quantities are defined in an Eulerian fashion with no need to introduce markers for the tracking as in the Lagrangian methods. Thus every equation for the transport of the auxiliary field function is written with the Cartesian grid as reference system.

In the chapter 1 we defined the Level Set function as a field function that lies in a space one dimension higher than the space where we define the interface. Therefore the Level Set function must be initialized over the entire domain even though the values of the interface are stored just in few points over the domain. The interface in most of our cases is localized where the level set function is zero, i.e. in the nodal points where the change in sign of the function is observed. Since the function of the surface is implicitly defined, we do not need to store the values of the interface in an additional array. All the information that we need will be deduced with thanks to the tools that we have available from the Level Set function theory when necessary.

In this section the mathematical models to resolve the transport equation are presented. First, we show the original method used within our study group. Such method is a good compromise between computational effort, ease in management and mass conservation capability. It is very good in keeping the right shape of the interface thanks to WENO 5 scheme but it still lacks too much in mass conservation. In fact in several test cases the trend to shrink is evident. Then the description of the new models follows. Notice that every proposed method is developed aiming to improve the global mass conservation.

3.3.1 Original method: the standard approach

The traditional method advects the Level Set function using the equation (2.6). The smoothness of the ϕ function makes the function to be easily resolved numerically. This equation will move the zero contour exactly as it is supposed to move. The flux direction was tracked by an upwind method, implemented in the code with the Godunov scheme (see equation 3.7) and both the right hand and left hand fluxes at the cell faces were calculated using the WENO 5 scheme. The WENO 5 scheme was used in conjunction with high-order TVD Runge-Kutta schemes. For example, a second or third-order Runge-Kutta method is written as a linear combination of Euler steps [33]. We show the time procedure adopted in the thesis to obtain the advance in time, that is, the third-order Runge-Kutta

scheme. First we define $\tilde{\phi}^{n+1}$ and $\tilde{\phi}^{n+2}$ with two consecutive Euler steps:

$$\frac{\tilde{\phi}^{n+1} - \phi^n}{\Delta t} = RHS(\phi^n),$$
$$\frac{\tilde{\phi}^{n+2} - \tilde{\phi}^{n+1}}{\Delta t} = RHS(\tilde{\phi}^{n+1}),$$

where $RHS(\phi)$ is the right-hand side of the transport equation. Thus we define an intermediate value $\tilde{\phi}^{n+\frac{1}{2}}$ by simple averaging

$$\tilde{\phi}^{n+\frac{1}{2}} = \frac{3}{4}\phi^n + \frac{1}{4}\tilde{\phi}^{n+2},$$

and then we advance $\tilde{\phi}^{n+\frac{1}{2}}$ for another time step doing

$$\frac{\tilde{\phi}^{n+\frac{3}{2}} - \tilde{\phi}^{n+\frac{1}{2}}}{\Delta t} = RHS(\tilde{\phi}^{n+\frac{1}{2}}).$$

Finally, ϕ^{n+1} is the linear combination of

$$\phi^{n+1} = \frac{1}{3}\phi^n + \frac{2}{3}\tilde{\phi}^{n+\frac{3}{2}}. \tag{3.11}$$

Since we discretized all terms explicitly, to preserve the stability there is a restriction in the time step. The restriction on the convective time step in a 2-D framework is given by the CFL condition

$$\Delta t\left(\frac{|u|_{max}}{\Delta x} + \frac{|v|_{max}}{\Delta y}\right) < 1, \tag{3.12}$$

where the subscript max specifies the maximum magnitude of the velocities.

While during the advance in time the interface keeps the right position, the Level Set function outside the zero level is distorted, no longer conserving the signed distance function properties (see chapter 2). In order to maintain the Level Set function a signed distance function we must apply an intermediate step after the transport. We adopted the eq. (2.11) to readjust the function to be a signed distance function. The reinitialization step always follows the transport equation resolution, but was not performed every time step. We chose to apply the reinitialization equation at least every 20 time-steps, mainly for two reasons:

- the resolution of the reinitialization equation every time step requires additional effort from the calculator, excessively protracting the CPU time;
- although the analytic solution of the equation provides a signed distance function with the zero iso-contour unchanged, the numerical resolution tends to shift the zero iso-contour.

Within every time-step the average slope of the signed distance function around the zero contour was monitored, to ensure that it does not excessively change its value. If $|\nabla\phi|$ moves too much away from 1 the algorithm reinitializes the function even if the counter did not reach 20.

In the reinitialization equation (2.11) the flux was calculated with WENO 5 schemes, the flux direction was checked using an upwind method, where the sign function stands in for the velocity. The sign function (2.12) is usually modified for numerical reasons as

$$\text{sgn}(d) = \frac{d}{\sqrt{d^2 + \varepsilon^2}} \tag{3.13}$$

introducing ε, usually equal to Δx, that is a small number to avoid the denominator becomes equal to zero. Furthermore in this thesis we implemented the correction proposed by Peng et al. [29]. They pointed out the existence of a bad behavior of the equation evolution if the ϕ function is too flat or too steep near the interface. In those conditions the function rebuilding goes too much slow when the function is too flat, on the other hand if the function is too steep this approach could change the sign of d, since it could move the interface across a grid point. Thus we put

$$\text{sgn}(d) = \frac{d}{\sqrt{d^2 + |\nabla\phi|\Delta x}} \tag{3.14}$$

into the intermediate equation as the sign function, to slightly alleviate the problem.

The reinitialization step is resolved until the steady state is reached in a band around the interface, or when the maximum number of iterations, set to 20, is reached. Pay attention to the fact that this number is *not* the maximum number of time-steps performed without reinitializing. To preserve the stability the CFL condition must be respected, thus our choice goes for $\Delta\tau = \Delta x/2$. It is not a strict condition, but it is enough to ensure the convergence of the solution. The residual was calculated as

$$res = \int_{\Omega_m} |\phi^{n+1} - \phi^n| \, d\Omega_m/\Omega_m \tag{3.15}$$

and as the criteria for steady-state we used

$$res < TOL \cdot \Delta\tau, \tag{3.16}$$

where the subscript m denotes a particular mask defined over the domain drawn around the interface and TOL is a specified tolerance. The tolerance that we chose is Δx^2.

3.3.2 First method: Re-distancing operator in the reinitialization

The first method is very similar to the original one, except for the fact that instead of using the regular reinitialization equation a new constrain was added in order to conserve the area of the domain. This change was devised by Sussman and Fatemi [34, 37] and it was addressed to make the iteration procedure more accurate and efficient. To preserve the volume of the domain it is required that

$$\partial_t \int_\Omega H(\phi)\, d\Omega = 0, \tag{3.17}$$

where the subscript t indicates the temporal derivative and H is the Heaviside function. It is clear that satisfying this constrain means keeping the area of the domain always constant. The equation for the reconstruction of the interface

$$d_\tau = \text{sgn}(d)(1 - |\nabla\phi|) = G(d,\phi), \qquad \phi(x,0) = d(x,t) \tag{3.18}$$

is rewritten in this way

$$\phi_\tau = G(d,\phi) + \lambda f(\phi), \tag{3.19}$$

and λ is a function of t only, determined by requiring

$$\partial_t \int_\Omega H(\phi)\, d\Omega = \int_\Omega H'(\phi)\phi_\tau\, d\Omega = \int_\Omega H'(\phi)(G(d,\phi) + \lambda f(\phi))\, d\Omega = 0. \tag{3.20}$$

Thus λ is calculated from the previous equation and we obtain

$$\lambda = \frac{-\int_\Omega H'(\phi)G(d,\phi)\, d\Omega}{\int_\Omega H'(\phi)f(\phi)\, d\Omega}. \tag{3.21}$$

To ensure that the correction is applied only on the interface without disturbing the function properties in the rest of the domain, the choice of f is

$$f(\phi) = H'(\phi)|\nabla\phi|. \tag{3.22}$$

The authors also provided a second order scheme for the discretization, but we prefer to keep working with the WENO 5 scheme as we did in the previous method. The advance in time was made with a third-order Runge-Kutta method (3.11). For the computation of the λ parameter we followed the choices of the authors, where:

$$\begin{aligned}
G(\tilde{\phi}_{i,j}^{n+1}) &\approx \frac{(\tilde{\phi}_{i,j}^{n+1} - \phi_{i,j}^n)}{\Delta\tau}, \\
H'(\phi_{i,j}) &\approx \frac{\partial H_\varepsilon(\phi_{i,j})}{\partial\phi_{i,j}} = \delta(\phi), \\
f(\phi_{i,j}) &\approx H'(\phi_{i,j})|\nabla\phi_{i,j}|, \\
\lambda_{i,j}(\tilde{\phi}_{i,j}^{n+1}) &\approx \frac{-\int_{\Omega_{i,j}} H'(\phi_{i,j}^n)G(\tilde{\phi}_{i,j}^{n+1})\, d\Omega_{i,j}}{\int_{\Omega_{i,j}} H'(\phi_{i,j}^n)f(\phi_{i,j}^n)\, d\Omega_{i,j}}.
\end{aligned} \tag{3.23}$$

The $\lambda_{i,j}$ parameter is assumed to be constant in each cell $\Omega_{i,j}$ near the interface. $\lambda_{i,j}$ is a function that disappears outside of a small band around the interface.

The method is ease to implement since it requires only the addition of a term in the intermediate equation. The additional calculation time required by the calculator is barely significant, almost negligible. The effect of this correction mostly aids the model avoiding the shrinking and introducing a slight improvement in the mass conservation.

3.3.3 Second method: source term embedded

The model proposed by Gorokhovski et al. [31] illustrates a new equation for the transport of the Level Set function that theoretically should keep the function close to a signed distance function when integrated in time. In fact the Eikonal equation ($|\nabla\phi| = 1$) is automatically satisfied by embedding a new source term in the transport equation. Therefore after the integration of the new Level Set equation the reinitialization procedure of the Level Set function should be no longer necessary. Furthermore the source term on the front is zero and the equation behaves as the regular equation for the Level Set function advection.

We introduce the new equation pointing the presence of the new term:

$$\begin{aligned} &\phi_t + \vec{u}\cdot\nabla\phi = A(\vec{x},t)\phi, \\ &\phi(\vec{x},0) = d(\vec{x}), \qquad |\nabla d(\vec{x})| = 1, \end{aligned} \tag{3.24}$$

where the coefficient $A(\vec{x},t)$ is an arbitrary regular function that does not depend on ϕ. It is clear that the evolution of the zero level set remains unaffected if we choose any regular $A(\vec{x},t)$ that does not depend on ϕ.

Moreover in terms of Eikonal equation, we have at all times

$$|\nabla\phi(\vec{x},t)| = 1, \quad t > 1. \tag{3.25}$$

The authors rewrote the equation in suffix notation. We propose the same convention for the calculation of the source term. Hence the equation becomes

$$\frac{\partial\phi}{\partial t} + u_k\frac{\partial\phi}{\partial x_k} = A(\vec{x},t), \tag{3.26}$$

where k is the notation used for the k-th dimension. With this notation the equation is valid for every dimension of the domain. Our work is just related to a 2-D domain. Differentiating the equation with respect to x_i (where i is the i-$th \neq k$-th dimension) and multiplying it by $2\nabla_i\phi$, after rearranging we obtain

$$\phi n_i\frac{\partial A(\vec{x},t)}{\partial x_i} + A(\vec{x},t) = n_i\frac{\partial u_k}{\partial x_i}n_k, \tag{3.27}$$

here $n_i = \frac{\nabla_i \phi}{|\nabla_i \phi|}$. Since the Eikonal equation (3.25) is satisfied even for $t > 0$ we can write $n_i = \nabla_i \phi$. We rewrite the eq. (3.27) in terms of normal derivatives, if we move along the characteristic of the differential equation we can consider a point $\vec{x}$ from the interface that pinpoints a distance n. Clearly along this line the components of $n(\vec{x}, t) = \nabla\phi(\vec{x}, t)$ do not change. Since the eq. (3.25) is always valid, ϕ aligns with n:

$$\phi = n.$$

Thus we can rewrite the eq. (3.27) as

$$\frac{\partial(An - u_k n_k)}{\partial n} = 0. \tag{3.28}$$

Solving the differential equation provides the exact solution

$$A(\vec{x}, t)n = u_k n_k - (u_k n_k)|_{n=0}, \tag{3.29}$$

where the suffix $n = 0$ denotes the condition $\phi(\vec{x}, t) = 0$. If we go back to the Level Set formulation, the equation turns in

$$\frac{\partial \phi}{\partial t} + u_k \frac{\partial \phi}{\partial x_k} = (u_k - (u_k)|_{n=0}) \frac{\partial \phi}{\partial x_k}. \tag{3.30}$$

Therefore the reinitialization equation is no longer necessary, but the problem of calculating $(u_k)|_{n=0}$ appears. We need to calculate it at each point and time to obtain the value of the source term A. That is not a trivial problem, and creates a lot of numerical issues. Hence the authors proposed to use an approximate form of A avoiding the problem of calculating the exact source term at the front. However the authors and some others proposed a solution addressed in the direction of directly calculating the source term, if interested we recommend the reader to see [12, 28].

Taylor expansion and zero-order local approximation of the source term
The expansion of the source term in a Taylor series up to the second-order term is

$$A(\vec{x}, t) = A_0 + A_1 n + A_2 n^2 + O(n^3) \tag{3.31}$$

and the expansion of $u_k n_k$ up to the third-order term is

$$u_k n_k = (u_k n_k)|_{n=0} + \left(\frac{\partial u_k n_k}{\partial n}\right)\Big|_{n=0} n + \frac{1}{2}\left(\frac{\partial^2 u_k n_k}{\partial n^2}\right)\Big|_{n=0} n^2 + \frac{1}{6}\left(\frac{\partial^3 u_k n_k}{\partial n^3}\right)\Big|_{n=0} n^3 + O(n^4). \tag{3.32}$$

Hence using the eq. (3.29) we obtain

$$A_0 + A_1 n + A_2 n^2 + O(n^3) = \left(\frac{\partial u_k n_k}{\partial n}\right)\Big|_{n=0} + \frac{1}{2}\left(\frac{\partial^2 u_k n_k}{\partial n^2}\right)\Big|_{n=0} n + \frac{1}{6}\left(\frac{\partial^3 u_k n_k}{\partial n^3}\right)\Big|_{n=0} n^2 + O(n^4). \quad (3.33)$$

Now, the zero-order term is

$$A_0 = \left(\frac{\partial u_k n_k}{\partial n}\right)\Big|_{n=0}. \quad (3.34)$$

Evaluating A_0 exactly on the interface is not easy since the quantities are known on the nodal points. Thus we evaluate A_0 on the nearest grid point $\vec{x}$ to the front and we recast the eq. (3.34) as

$$\left(\frac{\partial u_k n_k}{\partial n}\right)\Big|_{n=0} = \left(\frac{\partial u_k n_k}{\partial n}\right)\Big|_{\vec{x}} + O(n). \quad (3.35)$$

Thus the approximation to the zero-order coefficient takes the following form

$$A_{LA,0}(\vec{x},) = \left(\frac{\partial u_k n_k}{\partial n}\right)\Big|_{\vec{x}}, \quad (3.36)$$

where LA stands for *Local Approximation*.

