POLITECNICO DI TORINO

I Facoltà di Ingegneria
Corso di Laurea in Ingegneria Matematica

Tesi di Laurea Magistrale

Adaptive Mesh Refinement for Multiphase Flow CFD

Relatore:
Claudio Canuto

Candidato:
Matteo Icardi

Supervisori aziendali
Djamel Lakehal
ASCOMP GmbH

Valerio Marra
COMSOL Srl

Maggio 2008

Title

Adaptive Mesh Refinement for Multiphase Flow CFD

Abstract

CMFD (Computational Multi-Fluid Dynamics) is one of the fields where advanced mathematical tools are mostly needed, together with physical analysis, mathematical modeling and computer science, to develop new models and numerical methods to simulate complex real problems. In this work, done in Zurich (CH) at ASCOMP GmbH, we developed and implemented two different methods of mesh refinement for single and multi-phase flows. The first one, called BMR, is a local block-based mesh refinement done with a hierarchy of Cartesian blocks. We developed for this a technique, similar to the multigrid one, to approximate the solution on grids with different resolution together with a mesh generator building automatically refined blocks around complex three dimensional geometries. The second one is an Adaptive Mesh Refinement (AMR) method to capture the interface between fluids with high resolution. This has been implemented using an octree structure and applying special numerical methods to solve Hamilton-Jacobi equations, obtained from the Level Set method. We implemented these two mesh refinement methods in the software TransAT, a multi-phase flow solver using the Immersed Surfaces Technique to treat solid boundaries. This technique gives the possibility to simulate complex geometries using simple structured Cartesian grids. In this thesis we first review different multi-phase flows mathematical models and the most important numerical methods for front tracking, focusing in particular on the Level Set method, then we analyze the theory of mesh refinement and explain the work done. Finally some simulation results are proposed to show the capabilities of our methods.

Titolo

Raffinamento adattativo di griglia per la Fluidodinamica Computazionale Multi-Fase

Sommario

La Fluidodinamica Computazionale Multi-Fase è un campo molto complesso in cui sono necessari strumenti matematici avanzati, insieme all'analisi fisica, alla modellazione matematica e all'informatica, per sviluppare nuovi modelli e metodi numerici e simulare problemi reali. In questa tesi, svolta a Zurigo (CH) presso ASCOMP GmbH, abbiamo sviluppato e implementato due meodi di raffinamento di griglia per fluidi mono-fase e multi-fase. Il primo, chiamato BMR, è un raffinamento locale a blocchi ottenuto con una gerarchia di griglie cartesiane. In questa tecnica abbiamo sviluppato un algoritmo simile al multigrid per approssimare la soluzione su griglie con differenti risoluzioni, e un generatore di griglia in grado di costruire automaticamente blocchi rifiniti attorno a complesse geometrie tri-dimensionali. Il secondo metodo proposto è un raffinamento adattativo di griglia per catturare l'interfaccia tra fluidi con alta risoluzione. Questo è stato implementato usando una struttura octree e applicando appositi metodi numerici per risolvere le equazioni di Hamilton-Jacobi, ottenute con il metodo Level Set. Abbiamo implementato questi due metodi in TransAT, un software per fluidi multi-fase con la tecnica delle superfici immerse per trattare i bordi solidi. Questa tecnica permette di simulare geometrie complesse usando semplici griglie strutturate cartesiane. Dopo aver illustrato i differenti modelli per fluidi multi-fase e i più importanti metodi numerici di tracciatura dell'interfaccia, in particolare il metodo Level Set, analizziamo la teoria del raffinamento di griglia e illustriamo i procedimenti utilizzati. Infine mostriamo i risultati di alcune simulazioni per testare le potenzialità dei metodi sviluppati.

Contents

Chapter 1

Introduction

The numerical simulation of multi-phase flows is relevant to several fields such as environment, geophysics, chemistry, engineering and basic physics. In particular, the study of interface motion is of primary importance in combustion problems, droplet clouds or sprays formation, chemical, oil and nuclear energy industry. Unfortunately in many of these applications it is still difficult to solve the whole system with a computational grid fine enough to capture the smallest scales that usually can have a big influence on the final solution.