Numerical implementation
The equation we are going to solve is the following, again, in suffix notation:

$$\frac{\partial \phi}{\partial t} = -u_k \frac{\partial \phi}{\partial x_k} + A_{LA,0}\phi. \quad (3.37)$$

We recalled the first term of the equation on the right as RHS_1 and the second term on the right side, i.e. the source term $A_{LA,0}\phi$, as RHS_2. The sum of these two terms is the right hand side of the equation. We resolved them separately. In particular RHS_1 was resolved in the same way as in the original method, where the divergence was discretized with the WENO 5 scheme. The source term gave more problems. First of all we redefined the zero-order coefficient. Aided by the following definitions $n_k = \frac{\partial \phi}{\partial x_k}$, $\frac{\partial}{\partial n} = \frac{\partial \phi}{\partial x_j}\frac{\partial}{\partial x_j}$ the eq. (3.34) can be rewritten in the following form:

$$A_{LA,0}(\vec{x}, t) = \frac{\partial \phi}{\partial x_j}\frac{\partial u_k}{\partial x_j}\frac{\partial \phi}{\partial x_k}. \quad (3.38)$$

We adopted two different approaches to discretize this term. The first is a "rough" method, we just reduced the derivatives with the WENO 5 scheme and we applied a first-order forward scheme for the velocity derivative. We stress that the velocity was averaged over the cell, due to the de-localization of the non-conservative formula (see the *remark* in section 3.2.4). This method works good until the system to be solved is very simple and without the coupling with the Navier-Stokes equations. However the goal of the model is to manage physical problems where high velocities, huge pressure gradients and thus quick topological changes occur. The "rough" scheme is not stable, and shows its limits in those situations. For some examples see chapter 4,5. The second approach uses the WENO 5 paired with the local Lax-Friedrichs function, providing the following numerical Hamiltonian:

$$\begin{aligned} \hat{RHS}(\phi_x^+, \phi_x^-, \phi_y^+, \phi_y^-) = RHS\left(\frac{\phi_x^+ + \phi_x^-}{2}, \frac{\phi_y^+ + \phi_y^-}{2}\right) \\ - \alpha(\phi_x^+, \phi_x^-)\frac{\phi_x^+ - \phi_x^-}{2} - \beta(\phi_y^+, \phi_y^-)\frac{\phi_y^+ - \phi_y^-}{2}, \end{aligned} \tag{3.39}$$

where

$$\begin{aligned} \alpha(\phi_x^+, \phi_x^-) &= \max_{\substack{\phi_x \in I(\phi_x^-, \phi_x^+) \\ \phi_y \in [C,D]}} |RHS_1(\phi_x, \phi_y)|, \\ \beta(\phi_y^+, \phi_y^-) &= \max_{\substack{\phi_x \in [A,B] \\ \phi_y \in I(\phi_y^-, \phi_y^+)}} |RHS_2(\phi_x, \phi_y)|. \end{aligned}$$

Here RHS_1 and RHS_2 stand for the partial derivatives of RHS with respect to ϕ_x and ϕ_y; $[A, B]$ and $[C, D]$ are a range of discrete values within $\phi_x^\pm$ and $\phi_y^\pm$ respectively; $I(a, b) = [\min(a, b), \max(a, b)]$.

The advantage of this scheme is brought with the introduction of the viscous terms α and β that produce artificial diffusion in the equation. The artificial diffusion helps the stability of the method and avoids undesirable oscillations. On the other side the solution will be less accurate. Indeed in the experiments showed in this thesis it is evident that the viscous terms enhance the loss of mass problem if they are not tuned accurately. Thus the choice of the viscous terms values was a tricky issue. Here we presented the original fashion to derive them, but it was not rare the case where we had to fix them manually.

The time integration was accomplished using the third-order Runge-Kutta scheme (3.11). Since we discretized all terms explicitly the stability must be preserved introducing restriction on the time-step. We used the restriction (3.12) for the convective time-step and a new restriction in time due to the source term.

The new restriction in time is the following:

$$M\Delta t < 1, \qquad M = \max \left| \frac{\partial u_k}{\partial x_i} \frac{\partial \phi}{\partial x_i} \frac{\partial \phi}{\partial x_k} \right| = \max \left| S_{ik} \frac{\partial \phi}{\partial x_i} \frac{\partial \phi}{\partial x_k} \right| \tag{3.40}$$

where S_{ik} is the rate of strain tensor.

As we will see at a later time this condition is not always strict enough. In effect the stability problem will be the main concern about this method, we will be forced to further diminish the time-step in order to reach the convergence and do not let the code blowing up.

3.3.4 Third method: conservative phase-field method

The resolution of the reinitialization equation brings too many drawbacks in its numerical implementation. On the other hand it is basically essential the reinitialization step due to the nature of the advection equation. Nevertheless, several attempts to abolish such equation exist. In particular the Olsson and Kreiss idea [23] is very interesting and we are going to present it in this section.

Instead of transporting the signed distance function, the Heaviside function (2.14) is directly transported. Thus the initialization of the function is made building a Heaviside function without the concern of respecting the Eikonal equation. In the numerical implementation we preferred to use the smeared-out Heaviside function (2.20). Analytically what we want is

$$\tilde{\phi}(\vec{x}) = H_{sm}(\phi(\vec{x})), \tag{3.41}$$

where H_{sm} is the smeared Heaviside function (2.20) but taken with a different convention, that is $H_{sm} = 1 - H_\varepsilon$. The goal is to obtain a conservative numerical method to advect the $\tilde{\phi}(\vec{x})$ that preserves its smooth profile. Having this method means to ensure the conservation of the integral $\int \phi$ over the domain. The Heaviside function defines two sub-regions on the domain, and we expect that also the sub-regions behave in a conservative way. In particular we set the interface at the level $\phi = 0.5$ and we want the area of the inner region to be $A_{\tilde{\phi}=0.5} \approx \int \phi$. From now we rename $\tilde{\phi}(\vec{x})$ as $\phi(\vec{x})$, which becomes our phase-field function.

To advect the phase-field function let us consider the standard equation (2.6). Since for incompressible flows the $\vec{u}$ field is always divergence free we can write

$$\phi_t + \nabla \cdot (\phi \vec{u}) = 0, \tag{3.42}$$

and we can adopt this form for the numerical resolution. The authors stressed the following points for the choosing of the numerical method:

- the method should be conservative;

- no spurious oscillations should be introduced;
- the thickness of the interface and the profile of ϕ should be kept constant.

A conservative method can be written in the following form

$$\frac{\phi_{i,j}^{n+1} - \phi_{i,j}^{n}}{\Delta t} = -\frac{1}{\Delta x}(F_{i+\frac{1}{2},j} - F_{1-\frac{1}{2},j}) - \frac{1}{\Delta y}(G_{i,j+\frac{1}{2}} - G_{i,j-\frac{1}{2}}), \tag{3.43}$$

where $F_{i+\frac{1}{2},j}$ and $G_{i,j+\frac{1}{2}}$ are approximations of the flux $(F, G) = \phi\vec{u}$ on the staggered grid. We calculated the fluxes using the central averaging

$$F_{i+\frac{1}{2},j} = \frac{1}{2}(\phi_{i,j} + \phi_{i+1,j})u_{i+\frac{1}{2},j}, \qquad G_{i,j+\frac{1}{2}} = \frac{1}{2}(\phi_{i,j} + \phi_{i,j+1})v_{i,j+\frac{1}{2}}. \tag{3.44}$$

Although the method is conservative, the central difference introduces oscillations on the front. Hence, in addition to this scheme we introduced also a TVD scheme with superbee limiter (see subsection 3.2.3) that damps the oscillations and helps in the preserving of the interface thickness.

Intermediate step
The advection of the phase field cannot keep by itself the thickness of the interface constant, nor the right shape of the smeared-out Heaviside function. It is necessary to introduce a new equation, an intermediate step addressed to solve this problem. We can add artificial compression in order to maintain the resolution of contact discontinuities as proposed by Harten [14]. The artificial compression can be viewed as an intermediate equation that is solved after the advection equation:

$$\phi_\tau + \nabla \cdot \vec{f}(\phi) = 0, \tag{3.45}$$

where f is the compressive flux. We want that the flux operates only in the region where $0 < \phi < 1$ and along the normal direction to the interface. Thus we can choose $\vec{f} = \phi(1-\phi)\hat{n}$, where $\hat{n}$ is the unit vector normal to the interface calculated using the definition (2.9). The equation in this form is a conservation law that forces the reconstruction of the function over the front and keeps the area constant through the time. To avoid discontinuities at the interface we introduced a small dissipation term ε modifying the conservation law:

$$\phi_\tau + \nabla \cdot \vec{f}(\phi) = \varepsilon\Delta\phi. \tag{3.46}$$

Thus we can write it in the conservative form:

$$\phi_\tau + \nabla \cdot \tilde{f}(\phi) = 0, \tag{3.47}$$

where

$$\tilde{f}(\phi) = \vec{f}(\phi) - \varepsilon\nabla\phi. \tag{3.48}$$

That ensures the interface thickness to be constant and defined by a finite value proportional to ε. For the implementation of the method we used the (3.48). The numerical reduction of the mathematical model is made using the conservative scheme

$$\frac{\phi_{i,j}^{n+1} - \phi_{i,j}^{n}}{\Delta\tau} = -\frac{1}{\Delta x}(F_{i+\frac{1}{2},j} - F_{i-\frac{1}{2},j}) - \frac{1}{\Delta y}(G_{i,j+\frac{1}{2}} - G_{i,j-\frac{1}{2}}), \tag{3.49}$$

with $F_{i+\frac{1}{2},j}$ and $G_{i,j+\frac{1}{2}}$ approximate the flux at the cell faces. In this case

$$F_{i+\frac{1}{2},j} = \frac{f(\phi_{i,j}) + f(\phi_{i+1,j})}{2} - \varepsilon\frac{\phi_{i+1,j} - \phi_{i,j}}{\Delta x},$$
$$G_{i,j+\frac{1}{2}} = \frac{g(\phi_{i,j}) + g(\phi_{i,j+1})}{2} - \varepsilon\frac{\phi_{i,j+1} - \phi_{i,j}}{\Delta y}.$$

The normal $\hat{n}_{i,j}$ is calculated just once for each intermediate step and then kept fixed until the convergence is reached. We also implemented an intermediate step correction proposed in [24, 8] demanding to enhance the accuracy of the front definition. The variation of the velocities in the domain distorts the shape of the interface, thus a stabilizing term is introduced. The intermediate step is recast by:

$$\phi_\tau + \nabla\cdot\vec{f}(\phi) = \varepsilon\nabla\cdot((\nabla\phi\cdot\hat{n})\hat{n}), \tag{3.50}$$

we rewrite it in a conservative form, thus

$$\phi_\tau + \nabla\cdot\tilde{f} = 0,$$

where

$$\tilde{f} = \vec{f} - \varepsilon((\nabla\phi\cdot\hat{n})\hat{n}).$$

To resolve it we use the conservative scheme (3.49) and for the fluxes we use

$$F_{i+\frac{1}{2},j} = \frac{f(\phi_{i,j}) + f(\phi_{i+1,j})}{2} - \varepsilon((\nabla\phi_{i+\frac{1}{2},j}\cdot\hat{n}_{i+\frac{1}{2},j})\hat{n}_{i+\frac{1}{2},j}),$$
$$G_{i,j+\frac{1}{2}} = \frac{g(\phi_{i,j}) + g(\phi_{i,j+1})}{2} - \varepsilon((\nabla\phi_{i,j+\frac{1}{2}}\cdot\hat{n}_{i,j+\frac{1}{2}})\hat{n}_{i,j+\frac{1}{2}}).$$

The normal and the gradient, only for this last scheme, are calculated with forward differences, it is obviously related with the fact that we want to keep coherence in the operations between the values stored in the face cells throughout the entire domain. The $\hat{n}_{i,j}$ is just calculated once for every time-step, in other words we used the same value of the normal until the steady-state is reached.

For the time integration we used a Runge-Kutta scheme, that coupled with a TVD scheme ensures the stability of the method. The ε is chosen to be dependent on the grid size in the following way:

$$\varepsilon = \frac{(\Delta x)^{1-d}}{2}. \tag{3.51}$$

d is usually equal to zero, but for the most difficult cases (i.e. the vortex field) we take $d = 0.1$ to improve the stability. For the resolution of the intermediate equation we use as fictitious time-step

$$\tau = \frac{(\Delta x)^{1+d}}{2}. \tag{3.52}$$

Using this method we no longer need to initialize the ϕ field as a signed distance function. Now the phase field is initialized as a smeared-out Heaviside function with values equal to one in the inner region, equal to zero in the rest of the domain with a smooth transition between the two phases across the thickness of the interface. There are two easy ways to set the initial condition of the ϕ function. We can initialize the first phase storing ones, the second one storing zeros and then resolve the artificial compression equation until it reaches convergence; or if we deal with simple cases we can use the directly the analytic equation. The authors provided the analytic equation for the circle

$$\phi = (1 + e^{(|\vec{x}-\vec{x}_c|-r)/\varepsilon})^{-1}, \tag{3.53}$$

and for the horizontal interface

$$\phi = (1 + e^{(y-y_{int})/\varepsilon})^{-1}. \tag{3.54}$$

No matter how we decide to initialize the ϕ function, the interface will lie on $\phi = 0.5$.

3.4 Navier-Stokes equations

The first part of the thesis is related to the simple advection of the phase-field to make a verification of the resulting numerical model. Our concern is also simulating a real physical system combining the transport equation of the phase field with the Navier-Stokes equations. In the aerospace propulsion field the description of the phenomena that govern the fuel injection, the combustion and the flame propagation are crucial in order to obtain better efficiencies, yields and performances. In this context, the falling droplet is a good benchmark for

a multi-phase solver, in fact it involves all the typical problems of a multiphase flow as the surface tension forces. We remind the reader that our solver does not consider the equation of the energy, every test was drove with the hypothesis of iso-thermal conditions and therefore the transfer of the mass across two phases was not considered.

In this section we described the law of the mass conservation and the law of the momentum conservation. Then we focused our attention on the discretizations of these equations and their numerical resolution.

3.4.1 Momentum and mass equations

The one fluid approach described by Tryggvason [39] and Chang et al. [6] was chosen. A weak formulation of the incompressible multi-phase flow was derived by coupling the Level Set formulation with the fluid equations. Basically one set of equations was resolved over the entire domain and the interface effects, i.e. the surface tension, were added only in a tiny band around the interface. We used the dimensionless Navier-Stokes equations in the context of the one fluid approach valid for an incompressible, viscous, unsteady and immiscible multi-phase system. Thus we write

$$\frac{\partial \vec{u}}{\partial t} + \nabla \cdot (\vec{u}\vec{u}) = \frac{1}{\rho}\left[- \nabla p + \frac{1}{Re}\nabla \cdot (2\mu D) - \frac{1}{Fr^2}(\rho - 1)\mathbf{k} + \frac{2}{We}\kappa\delta(n)\mathbf{n}\right], \quad (3.55)$$

where $\vec{u}$ is the velocity, p is the dynamic pressure, μ the dynamic viscosity, $\mathbf{k}$ the z-direction versor, ρ the density, κ the mean curvature, δ is the one dimensional delta function of the normal coordinate n and $\mathbf{n}$ the normal direction versor. Re is the Reynolds number, Fr is the Froude number and We is the Weber number defined as follows:

$$Re = \frac{\rho_g U L}{\mu_g}, \qquad Fr^2 = \frac{U^2}{gL}, \qquad We = \frac{\rho_g U^2 L}{\sigma}. \quad (3.56)$$

Here, U is the reference velocity, L is the reference length, ρ_g and μ_g are the density and dynamic viscosity of the reference phase that in our case is the gas phase, g is the gravity acceleration and σ the surface tension coefficient. Therefore, the right hand terms of the equation (3.55) correspond to, from left to right:

- pressure term. The pressure appears in the Navier-Stokes equation only in a differential form, thus it is relative to the reference system;
- viscous term, proportional to the Re number;
- buoyancy force that acts only along the vertical component of the system;

- the surface tension term. This term is localized only within the thickness of the smeared-out δ function.

We ensure the mass conservation introducing the continuity equation:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{u}) = 0. \tag{3.57}$$

The fluids are assumed to keep constant material properties, therefore the density and the viscosity of the fluid particles remain constant. Thus it is true that

$$\frac{D\rho}{Dt} = 0, \tag{3.58}$$

$$\frac{D\mu}{Dt} = 0. \tag{3.59}$$

Thanks to the (3.58) the continuity equation assumes the form

$$\nabla \cdot \vec{u} = 0, \tag{3.60}$$

that is, the velocity field for an incompressible system is always considered divergence-free.

3.4.2 Surface tension

The critical issue in the simulation of the multi-phase flow is the calculation of the singularities at the interface. The fluid properties like the density and the viscosity across the interface change discontinuously and thus it is hard to manage the numerical implementation. Moreover the surface tension is treated as a local source term only along the interface. Brackbill et al. [5] proposed to treat the interface as a finite thickness region across whom the function value varies continuously, that is, as a Continuum Surface Force. From here the name of the method CSF. Such reconstruction is not physically true, but allows to numerically resolve the problem in a stable fashion. Moreover every restriction to the topology of the problem is eliminated, indeed we will adopt a bi-phase system where the density ratio is 1000 without reporting stability problems.

Let us consider the last term of the equation (3.55). The CSF calculates the material properties exploiting the phase-field function ϕ that is defined in every point throughout the domain. The smeared-out Heaviside function (2.20) is used to obtain the values of the density and the viscosity taking into account the different phases. To calculate ρ and μ we defined the ratios

$$\eta_\rho = \frac{\rho_l}{\rho_g}, \qquad \eta_\mu = \frac{\mu_l}{\mu_g}, \tag{3.61}$$

thus

$$\rho_\varepsilon = \eta_\rho + (1-\eta_\rho)H_\varepsilon(\phi), \tag{3.62}$$
$$\mu_\varepsilon = \eta_\mu + (1-\eta_\mu)H_\varepsilon(\phi), \tag{3.63}$$

These are the values of the density and the viscosity smeared-out across the finite thickness of the interface. The surface tension is only present in the $\delta_\varepsilon(\phi)$ region across the interface.

Then, instead of solving the Navier-Stokes equations (3.55) we used a smeared-out version by the introduction of the ϕ function dependency:

$$\frac{\partial \vec{u}}{\partial t} + \nabla\cdot(\vec{u}\vec{u}) = \frac{1}{\rho_\varepsilon(\phi)}\left[-\nabla p + \frac{\nabla\cdot(2\mu_\varepsilon(\phi)D)}{Re} - \frac{(\rho_\varepsilon(\phi)-1)}{Fr^2}\mathbf{k} + \frac{\kappa\delta_\varepsilon(\phi)\nabla\phi}{We}\right]. \tag{3.64}$$

We write the normal $\mathbf{n}$ as $\nabla\phi$ according to the (2.9) since $|\nabla\phi| = 1$ every time the Level Set function is reinitialized as signed distance function. The smeared-out delta function corresponds with the (2.21).

Discretization

It is useful to show the numerical schemes used to resolve the surface problem. To calculate ρ at the cell faces we used:

$$\rho_{i+\frac{1}{2},j} = \frac{1}{2}(\rho_{i,j} + \rho_{i+1,j})$$
$$\rho_{i,j+\frac{1}{2}} = \frac{1}{2}(\rho_{i,j} + \rho_{i,j+1}),$$

and respectively, we did the same for μ. The gradient of ϕ is approximated using the centered differences:

$$(\nabla\phi)_{i,j} = \left(\frac{\phi_{i+1,j}-\phi_{i-1,j}}{2\Delta x}, \frac{\phi_{i,j+1}-\phi_{i,j-1}}{2\Delta y}\right) = \big((\phi_x)_{i,j}, (\phi_y)_{i,j}\big), \tag{3.65}$$

and the normal to the interface (2.9) becomes

$$\hat{n}_{i,j} = \frac{(\nabla\phi)_{i,j}}{|(\nabla\phi)_{i,j}|}. \tag{3.66}$$

The gradient of H is approximated with forward differences:

$$(\nabla H)_{i+\frac{1}{2},j} = \left(\frac{\phi_{i+1,j}-\phi_{i,j}}{\Delta x}, \frac{\phi_{i,j+1}-\phi_{i,j}}{\Delta y}\right). \tag{3.67}$$

The curvature κ is expressed as

$$\kappa = \frac{\phi_{yy}\phi_x^2 - 2\phi_x\phi_y\phi_{xy} + \phi_{xx}\phi_y^2}{\big(\phi_x^2+\phi_y^2\big)^{\frac{3}{2}}} \tag{3.68}$$

where the terms are discretized according the schemes presented in the section 3.2, and we centred on the cell faces

$$\kappa_{i+\frac{1}{2},j} = \frac{1}{2}(\kappa_{i,j} + \kappa_{i+1,j}) \tag{3.69}$$

to obtain the surface tension

$$F^x_{i+\frac{1}{2},j} = \frac{1}{We}\left(\frac{1}{\rho_{i+\frac{1}{2},j}}\kappa_{i+\frac{1}{2},j}(\nabla H)_{i+\frac{1}{2},j}\right),$$
$$F^y_{i+\frac{1}{2},j} = \frac{1}{We}\left(\frac{1}{\rho_{i,j+\frac{1}{2}}}\kappa_{i,j+\frac{1}{2}}(\nabla H)_{i,j+\frac{1}{2}}\right).$$

Since the third method (section 3.3.4) adopts another phase field function instead of the signed distance function, it was necessary to slightly modify the numerical scheme to make the method stable and without spurious oscillations around the interface. We took advantage from the fact that the term $\kappa\nabla H$ can be rewritten as

$$\kappa\nabla H = \nabla\kappa H - H\nabla\kappa, \tag{3.70}$$

where κ is independent from the spatial coordinate since it is defined locally on the interface. Thus the last term of the eq. (3.70) is equal to zero. Therefore we can calculate the surface tension in a more suitable way for our purpose:

$$F^x_{i+\frac{1}{2},j} = \frac{1}{We}\frac{1}{\rho_{i+\frac{1}{2},j}}\frac{1}{2}\big(\kappa_{i,j}(\phi_x)_{i,j} + \kappa_{i+1,j}(\phi_x)_{i+1,j}\big),$$
$$F^y_{i,j+\frac{1}{2}} = \frac{1}{We}\frac{1}{\rho_{i,j+\frac{1}{2}}}\frac{1}{2}\big(\kappa_{i,j}(\phi_y)_{i,j} + \kappa_{i,j+1}(\phi_y)_{i,j+1}\big).$$

3.4.3 Numerical solution of the N-S equations

The integration of the Navier-Stokes equations was performed with the projection method [7, 4]. The adopted procedure is as follows.

1. At the beginning of each time step the Level Set function is advanced in time from ϕ^n to ϕ^{n+1} as already said in the section 3.3 of this chapter.

2. The fluid acceleration term R_u due to inertial and viscous forces is defined as:
$$R_u = -\nabla\cdot(\vec{u}\vec{u}) + \frac{1}{Re}\frac{\nabla_h\cdot(2\mu_\varepsilon(\phi^{n+1})D)}{\rho_\varepsilon(\phi^{n+1})}, \tag{3.71}$$
and it is computed at the current time-step.

3. We integrate the equation (3.64) in time without considering the pressure gradient term. A provisional velocity $\vec{u}^*$ is computed:

$$\frac{\vec{u}^* - \vec{u}^n}{\Delta t} = \frac{3}{2}R_u^n - \frac{1}{2}R_u^{n-1} - \frac{\mathbf{k}}{Fr^2}\left(1 - \frac{1}{\rho_\varepsilon(\phi^{n+1})}\right) + \frac{1}{We}\frac{\kappa\delta_\varepsilon(\phi^{n+1})\nabla_h\phi^{n+1}}{\rho_\varepsilon(\phi^{n+1})}. \tag{3.72}$$

A multi-step explicit Adam-Bashforth scheme is adopted for the calculation of the tentative velocity.

4. A variable-coefficient Poisson's equation is solved to compute p^{n+1}, that is the pressure at the next time-step is:

$$\nabla_h \cdot \left(\frac{\nabla_h p^{n+1}}{\rho_\varepsilon(\phi^{n+1})}\right) = \frac{\nabla_h \cdot \vec{u}^*}{\Delta t}. \tag{3.73}$$

The solution of the equation is obtained by means of a preconditioned conjugate gradient method.

5. Finally, the provisional velocity $\vec{u}^*$ is corrected by the pressure term to obtain $\vec{u}^{n+1}$:

$$\vec{u}^{n+1} = \vec{u}^n - \Delta t\frac{\nabla_h p^{n+1}}{\rho_\varepsilon(\phi^{n+1})}. \tag{3.74}$$

CHAPTER 4

Advection of the Level Set function: results

In this chapter, the verification of the proposed numerical models is presented. Here we will resolve the advection equation of the ϕ function, using all the different methods introduced. When necessary, we will also resolve the reinitialization equation. The Navier-Stokes equations are not taken into account since no physical phenomenon is inspected; we will insert them in the next chapter. These tests are addressed to:

- arrange a serviceable numerical algorithm;
- test the working parameters of the model using various benchmarks;
- calculate the ability of each method in conserving the initial area and the initial shape;
- define limits and operative boundaries of the numerical models.

In order to accomplish the tasks we will simulate the models over three test cases. First, the revolution of a circular disk in a rotational field, second the the revolution of the Zalesak's disk in a rotational field, and as last the deformation of a circular disk immersed in a vortex flow.

DNS pretends to predict engineering flows without any approximation in the equations. Its ability in providing real results is paid with an high computational effort by the supercomputers. Due to the limited ability of the methods and algorithms to track the interfaces and the limitation in memory and speed of current supercomputers, we are not able to simulate complex problems with complex geometries. Thus, another crucial parameter of the methods that we are

going to verify is the CPU time required to complete the simulation. It is well know that the bottle neck of the simulation is the Poisson solver, however the advection of the ϕ function takes more or less 20% in time of the entire simulation overall. Since it is not a negligible number, we will calculate the average CPU time deployed to complete the experiments.

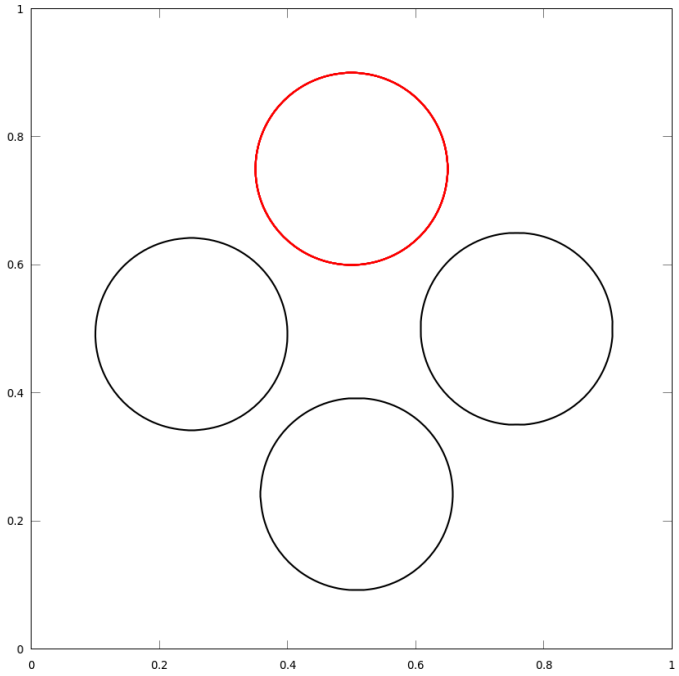


Figure 4.1: *The red shape is the initial condition of the circular disk at $t = 0$. The black shapes denote the evolution of the disk during the rotation.*

4.1 Circular disk rotation

In this simple test a circular interface is subjected to a rotational field. The computational domain is a square of unitary length $\Omega : [0, 1] \times [0, 1]$. The zero ($0.5$ for the *conservative method*) level set of the function is a circular shape with the center in $x_c = 0.5, y_c = 0.75$. The radius is $R = 0.15$. To initialize the surface as a signed distance function we can use the following analytic equation:

$$\phi = \sqrt{(x - x_0)^2 + (y - y_0)^2} - R. \tag{4.1}$$

Otherwise the reader can adopt any surface function with the zero level values corresponding to the circle drew and then apply the reinitialization step. When

we tested the *conservative method* we initialized the function with (3.53). The disk is convected by a rotational velocity field:

$$\begin{cases} u = 0.5 - y, \\ v = x - 0.5, \end{cases} \tag{4.2}$$

which represents a rigid body rotation with respect to $(0.5, 0.5)$. The Figure 4.1 depicts the case set-up.

We rotated the disk for $T = 6.28$ time units, that correspond to a complete revolution with respect to the center of the rotational field. After the experiments we compared the numerical solution with the analytic solution. We resolved the problem for the following grid sizes: 64×64, 128×128, 256×256. The *source term method* never reinitializes, the original and the *re-distancing method* apply the reinitialization every 20 time-steps. In this experiment, and in all the experiments that follow, the *conservative method* resolves the intermediate step every time-step. For every method we used $\Delta t = C\Delta x$ and $\Delta \tau = C\Delta x$, where the Courant number is chosen to be $C = 0.5$, that it was enough in both cases to ensure stability and damp oscillations. At the borders of the domain we can both adopt the Neumann conditions and the Dirichlet conditions as boundary conditions for the Level Set function. The remark is valid for every case we considered in this chapter.

In the table 4.1 we compared the area error and the distance error at the end of one revolution of the circular disk. The area is well conserved in all cases, especially with the grid refinement the results show up low errors. The *source term* method stands out since its area error is always at least one order of magnitude smaller than the others. Such a good behaviour is justified by the fact the method reinitializes just once in a while, thus the interface is never shifted from its exact position. The Figure 4.2 compares the evolution of the area between the *source term* method and the original method. The measure of the distance error is good for every case, but the *conservative method*.

The average CPU time to complete the experiments is reported in table 4.2. Notice that the grid discretization is the main reason in the increase of the computational time. We can observe that the *re-distancing method* is more demanding than the original one. We expected that since the introduction of a new additional term inside the reinitialization step. Also the *source term method* requires more time to accomplish the simulation, although the reinitialization equation is not resolved. This fact is explained considering the fact that the source term added to the equation is not straightforward to compute and needs additional efforts. The *conservative method* shows the lowest time, and gets a head start with respect to the others, especially for fine meshes. In effect numerical tests showed that only few iterations had to be performed in order to reach steady-state solu-

Table 4.1: *The area and the distance errors for the circular disk case are represented. The methods are tested with different grid sizes.*

	Method	64x64	128x128	256x256
$e_{m\%}$	Original	0.33	0.105	0.04
	Re-distancing	0.27	0.02	0.035
	Source term	0.02	0.0006	0.0001
	Conservative	1.22	0.002	0.006
		1×10^3	1×10^3	1×10^3
e_d	Original	2.54	0.89	0.52
	Re-distancing	1.41	0.73	0.41
	Source term	0.37	0.05	0.21
	Conservative	4.74	2.02	0.77

tion of the intermediate equation, compared to the average number of iterations that the reinitialization step requires.

4.2 Zalesak's disk rotation

We proposed as second test the rotation of the Zalesak's disk. This test is interesting since the shape presents sharp corners, thus we evaluated how much every method smears the corners out during the revolution. The domain is again a square with length equal to 1. The velocity field is the same as the previous section, that is the equation (4.2). The Zalesak's disk is centered in $x_c = 0.5$, $y_c = 0.75$ and the radius is $R = 0.15$. The slot width is 0.075 and the slot length is 0.25. The Figure 4.3 shows the case setup. The time step and the fictitious time step are $\Delta t = \Delta\tau = C\Delta x$. the Courant number is $C = 0.5$. One complete revolution of the disk was accomplished.

We summarized the results in the table 4.3. The *re-distancing method* works very good when the mesh is coarse, while it loses its advantage refining the mesh. The *source term method* is still better than the original one thanks once again to its ability not to reinitialize often. In terms of area conservation the *conservative method* seems to be the worst above all, but actually it is not true. In fact, let us consider the area value versus time of Figure 4.4. On the right we can see

Table 4.2: *Average CPU time elapsed to simulate the rotation of the circular disk. Time is expressed in seconds. Several tests were made in order to obtain an average time.*

Method	64x64	128x128	256x256
Original	11.66	64.35	540.08
Re-distancing	19.09	80.01	617.57
Source term	20.23	83.09	582.49
Conservative	7.23	30.47	215.97

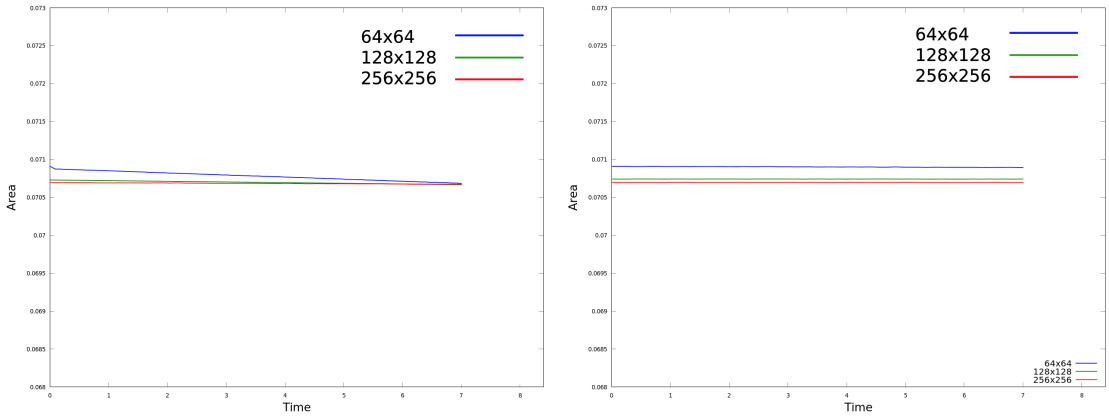


Figure 4.2: *Area conservation of the original method (left) and the source method (right) for the rotation of the circular disk.*

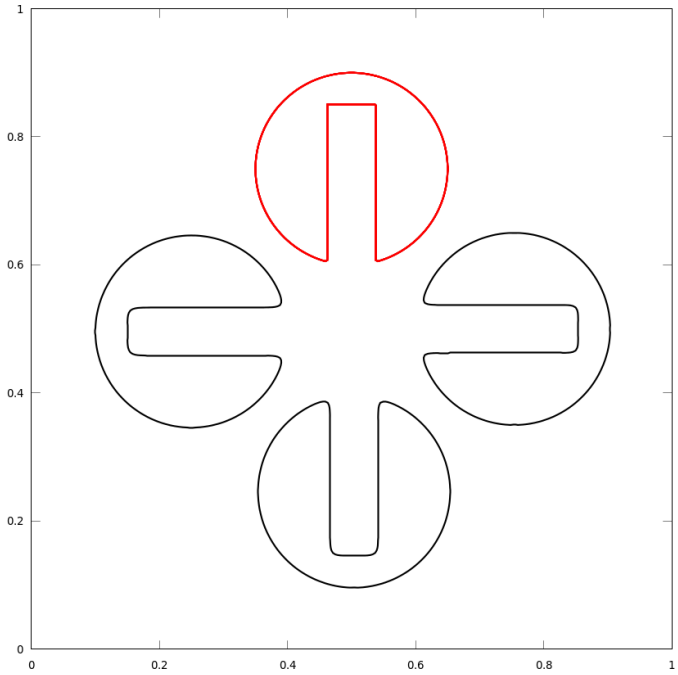


Figure 4.3: *The red shape is the initial condition of the Zalesak's disk at $t = 0$. The black shapes denote the evolution of the disk during the rotation.*

Table 4.3: *The area and the distance errors for the Zalesak's disk case are represented. The methods are tested with different grid sizes.*

	Method	64x64	128x128	256x256
$e_{m\%}$	Original	9.77	1.08	0.17
	Re-distancing	1.64	0.67	0.40
	Source term	7.75	0.74	0.04
	Conservative	-	5.02	2.63
e_d	Original	0.03	0.017	0.009
	Re-distancing	0.02	0.001	0.006
	Source term	0.02	0.009	0.005
	Conservative	-	0.015	0.010

the original method. For a coarse mesh it tends to lose area advancing in the time, without any gesture to adjust the problem. The main issue is that every time we apply the reinitialization step the shape shrinks and a little amount of area is lost. A fine mesh diminishes the problem, but doesn't prevent it. On the left the *conservative method* is shown. During the time, besides some rumor, if we look at the error calculation the area is perfectly conserved with a little deviation ($< 0.5\%$ for the coarsest grid). Even though there is a small initial offset of the area, there is no drift and the area does not increase nor decrease. We did not perform the 64×64 case for this method because problems in the initialization of the shape arose. The grid is too much coarse and the intermediate step scheme is not accurate enough to grant an acceptable result to be employed in the simulation.

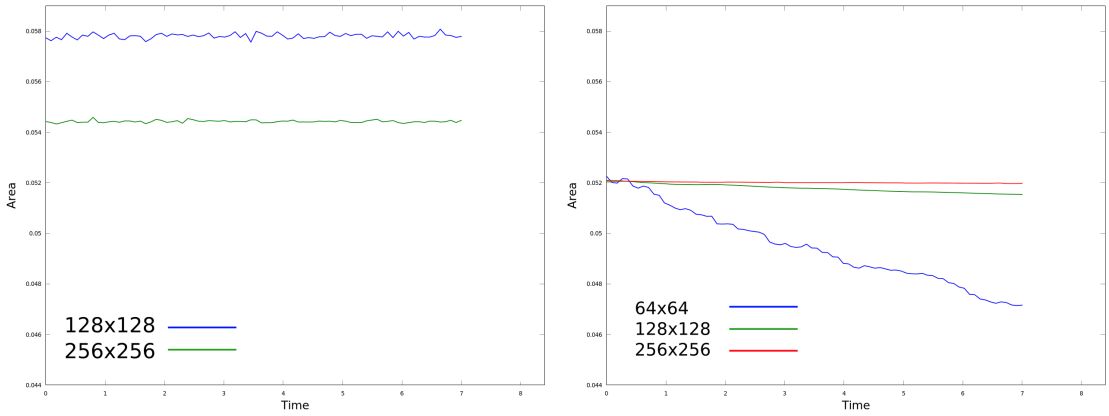


Figure 4.4: *Zalesak's disk area versus time. On the left: conservative method. On the right: original method*

In Figure 4.5 we compared the final stage of every method with the exact result after one turn, for the 128×128 grid. The most difficult parts to resolve are the corners, where the numerical schemes easily fail to reproduce the correct geometric shape. The *re-distancing method* slightly improves the accuracy with respect to the original method, while the *source term method* deploys the better approximation to the exact solution. The *conservative method* is not good in the shape conservation, even though his excellent ability to conserve area. The bottom-right picture from Figure 4.5 does not exactly respect the reality since the red shape is initialized as a signed distance function, while the blue shape was initialized by a Heaviside function.

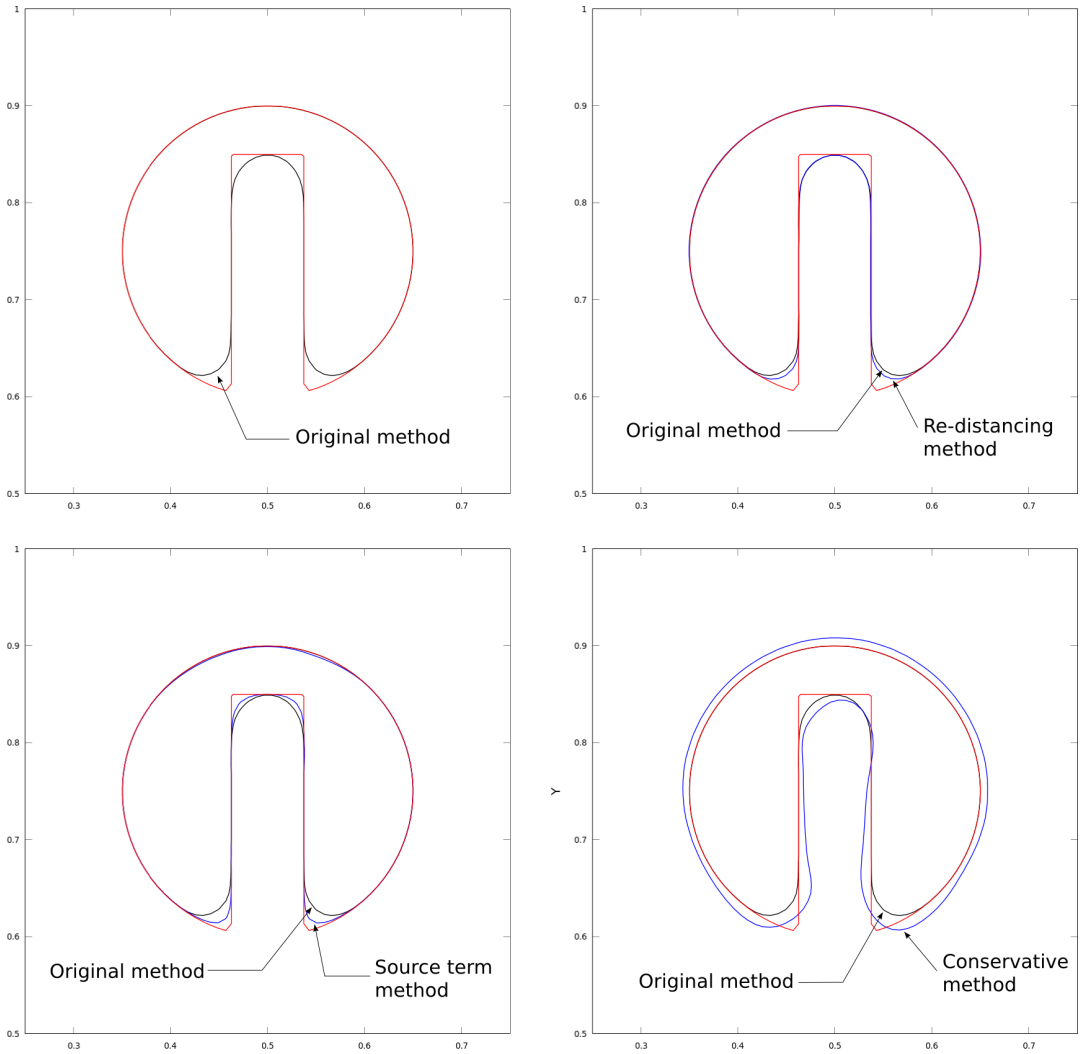


Figure 4.5: *Final stage comparison between numerical solutions (black, blue) and exact solution (red) on a* 128×128 *grid. Top-left: original method; top-right: re-distancing method; bottom-left: source term method; bottom-right: conservative method.*

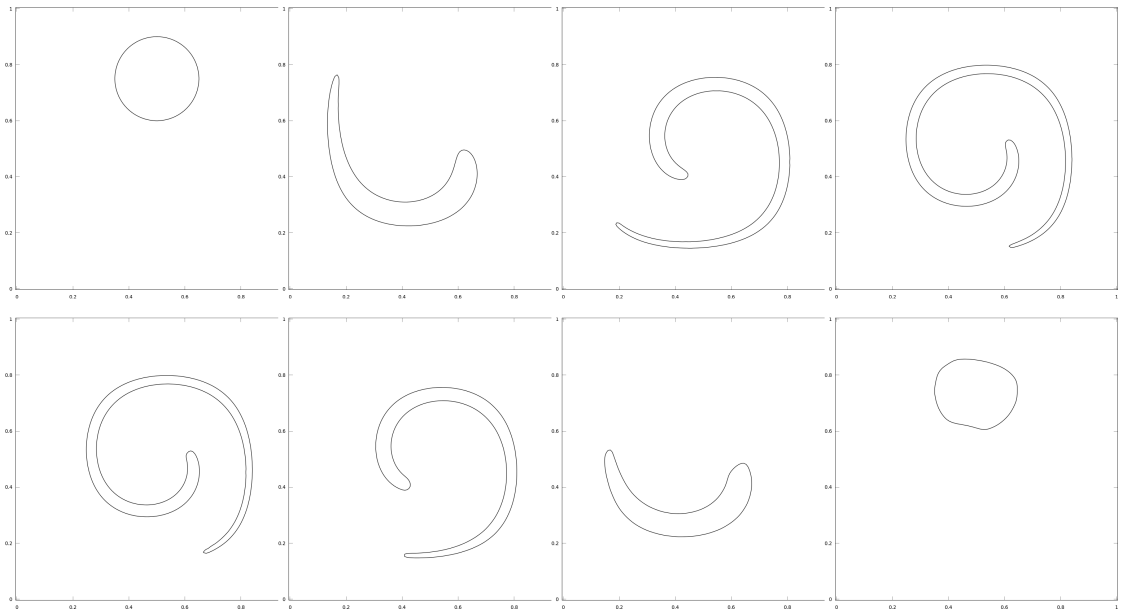


Figure 4.6: *Vortex field applied to a disk, original method.*

4.3 Vortex test

The vortex test is the most arduous between the three since the velocity field warps the original shape. We settled the circle in the same position of the circular disk in the rotation test, and the velocity field adopted is described by the following set of equations:

$$\begin{cases} u = -\sin^2(\pi x)\sin(2\pi y)\cos(\pi t/T), \\ v = \sin(2\pi x)\sin^2(\pi y)\cos(\pi t/T). \end{cases} \tag{4.3}$$

The twisting is made with respect to the central point of the domain, $(0.5, 0.5)$. The circle is stretched out into a filament and then reversed to the original position. The period of the round cycle T was chosen to be $T = 4$. At $t = T/2$ the disk reaches the maximum deformation, then it comes back to the initial state. The evolution of the disk is portrayed in Figure 4.6. Increasing the T period makes the tail of the filament resolution harder to catch if we use meshes not fine enough. Usually the grid refinement that we deployed was enough for our needs, but we have to remark that the *conservative method* struggled. In the Figure 4.7 the tail effect of the *conservative method* is shown. When the thickness of the shape becomes too much thin it will not be well resolved, and leaves behind "chunks" of area that detach from the original figure. This numerical effect can only be avoided if the thickness of the interface is smaller that the dis-

tance between two interfaces, that is, refining the mesh or reduce the value of ε. Changing ε is not easily viable, because ε influences the fictitious time-step and thus the overall method ability to converge.

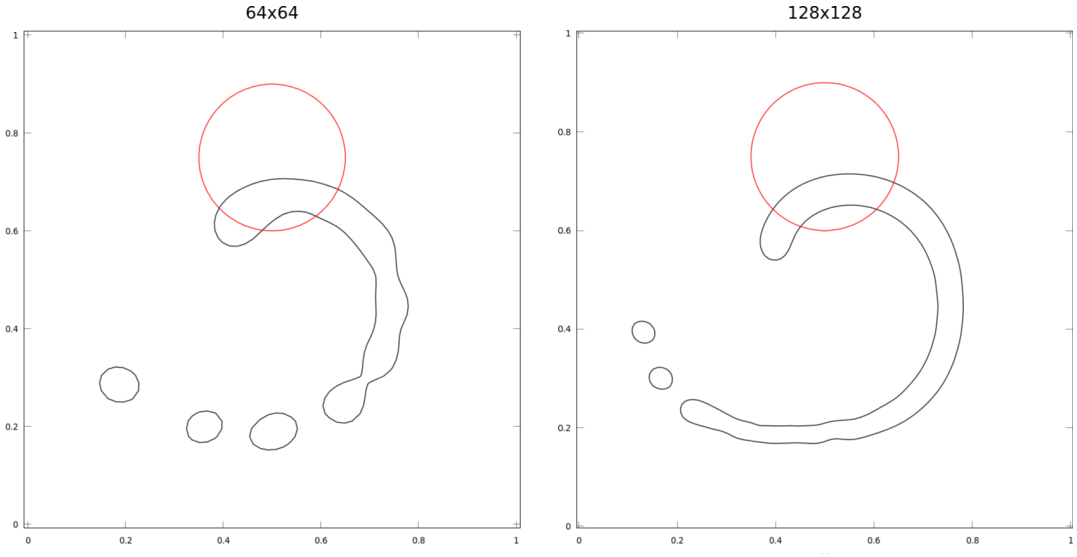


Figure 4.7: *Circular disk at $t = 0$ in red and at $t = T/2$ in black. The conservative method bad reproduces the stretched disk at the maximum elongation.*

The velocity field is solenoidal, thus ideally the shape at the end of the period T should be like the original circle at the starting point. In the numerical implementation some area will be lost for every test and the shape will be not conserved exactly. We simulated the methods over different grid sizes: 64×64, 128×128, 256×256. The time-step for the original method and the *re-distancing method* is unvaried. For resolving the vortex with the *conservative method* we opted for $d = 0.1$, where d is the parameter to control the fictitious time-step as shown in eq. (3.52). The time-step is unvaried. The *source term method* requests to adopt a very strict time-step condition. Although the Lax-Friedrichs scheme was applied, the code showed limitations in the stability and easily failed to converge unless we reduce the time-step. To get the job done it required $C = 0.1$, even though there was already an additional source term restriction (3.40) over the time-step. The fictitious time-step was unvaried.

The table 4.4 shows the ability of every method to conserve the area after one period of time. The *re-distancing* method behaves better than the original method if compared over the coarse mesh. Increasing the resolution of the mesh, the gain in adopting the modification to the reinitialization equation fades out. However, this method has no drawbacks compared with the original one. The

Table 4.4: *Area error after one period, $t = T$. Different grid sizes have been considered.*

Method	64x64	128x128	256x256
Original	22.79	4.87	1.40
Re-distancing	17.35	4.57	1.64
Source term	31.53	9.60	3.05
Conservative	0.46	0.26	0.17

Table 4.5: *CPU time elapsed in seconds to simulate an entire period of the vortex field. Several tests were made in order to obtain an average time.*

Method	64x64	128x128	256x256
Original	19.20	70.77	522.98
Re-distancing	19.92	76.77	554.56
Source term	78.80	448.52	5532.52
Conservative	16.53	40.08	243.33

only disadvantage could be a slightly increasing in the computational time, but totally negligible for the purposes of our job.

The *source term method* in this framework presents the worst behaviour. The area loss is higher than all the others as we can see from Figure 4.10, and moreover if we take a look at table 4.5, we see that its computational CPU time is about one order of magnitude higher than the average. We must stress that this is mostly due to the time-step restriction and not to the complexity of the numerical equations. In order to accomplish the entire revolution the *source term method* needs to reinitialize often than the previous cases. In fact the vortex field distorts excessively the Level Set function and the equation (3.37) seems no longer able to maintain it as a signed distance function. It is useless to remark that the reintroduction of the reinitialization equation further slows down the computation speed and lets the shape shift from its original position, nullifying the advantage that the method claims.

The *conservative method* shows a temporary area loss of the inner region since the shape is subject to the pinch at $t = T/2$ (Figure 4.8). The shape is however recovered at the final stage of the vortex and the area loss is small compared with the original method. We stress that the intermediate step always ensures the

conservation of the area. Moreover, if we calculate the integral $\int \phi$ throughout the entire domain the area error is closed to zero as it is shown in Figure 4.9, no matter the instant of time t we are considering, since the method is intrinsically conservative. In Figure 4.10 is shown the final stage of the disk. This method does not succeed in keeping the exact shape, this problem is related to the difficult to reproduce the tail when the distance between the interface becomes to small, as described above. This method stands out for the computational CPU time employed (table 4.5). For a coarse grid it saves about 14% in time while for the finest grid saves up to 53% of the time compared to the original method.

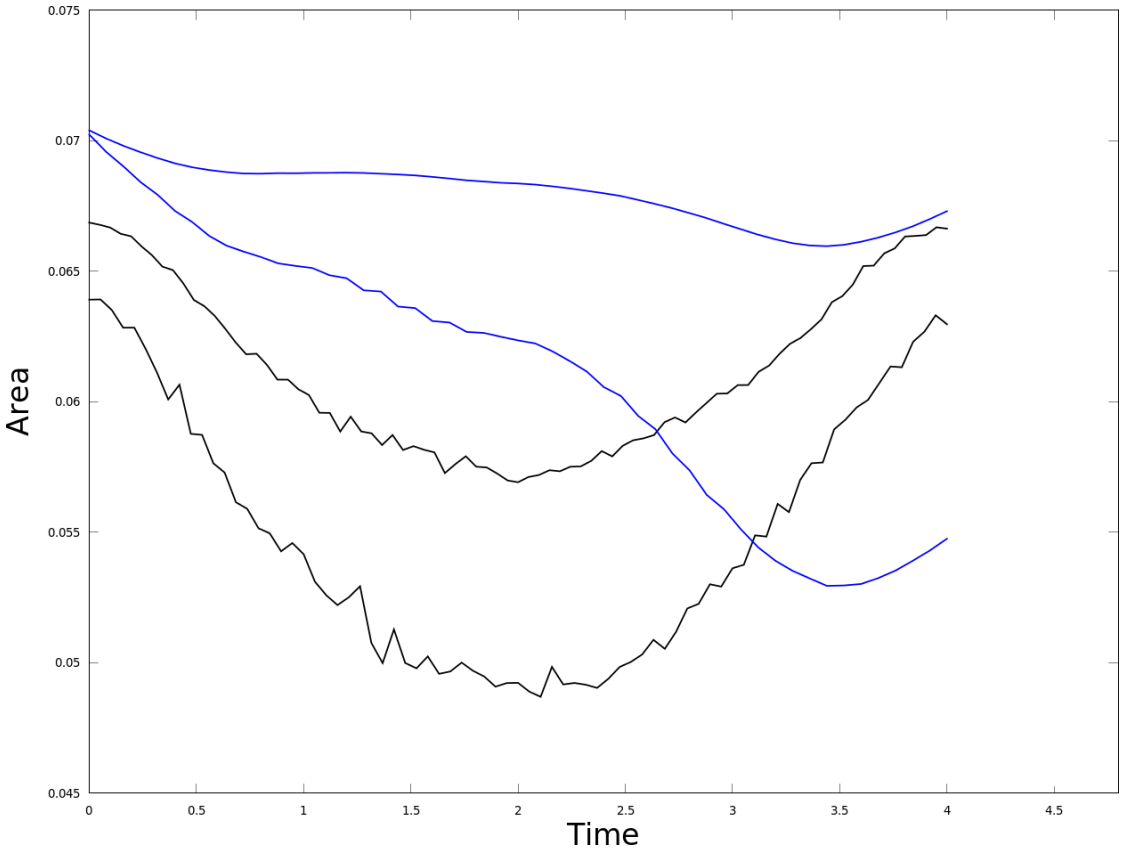


Figure 4.8: *Area comparison of the inner region between the conservative method and the original method. The* 64×64 *and* 128×128 *grid cases are represented.*

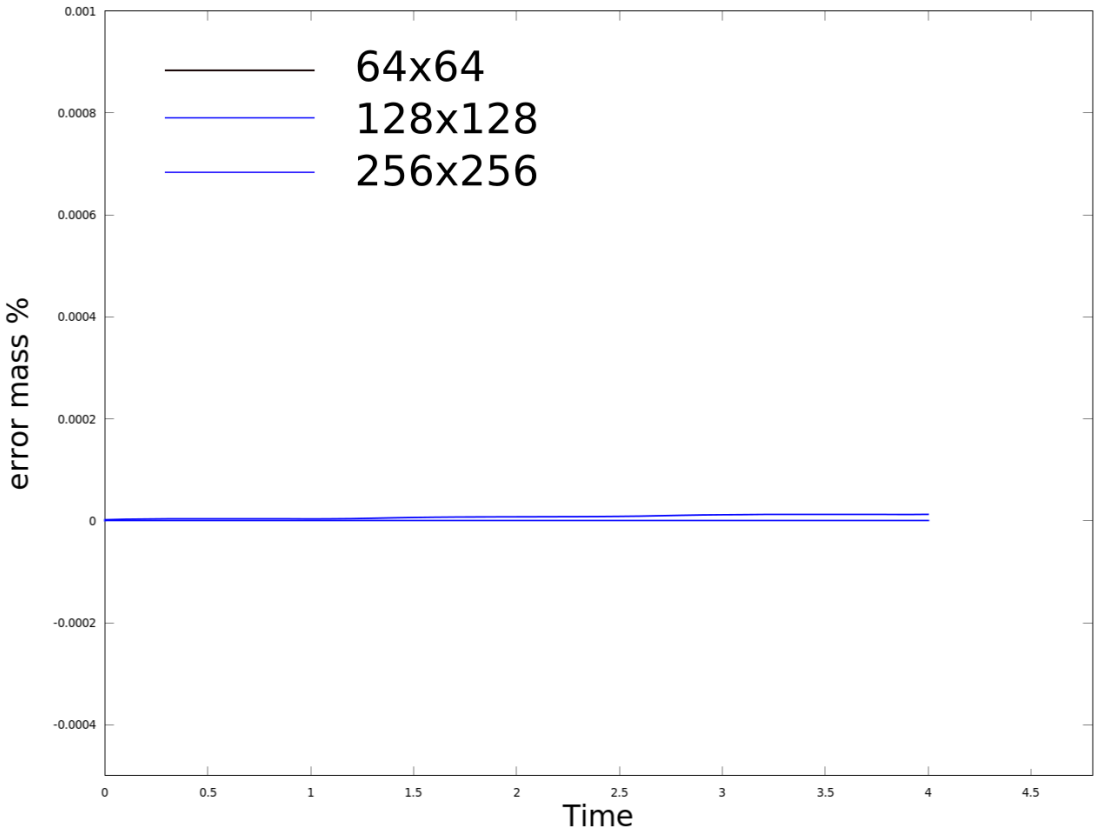


Figure 4.9: *Total area error with respect to the time of the conservative method. All the grid sizes are represented. It is difficult to identify the coarse mesh line since the overlapping.*

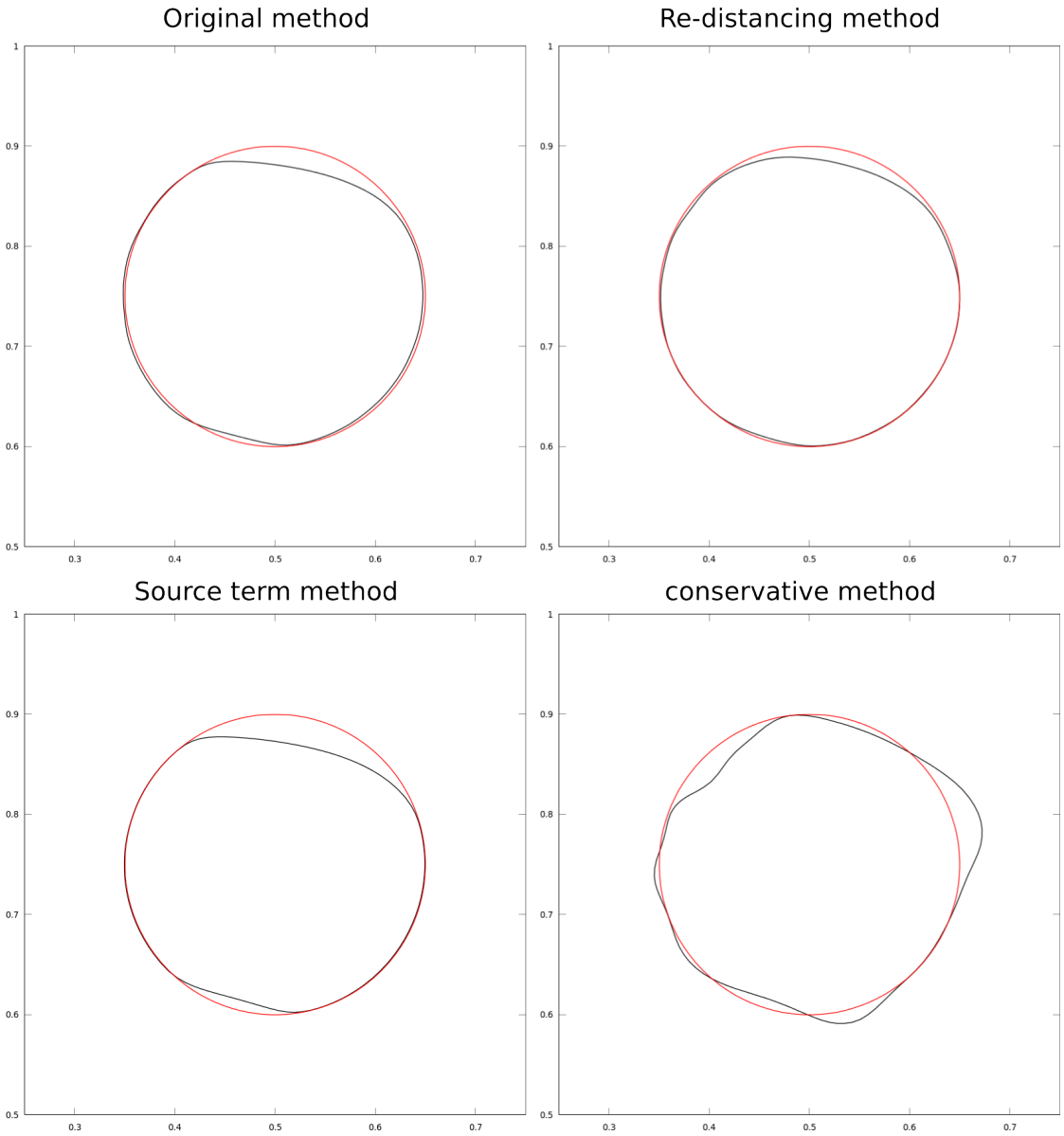


Figure 4.10: *Final stages of the vortex rotation for the various methods, with a* 128×128 *grid. Top-left: original method; top-right: re-distancing method; bottom-left: source term method; bottom-right: conservative method.*

CHAPTER 5

Results on a falling droplet

In this chapter the Level Set function methodology was applied to a physical phenomenon to simulate the interface between two different phases. The Navier-Stokes equations are added to the existent model in order to introduce a physical behaviour in the Level Set formulation. We investigated the evolution of a falling droplet in a stagnant flow, for a 2-D case. We are considering a system with low speed velocities involved ($Ma < 0.3$), thus we made the approximation of incompressible flow. Moreover we do not simulate the turbulence, in fact we cannot demand to simulate with a good accuracy the real turbulence phenomena since the 2-D limitation. In effect the scope of this chapter is mainly to verify the compatibility of the code developed with the Navier-Stokes equations. That is, we want to expand the usability of the proposed methods to the fluid-dynamics framework. This work supports a bigger project to develop a complete toolbox that simulates accurately a multi-fluid 3-D turbulent isotropic system with sharp interface (GFM) between two high-density ratio phases.

With the introduction of the surface tension several issues in the coupling of the phases emerged. It is common that spurious currents rise in the vicinity of the interface that have no physical meaning. As it was pointed out in [11], the main reason of the generation of these currents is a numerical unbalance between the tension forces and the relative pressure gradient. The CSF method is definitely subject to the phenomenon, but also the Ghost Fluid Method [17, 10, 20] is not immune to the problem. The problem is more sensitive for the VOF and FT methods. In fact the reconstruction of the interface via interpolation can lead to an erroneous or a not enough accurate calculation of the interface normals. Even though in a reduced scale, also the LSM is subject to spurious currents, in

the measure that the precision of the calculation of the normals lays on how well the auxiliary function ϕ is rebuilt in the interface neighbourhood.

Our efforts were addressed to minimize the numerical currents and obtain a 2-D simulation of a falling droplet. We then compared the results of the performed simulations, taking into account the ability of each method to conserve the mass.

5.1 Chosen scenario and physical framework

A water droplet falls in air under the effect of gravity and surface tension. The technique adopted to the treatment of the jump evolution is the CSF methodology. The treating of the interface evolution was made with, first, the original method, then with the proposed methods. The velocity field and the pressure were calculated with the one-fluid approach presented in chapter 3. For the simulations we used the following physical constants: $\rho_g = 1.204\,\mathrm{kg/m^3}$, $\rho_l = 998.2\,\mathrm{kg/m^3}$, $\mu_g = 1.983 \times 10^{-5}\,\mathrm{kg/(m\,s)}$, $\mu_l = 1.002 \times 10^{-3}\,\mathrm{kg/(m\,s)}$.

At $t = 0$ the droplet is initialized as an exact circle with initial diameter $d_0 = 1.250\,\mathrm{mm}$ and we assume the reference length to be $L = d_0$. Thus, the normalized droplet diameter is $d_0^* = 0.5$. The simulations were performed in a box of length $l_x = 10d_0^*$, that is 5 units and of height $l_y = 10d_0^*$, that is 5 units. The cell dimensions are $\Delta x = \Delta y = 1/6$ units for a total of 60 nodes along the x-axis and the y-axis. Approximately the diameter contains 7 nodes. The initial position of the center of the droplet is $(x_0, y_0) = (0.5l_x, 0.75l_y)$.

The reference velocity is $U^* = \sqrt{gL} = 0.110\,68\,\mathrm{m/s}$, where g is the gravity acceleration. The surface tension between the water phase and the air phase is $\sigma = 7.28 \times 10^{-2}\,\mathrm{N/m}$. Thus, the dimensionless groups assume the following values:

$$\begin{aligned} Re &= 9.372, \\ Fr^2 &= 1, \\ We &= 2.52 \times 10^{-4}. \end{aligned}$$

The simulation is stopped after 3 units of time, during this interval the droplet falls with constant acceleration increasing its velocity until it splashes against the boundary wall at the bottom. The Figure 5.1 displays the initial condition of the ϕ, where the black contour represents its zero level and the coloured function in the background is the Level Set function.

Boundary conditions
The rectangular dominion is delimited at the borders by boundary conditions.

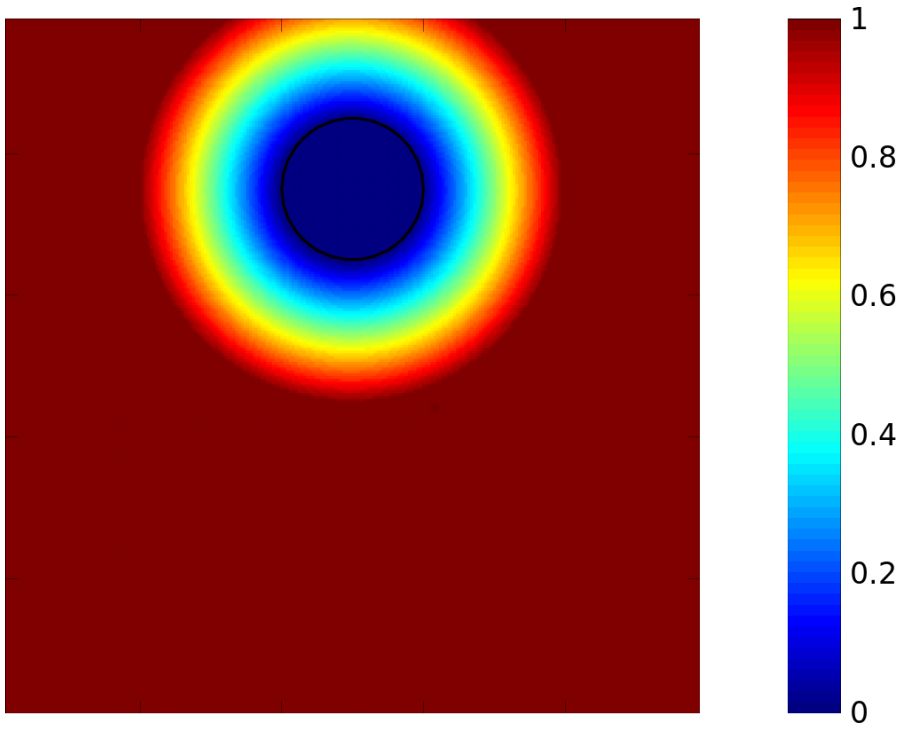


Figure 5.1: *Level Set function at $t = 0$ of the droplet initialized as described above.*

The top edge is open, while the lateral and the bottom edges are rigid walls. Let us consider as an example the west wall. The function ϕ along the wall respects the Neumann condition

$$\frac{\partial \phi}{\partial x} \approx \frac{\phi_{2,j} - \phi_{1,j}}{\Delta x}, \tag{5.1}$$

where the index i according with the Octave notation denotes the first element of the array, that is the ghost cell. The tangential velocity v has a prescribed value v_{wall} on the west boundary, the values in ghost cell nodes must be adjusted such that the average on the boundary is v_{wall} [13]:

$$v_{1,j} = 2v_{wall} - v_{2,j}. \tag{5.2}$$

Where, in this context, the equal sign is an explicit assignment. The prescribed value for the velocity v_{wall} in both directions in our simulations was always taken equal to 0.

For the pressure field we adopted the Neumann condition at the boundaries.

5.2 Surface tension treatment

A good modelling of the surface tension, that acts localized along the interface of the droplet, lies on the calculation accuracy of the normal vectors to the interface calculation. The surface tension force is embedded in the Navier-Stokes equations (3.64) and the localization of the source term is enforced by the delta function (according with the CSF model). The source term is not localized exactly on the interface, but also within a small band with finite thickness around the interface. In fact a smeared-out delta function instead of an exact function is adopted to deal with the numerical problems. The surface tension is calculated as in the eq. (3.64). Instead of using the smeared-out delta function (2.21) we took advantage from the equation (2.18) and we directly calculated the gradient of H_ε using a forward difference scheme (3.2) to avoid the numerical oscillations that are typically encountered from the discretization of the delta function. The discretization of the model is explained in the section 3.4.2.

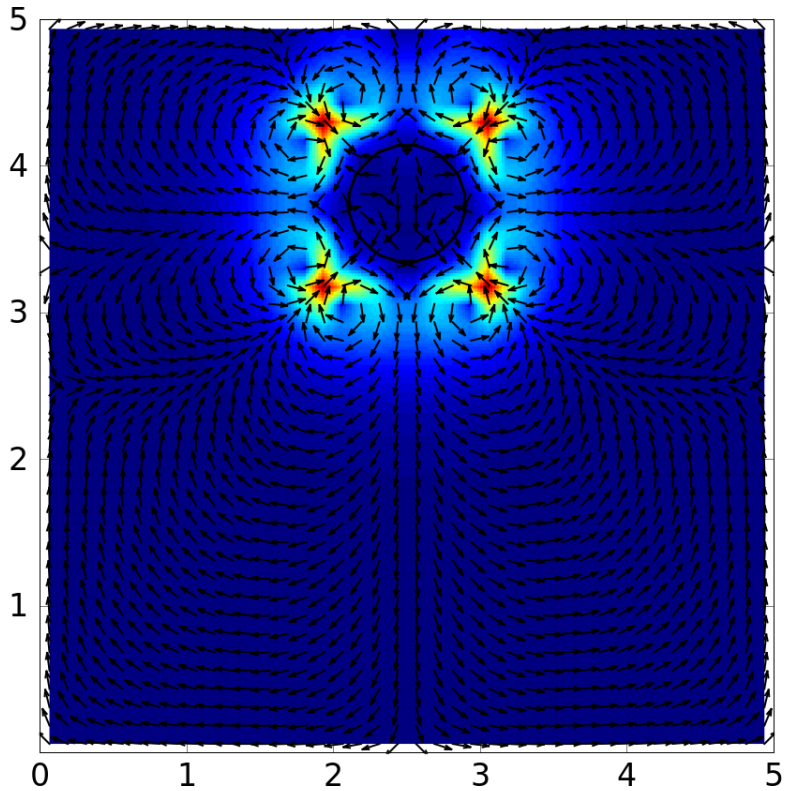


Figure 5.2: *Spurious currents around the interface of the droplet. The turbulence vortexes around the interface are a side effect of the numerical resolution of the surface tension explicit term in the Navier-Stokes equations.*

Since the normals are derived from the shape of the Level Set function, it is crucial to ensure that its transport is made in the right fashion along the interface and within a large enough band around it. For the original method, the *source term method* and the *re-distancing method* the discretization is made accordingly with the subsection 3.4.2. The *conservative method*, due to the fact it uses a function which is not a signed distance function, does not provide good results if we implement the discretization presented above. The Figure 5.2 shows what happens after few iterations. The coloured function in the background is the magnitude of the velocity field, while the arrows represent the direction of the velocity. Near the interface, the adopted numerical approach for the resolution of the surface tension does not prevent the generation of spurious currents. These currents denote a non-physical behaviour of the droplet during its falling. In order to avoid them it is necessary to rewrite the surface tension discretization, in particular paying attention to the density term treatment. According to the subsection 3.4.2 we modelled the numerical scheme, and then we recalculated the density around the interface as

$$\rho_\varepsilon = \frac{\max_\Omega(\rho_\varepsilon) + \min_\Omega(\rho_\varepsilon)}{2}. \tag{5.3}$$

With this modification in the code we damped efficiently the spurious oscillations.

5.3 Results

The performed experiments have shown good results in the simulation of the droplet. The Figure 5.3 illustrates frames of the droplet interface that evolves under the effect of the gravity force and the surface tension force in the air. The shape keeps a round profile until it reaches the ground; that is what we expected since the droplet is small ($d_0 = 1.250\,\text{mm}$) and the acting tension force is strong enough to counteract the distortion of the droplet.

For the cases that advected the signed distance function, the mass M is calculated as the summation over the entire domain of the smeared-out Heaviside function values, then multiplied for the density ratio. For the *conservative method*, we calculated the mass as the summation over the domain of the phase-field function. In the Figure 5.4 the droplet mass for the traditional method and the *conservative method* is drawn with respect to the time. The original method follows the trend to slightly lose mass during the time, since the reinitialization step does not keep the interface in the exact position where it is supposed to be. A similar behaviour is observed for *re-distancing method* and the *source term method*, even though in both cases there is a slightly improvement in the mass

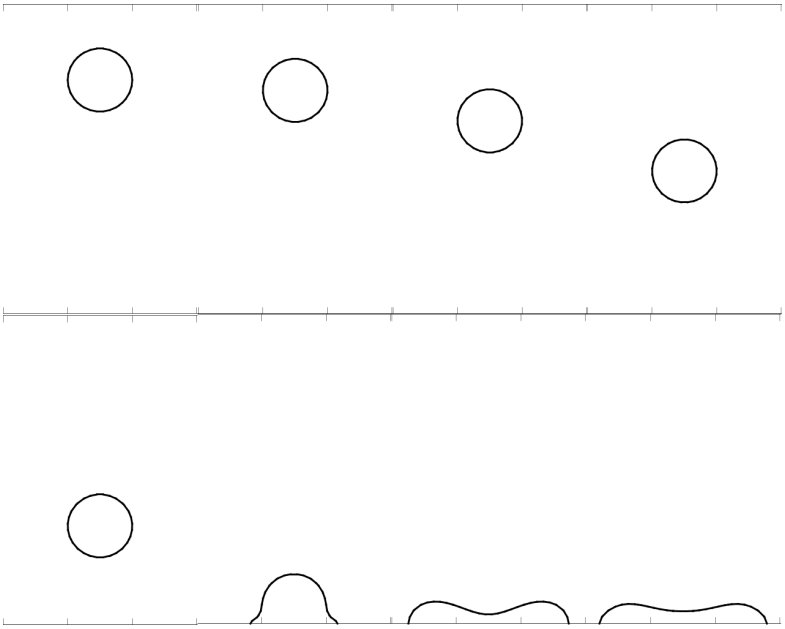


Figure 5.3: *Falling droplet from $t = 0$ to $t = 3$ time units.*

conservation. The *conservative method* is not affected by the problem of loosing mass with respect to the time. The abrupt leap of the curve at about $t = 2.5$ is due to the impact of the droplet with the ground wall.

In the Figure 5.5 the velocity field of the falling droplet for the conservative case is represented. The dominion is wide enough to not affect the velocity field considerably. The velocity field still shows small oscillations of the velocity inside the droplet (look at the second and third frames), but the spurious currents were substantially damped compared with the initial simulations.

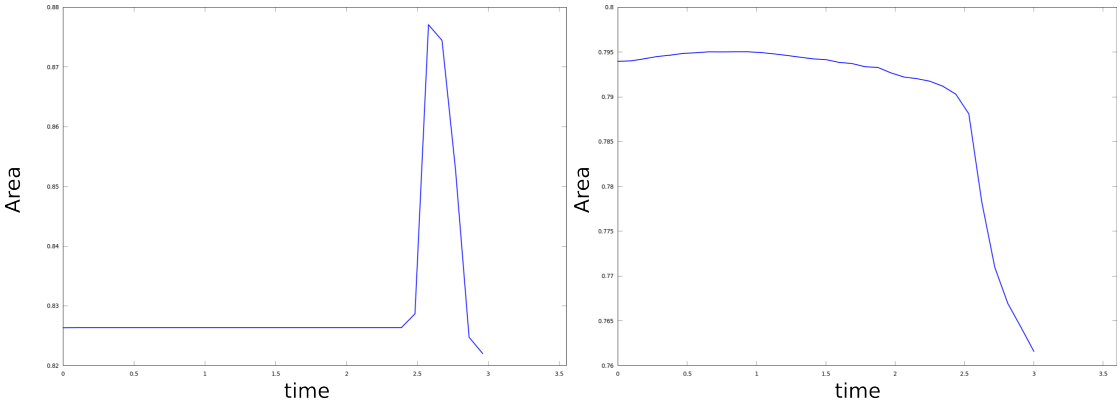


Figure 5.4: *Droplet mass with respect to the time. On the left, the conservative method, on the right, the original method.*

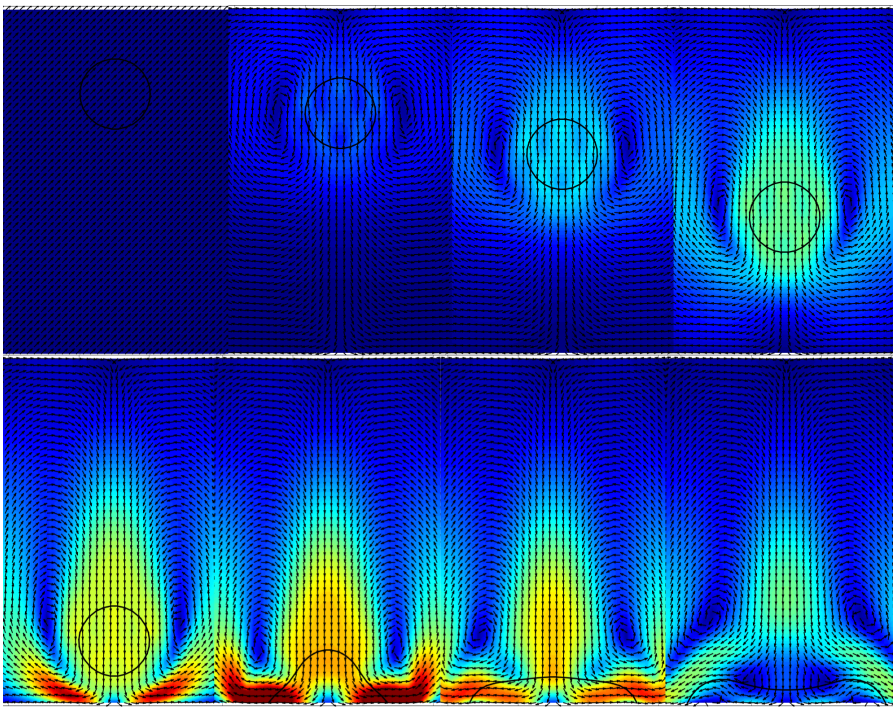


Figure 5.5: *Evolution of the falling droplet until t reaches $t = 3$. The interface is captured by the conservative method*

CHAPTER 6

Conclusions

In this chapter we summarized the obtained results and final considerations, then possible evolutions of the presented work are shown.

6.1 Summary

In this thesis a numerical method for a non-turbulent 2-D two-phase flow has been implemented. The interface was transported via a partial differential equation and a reinitialization step was introduced in order to maintain the total mass during the evolution in time. The Navier-Stokes equations were resolved by a DNS approach, the projection method for decoupling the velocity from the pressure term was adopted. The treating of the multi-phase was accomplished using the CSF method. The implementation of the transport techniques was made from scratch using the the environment provided by Octave 3.8.1.

Thus, three more models to improve the mass conservation were implemented. The first proposed method was the *re-distancing method*. We modified the reinitialization step introducing a constrain in the equation. Such a modification reduces the shrinking behaviour that affects the interface during its advancing in time. Secondly, we proposed the *source term method*: a source term was embedded in the advection equation of the Level Set function that forces it to keep the properties of a distance function even outside the interface, clearing the algorithm from the reinitialization step. The third method is a *conservative method* that replaces the distance function with a hyperbolic tangent profile that is advected and reinitialized using conservative equations.

The verification step and the numerical assessment have been made testing all the four methods with three common benchmarks: the revolution of the circular disk, the revolution of the Zalesak's disk (both via a rotational field) and the evolution after a complete period T of a circular disk under the effect of a vortex field.

The tests have shown that the numerical implementation of the methods is consistent with the analytical equations, for all the four cases. The evidence was procured by the fact that increasing the grid resolution, the numerical solution approximates better the analytic solution.

In those benchmarks the *conservative method* stood out for its ability to keep the mass unaltered through the time-steps. The accuracy of the method is independent from the grid resolution. Nonetheless, it struggles in the reconstruction of the expected interface shape. We observed this behaviour where the grid resolution is too small to resolve adequately the interface thickness. The *source term method* worked good until there were not critical alterations of the shape to deal with, then failed both in accuracy and efficiency for complex cases, i.e. the vortex test. We suggest the *re-distancing method* as a smart fix for the reinitialization step. In fact it decreases the loss in mass during the time without introducing disadvantages: it is easy to implement, does not substantially increase the computational time and keeps the method stable.

The coupling of the proposed method with the Navier-Stokes equations has been accomplished. As a benchmark we proposed the falling droplet test, where a droplet of water falls under the effect of the gravity force. The tests showed that the methods simulate with a good accuracy the evolution of the droplet. Situations such as the droplet breaking against the wall and fluid merges were managed with no difficulties for most the simulations. The *source term method* showed some problems in the equations resolution. Due to the need to adopt a too much strict time-step to ensure convergence we do not suggest such a method to the reader, the gain is too small compared with the drawbacks.

6.2 Future works

The present thesis is the first part of a project addressed to enhance the accuracy of a numerical algorithm for the direct simulation of dispersed liquid droplets in isotropic turbulence. The straightforward steps that follow what presented here are:

- validation process for the coupled Navier-Stokes Level Set solver;
- implementation of the new models in Fortran 90 and parallelization in open MPI;

- extension to the 3-D case.

The provided model could be improved with the following changes:

- substituting the CSF method with the Ghost Fluid Method;
- implementing the WENO 5 scheme for the conservative method;
- introduction of the narrow band to resolve faster the interface.

The ghost fluid method allows the model to resolve the surface tension in a sharp fashion. Thus, it should introduce additional global accuracy.

APPENDIX

Algorithms

Here we reported a part of the code developed to show how we implemented the evolution of the interfaces.

Resolution of the transport equation, TVD upwind method with superbee limiter.

```
function [phi] = lsm_phasefield_advection_superbee(phi,ui,vi,dt,dx,dy)

  nx     = size(phi,1);
  ny     = size(phi,2);
  sx     = zeros(nx,ny-2);
  sy     = zeros(nx-2,ny);
  flux_x = zeros(nx-1,ny-2);
  flux_y = zeros(nx-2,ny-1);

  phin = phi;

  for n = 1:2

    rght_phix = (phi(3:end,2:end-1) - phi(2:end-1,2:end-1))/dx;
    left_phix = (phi(2:end-1,2:end-1) - phi(1:end-2,2:end-1))/dx;
    rght_phiy = (phi(2:end-1,3:end) - phi(2:end-1,2:end-1))/dy;
    left_phiy = (phi(2:end-1,2:end-1) - phi(2:end-1,1:end-2))/dy;

    sx(2:end-1,:) = limiter(rght_phix,left_phix);
    sy(:,2:end-1) = limiter(rght_phiy,left_phiy);
```

```
    % Overwriting variables: same name, different things!! Beware
    left_phix = phi(1:end-1,2:end-1) + 0.5*sx(1:end-1,:)*dx;
    left_phiy = phi(2:end-1,1:end-1) + 0.5*sy(:,1:end-1)*dy;
    rght_phix = phi(2:end,2:end-1)   - 0.5*sx(2:end,:)*dx;
    rght_phiy = phi(2:end-1,2:end)   - 0.5*sy(:,2:end)*dy;

    flux_x = max(ui(1:end-1,2:end-1),0).*left_phix +...
             min(ui(1:end-1,2:end-1),0).*rght_phix;
    flux_y = max(vi(2:end-1,1:end-1),0).*left_phiy +...
             min(vi(2:end-1,1:end-1),0).*rght_phiy;

    phi(2:end-1,2:end-1) +=...
                    - ((flux_x(2:end,:) - flux_x(1:end-1,:))/dx + ...
                       (flux_y(:,2:end) - flux_y(:,1:end-1))/dy)*dt;
    phi = bc(phi);

  endfor

  phi = 0.5*(phi + phin);

endfunction

function [phi] = bc(phi)
  phi(1,:) = phi(2,:);
  phi(end,:) = phi(end-1,:);
  phi(:,1) = phi(:,2);
  phi(:,end) = phi(:,end-1);
endfunction

function [s] = limiter(x,y)
  signx = x./(sqrt(x.^2) + eps);
  ax    = abs(x);
  ay    = abs(y);

 s = signx.*max(ax,ay).*((ay >= ax*0.5 & ay <= 2.0*ax) & x.*y > 0) +...
    2.0*signx.*min(ax,ay).*((ay <= ax*0.5 | ay >= 2.0*ax) & x.*y > 0);
endfunction
```

Intermediate step for the conservative method.

```
function [phi, iter] = intermediate_step(phi,dx,dy,epsilon,d,X,Y)

  dtau   = 0.5*dx^(1 + d);
  nx     = size(phi,1);
  ny     = size(phi,2);
  f      = zeros(nx,ny);
  g      = zeros(nx,ny);
  phix   = zeros(nx-2,ny-2);
  phiy   = zeros(nx-2,ny-2);
  n_x    = zeros(nx-2,ny-2);
  n_y    = zeros(nx-2,ny-2);
  flux_f = zeros(nx-1,ny-2);
  flux_g = zeros(nx-2,ny-1);

  phix = (phi(3:end,2:end-1)-phi(1:end-2,2:end-1))/2/dx;
  phiy = (phi(2:end-1,3:end)-phi(2:end-1,1:end-2))/2/dy;

  n_x = phix./(sqrt(phix.^2 + phiy.^2) + eps);
  n_y = phiy./(sqrt(phix.^2 + phiy.^2) + eps);

  for iter = 1:10

    phin = phi;

    % Runge-Kutta 2
    for n = 1:2

      f(2:end-1,2:end-1) = phi(2:end-1,2:end-1).*...
                     (1 - phi(2:end-1,2:end-1)).*n_x;
      g(2:end-1,2:end-1) = phi(2:end-1,2:end-1).*...
                     (1 - phi(2:end-1,2:end-1)).*n_y;

      f = bc(f);
      g = bc(g);

      flux_f = 0.5*(f(1:end-1,2:end-1) + f(2:end,2:end-1)) - ...
             epsilon*(phi(2:end,2:end-1) - phi(1:end-1,2:end-1))/dx;
      flux_g = 0.5*(g(2:end-1,1:end-1) + g(2:end-1,2:end)) - ...
```

```
            epsilon*(phi(2:end-1,2:end) - phi(2:end-1,1:end-1))/dy;

    phi(2:end-1,2:end-1) += - ((flux_f(2:end,:) - flux_f(1:end-1,:))/dx +...
                     (flux_g(:,2:end) - flux_g(:,1:end-1))/dy)*dtau;
      phi = bc(phi);

    endfor

    phi = 0.5*(phi + phin);

    mask  = (abs(phin) < 3*dx);
    err   = sum(sum(abs(phi - phin).*mask))/sum(sum(mask));
    if (err < dx**2*dtau)
      break;
    endif

  endfor

endfunction
```

Numerical scheme for the resolution of the source term from the *source term method*.

```
function [out] = rhs2_friedrichs(phi,u,v,dx,dy)

  nx   = size(phi,1);
  ny   = size(phi,2);
  phix = zeros(nx+3,ny-2);
  phiy = zeros(nx-2,ny+3);
  udx  = (u(1:end-2,2:end-1) - u(2:end-1,2:end-1))/dx;
  vdy  = (v(2:end-1,1:end-2) - v(2:end-1,2:end-1))/dy;
  udy  = (u(1:end-2,2:end-1) - u(2:end-1,2:end-1))/dy;
  vdx  = (v(2:end-1,1:end-2) - v(2:end-1,2:end-1))/dx;

  phix(3:end-2,:) = diff(phi(:,2:end-1),1,1)/dx;
  phiy(:,3:end-2) = diff(phi(2:end-1,:),1,2)/dy;

  v1 = phix(1:end-5,:);
  v2 = phix(2:end-4,:);
  v3 = phix(3:end-3,:);
```

```
  v4 = phix(4:end-2,:);
  v5 = phix(5:end-1,:);
  v6 = phix(6:end,:);

  left_phix = weno5(v1,v2,v3,v4,v5);
  rght_phix = weno5(v6,v5,v4,v3,v2);

  v1 = phiy(:,1:end-5);
  v2 = phiy(:,2:end-4);
  v3 = phiy(:,3:end-3);
  v4 = phiy(:,4:end-2);
  v5 = phiy(:,5:end-1);
  v6 = phiy(:,6:end);

  left_phiy = weno5(v1,v2,v3,v4,v5);
  rght_phiy = weno5(v6,v5,v4,v3,v2);

 term1 = udx.*(left_phix + rght_phix)*0.5.*(left_phix + rght_phix)*0.5;
 term2 = vdx.*(left_phix + rght_phix)*0.5.*(left_phiy + rght_phiy)*0.5;
 term3 = udy.*(left_phiy + rght_phiy)*0.5.*(left_phix + rght_phix)*0.5;
 term4 = vdy.*(left_phiy + rght_phiy)*0.5.*(left_phiy + rght_phiy)*0.5;

  H = phi(2:end-1,2:end-1).*(term1 + term2 + term3 + term4);

  H1a = (2*udx.*left_phix + vdx.*left_phiy + udy.*left_phiy);
  H1b = (2*udx.*left_phix + vdx.*rght_phiy + udy.*rght_phiy);
  H1c = (2*udx.*rght_phix + vdx.*rght_phiy + udy.*rght_phiy);
  H1d = (2*udx.*rght_phix + vdx.*left_phiy + udy.*left_phiy);

  H2a = (2*vdy.*left_phiy + vdx.*left_phix + udy.*left_phix);
  H2b = (2*vdy.*rght_phiy + vdx.*left_phix + udy.*left_phix);
  H2c = (2*vdy.*rght_phiy + vdx.*rght_phix + udy.*rght_phix);
  H2d = (2*vdy.*left_phiy + vdx.*rght_phix + udy.*rght_phix);

 a = max(max(abs(H1a),abs(H1b)),max(abs(H1c),abs(H1d))).*phi(2:end-1,2:end-1);
 b = max(max(abs(H2a),abs(H2b)),max(abs(H2c),abs(H2d))).*phi(2:end-1,2:end-1);

 out = H - a.*(rght_phix - left_phix)*0.5 - b.*(rght_phiy - left_phiy)*0.5;

endfunction
```

Surface tension treatment for the *conservative method*.

```
  [K(2:end-1,2:end-1), PHIx, PHIy] = curvature(PHI,dx,dy);
  FU = 0.5*(K(3:end,2:end-1).*PHIx(3:end,2:end-1) +...
            K(2:end-1,2:end-1).*PHIx(2:end-1,2:end-1));
  FU = -qWe*FU*(2/(max(max(RHO)) + min(min(RHO))));
  FV = 0.5*(K(2:end-1,3:end).*PHIy(2:end-1,3:end) +...
            K(2:end-1,2:end-1).*PHIy(2:end-1,2:end-1));
  FV = -qWe*FV*(2/(max(max(RHO)) + min(min(RHO))));
  FV = - qFr + FV;

function [out,phi_x,phi_y] = curvature_olsson(phi,dx,dy)

  nx = size(phi,1);
  ny = size(phi,2);
  phi_x = zeros(nx,ny);
  phi_y = zeros(nx,ny);

  phi_x(2:end-1,:) = 0.5*(phi(3:end,:) - phi(1:end-2,:))/dx;
  phi_y(:,2:end-1) = 0.5*(phi(:,3:end) - phi(:,1:end-2))/dy;

  gradphi = sqrt(phi_x.^2 + phi_y.^2 + eps);

  out = 0.5*(phi_x(3:end,2:end-1)./gradphi(3:end,2:end-1) -...
           phi_x(1:end-2,2:end-1)./gradphi(1:end-2,2:end-1))/dx +...
        0.5*(phi_y(2:end-1,3:end)./gradphi(2:end-1,3:end) -...
             phi_y(2:end-1,1:end-2)./gradphi(2:end-1,1:end-2))/dy;

endfunction
```

List of symbols and acronyms

	Description	Symbol
Greek letters	Phase-field function	ϕ
	Dominion of the system	Ω
	Geometric interface	Γ
	Fictitious time-step	τ
	Mean curvature	κ
	Delta function	δ
	Discrete interval	Δ
	Finite small thickness	ε
	Density	ρ
	Dynamic viscosity	μ
	Limiter function	ψ
	Source coefficient	λ
	Lax-Friedrichs coefficients	α, β
	Density and viscosity ratio	η

	Description	Acronym
Acronyms	Computational Fluid-Dynamics	CFD
	Direct Numerical Simulation	DNS
	Level Set Method	LM
	Volume Of Fluid	VOF
	Front Tracking	FT
	Constrained Interpolation Profile	CIP
	Total Variation Diminishing	TVD
	Essentially Non-Oscillatory scheme	ENO
	Weighted ENO	WENO
	Right-Hand Side	RHS
	Ghost Fluid Method	GFM
	Continuum Surface Force	CSF
	Navier-Stokes	NS

Bibliography

[1] David Adalsteinsson and James A Sethian. "A fast level set method for propagating interfaces". In: *Journal of computational physics* 118.2 (1995), pp. 269–277.

[2] Martino Bardi and Stanley Osher. "The nonconvex multidimensional Riemann problem for Hamilton-Jacobi equations". In: *SIAM Journal on Mathematical Analysis* 22.2 (1991), pp. 344–351.

[3] Guy Barles, H Mete Soner, and Panagiotis E Souganidis. "Front propagation and phase field theory". In: *SIAM Journal on Control and Optimization* 31.2 (1993), pp. 439–469.

[4] John B Bell, Phillip Colella, and Harland M Glaz. "A second-order projection method for the incompressible Navier-Stokes equations". In: *Journal of Computational Physics* 85.2 (1989), pp. 257–283.

[5] JU Brackbill, Douglas B Kothe, and C1 Zemach. "A continuum method for modeling surface tension". In: *Journal of computational physics* 100.2 (1992), pp. 335–354.

[6] Yu-Chung Chang et al. "A level set formulation of Eulerian interface capturing methods for incompressible fluid flows". In: *Journal of computational Physics* 124.2 (1996), pp. 449–464.

[7] Alexandre Joel Chorin. "Numerical solution of the Navier-Stokes equations". In: *Mathematics of computation* 22.104 (1968), pp. 745–762.

[8] Olivier Desjardins, Vincent Moureau, and Heinz Pitsch. "An accurate conservative level set/ghost fluid method for simulating turbulent atomization". In: *Journal of Computational Physics* 227.18 (2008), pp. 8395–8416.

[9] Douglas Enright et al. "A hybrid particle level set method for improved interface capturing". In: *Journal of Computational physics* 183.1 (2002), pp. 83–116.

[10] Ronald P Fedkiw et al. "A non-oscillatory Eulerian approach to interfaces in multimaterial flows (the ghost fluid method)". In: *Journal of computational physics* 152.2 (1999), pp. 457–492.

[11] Marianne M Francois et al. "A balanced-force algorithm for continuous and sharp interfacial surface tension models within a volume tracking framework". In: *Journal of Computational Physics* 213.1 (2006), pp. 141–173.

[12] Jose Gomes and Olivier Faugeras. "Reconciling distance functions and level sets". In: *Biomedical Imaging, 2002. 5th IEEE EMBS International Summer School on*. IEEE. 2002, 15–pp.

[13] Michael Griebel, Thomas Dornseifer, and Tilman Neunhoeffer. *Numerical simulation in fluid dynamics: a practical introduction*. Vol. 3. Siam, 1997.

[14] Amiram Harten. "The artificial compression method for computation of shocks and contact discontinuities. III. Self-adjusting hybrid schemes". In: *Mathematics of Computation* 32.142 (1978), pp. 363–389.

[15] Ami Harten et al. "Uniformly high order accurate essentially non-oscillatory schemes, III". In: *Journal of computational physics* 71.2 (1987), pp. 231–303.

[16] Guang-Shan Jiang and Danping Peng. "Weighted ENO schemes for Hamilton–Jacobi equations". In: *SIAM Journal on Scientific computing* 21.6 (2000), pp. 2126–2143.

[17] Myungjoo Kang, Ronald P Fedkiw, and Xu-Dong Liu. "A boundary condition capturing method for multiphase incompressible flow". In: *Journal of Scientific Computing* 15.3 (2000), pp. 323–360.

[18] Curtis Lee, John Dolbow, and Peter J Mucha. "A narrow-band gradient-augmented level set method for multiphase incompressible flow". In: *Journal of Computational Physics* 273 (2014), pp. 12–37.

[19] Randall J LeVeque. *Finite volume methods for hyperbolic problems*. Vol. 31. Cambridge university press, 2002.

[20] Xu-Dong Liu, Ronald P Fedkiw, and Myungjoo Kang. "A boundary condition capturing method for Poisson's equation on irregular domains". In: *Journal of computational Physics* 160.1 (2000), pp. 151–178.

[21] Takashi Nakamura et al. "Exactly conservative semi-Lagrangian scheme for multi-dimensional hyperbolic equations with directional splitting technique". In: *Journal of Computational Physics* 174.1 (2001), pp. 171–207.

[22] Jean-Christophe Nave, Rodolfo Ruben Rosales, and Benjamin Seibold. "A gradient-augmented level set method with an optimally local, coherent advection scheme". In: *Journal of Computational Physics* 229.10 (2010), pp. 3802–3827.

[23] Elin Olsson and Gunilla Kreiss. "A conservative level set method for two phase flow". In: *Journal of computational physics* 210.1 (2005), pp. 225–246.

[24] Elin Olsson, Gunilla Kreiss, and Sara Zahedi. "A conservative level set method for two phase flow II". In: *Journal of Computational Physics* 225.1 (2007), pp. 785–807.

[25] Stanley Osher and Ronald Fedkiw. *Level set methods and dynamic implicit surfaces*. Vol. 153. Springer Science & Business Media, 2003.

[26] Stanley Osher and Ronald P Fedkiw. "Level set methods: an overview and some recent results". In: *Journal of Computational physics* 169.2 (2001), pp. 463–502.

[27] Stanley Osher and James A Sethian. "Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations". In: *Journal of computational physics* 79.1 (1988), pp. 12–49.

[28] A Ovsyannikov, V Sabel'nikov, and M Gorokhovski. "A new level set equation and its numerical assessments". In: *Proceedings of the Summer Program*. 2012, p. 315.

[29] Danping Peng et al. "A PDE-based fast local level set method". In: *Journal of Computational Physics* 155.2 (1999), pp. 410–438.

[30] Giovanni Russo and Peter Smereka. "A remark on computing distance functions". In: *Journal of Computational Physics* 163.1 (2000), pp. 51–67.

[31] Vladimir Sabelnikov, Andrey Yu Ovsyannikov, and Mikhael Gorokhovski. "Modified level set equation and its numerical assessment". In: *Journal of Computational Physics* 278 (2014), pp. 1–30.

[32] Ruben Scardovelli and Stéphane Zaleski. "Direct numerical simulation of free-surface and interfacial flow". In: *Annual review of fluid mechanics* 31.1 (1999), pp. 567–603.

[33] Chi-Wang Shu and Stanley Osher. "Efficient implementation of essentially non-oscillatory shock-capturing schemes". In: *Journal of Computational Physics* 77.2 (1988), pp. 439–471.

[34] Mark Sussman and Emad Fatemi. "An efficient, interface-preserving level set redistancing algorithm and its application to interfacial incompressible fluid flow". In: *SIAM Journal on scientific computing* 20.4 (1999), pp. 1165–1191.

[35] Mark Sussman and Elbridge Gerry Puckett. "A coupled level set and volume-of-fluid method for computing 3D and axisymmetric incompressible two-phase flows". In: *Journal of Computational Physics* 162.2 (2000), pp. 301–337.

[36] Mark Sussman, Peter Smereka, and Stanley Osher. "A level set approach for computing solutions to incompressible two-phase flow". In: *Journal of Computational physics* 114.1 (1994), pp. 146–159.

[37] Mark Sussman et al. "An improved level set method for incompressible two-phase flows". In: *Computers & Fluids* 27.5 (1998), pp. 663–680.

[38] Hideaki Takewaki and Takashi Yabe. "The cubic-interpolated pseudo particle (CIP) method: application to nonlinear and multi-dimensional hyperbolic equations". In: *Journal of Computational Physics* 70.2 (1987), pp. 355–372.

[39] Grétar Tryggvason, Ruben Scardovelli, and Stéphane Zaleski. *Direct numerical simulations of gas–liquid multiphase flows*. Cambridge University Press, 2011.

[40] Salih Ozen Unverdi and Grétar Tryggvason. "A front-tracking method for viscous, incompressible, multi-fluid flows". In: *Journal of computational physics* 100.1 (1992), pp. 25–37.

[41] Henk Kaarle Versteeg and Weeratunge Malalasekera. *An introduction to computational fluid dynamics: the finite volume method*. Pearson Education, 2007.

[42] TAKASHI Yabe and T Aoki. "A universal solver for hyperbolic equations by cubic-polynomial interpolation I. One-dimensional solver". In: *Computer Physics Communications* 66.2 (1991), pp. 219–232.