For multi-phase flows this problem becomes even more difficult because scale sizes and turbulence phenomena strongly depend on the phase and the interface between the fluids, which is a variety with a lower dimensionality than the space where it is located (e.g. a moving and deforming surface in a three-dimensional domain). Capillary forces are concentrated on it and across it fluid properties, such as density and viscosity, are usually discontinuous. As a result we are dealing with a set of partial differential equations with discontinuous coefficients and with a few source terms, such as the surface tension force, which are singular (i.e. zero everywhere except on the interface) and that can be correctly expressed only in terms of distributions, or delta functions. Furthermore, everyday experience points out that interfaces are subject to topology changes, as in droplet coalescence or jet atomization, that are rather difficult to reproduce if the flow is assumed to be incompressible and therefore there is no singularity in the velocity field.

These difficulties are dealt with in a variety of ways, depending on the physical

model, the numerical methods and the interface representation. For single-phase flow we can use averaged equations (together with turbulence models) to model the smallest scales or we can refine the grid where is needed in order to resolve them. Both solutions are still important subjects of research and we will focus on the second one analyzing the various approach of mesh refinement. In many important applications however, only these two approaches together have some chances to obtain a good result. For multi-phase flows, where we have also to take into account the interactions between the different phases, another important possibility and research subject is the explicit tracking of the interface between the fluids instead of modeling these interactions. This approach, compared with other ones where phases are mixed, has many advantages because we can distinguish exactly the phases (e.g. we can use for each of them specific turbulence models), but needs a big computational effort and a very fine grid near the interface. This is the reason why Adaptive Mesh Refinement is very important for multi-phase flows, even more than for single phase flows.

Another important issue in flow simulations is the choice of the computational grid. In fact it must be able to represent both solid and fluid boundaries. Complex solid boundaries are usually represented in Computational Fluid Dynamics (CFD) using boundary-following structured curvilinear or unstructured grids. While boundary conditions can be easily and accurately applied on such grids, grid generation can be a difficult and time consuming process. In recent years, especially for multi-phase flows, Cartesian grids and "immersed boundary" techniques have known a regain of interest because they greatly simplify the grid generation process.

This thesis is structured to give a general introduction of the matter and to explain the work done during a 6-months internship at *ASCOMP GmbH* in Zurich (CH). We will focus especially on the mathematical tools used and we will briefly describe the implemented codes. In Chapter 2 we present the main concepts about the different mathematical models of multi-phase flows and front tracking methods while in Chapter 3 we analyzed the mathematical properties of the Hamilton-Jacobi equations that arise in the Level Set method. As we mentioned before, an important issue in multi-phase flows is mesh refinement. In chapter 4 we study different approaches for grid generation and refinement with special care

to their applications to interface tracking. In chapter 5 we present the software *TransAT* used for the simulation and the Immersed Surface Technique. Then in the last chapters we present the work done and relevant results obtained during the internship period started in October 2007. Chapter 6 and 7 deal with the development of *BMR (Block-based Mesh Refinement)* and *AMR (Adaptive Mesh Refinement)* for TransAT while in Chapter 8 we present some results to increase the accuracy of interface curvature calculation, which is very important for multi-phase flow with surface tension. In Chapter 9 we present some simulations of single-phase and multi-phase flows using TransAT and COMSOL Multiphysics 3.4 and finally we discuss all the results and the future developments in Chapter 10.

Chapter 2

Multi-phase flows

In this chapter we summarize the main characteristics of multi-phase flows, and their relevant mathematical models and numerical methods. We focus in particular on the Level Set method.

2.1 Multi-phase flows

When we model and simulate multi-phase flows we need to take care of the characteristics that differ from the usual single-phase CFD. The most important ones are

- Interactions between the different phases along the interface or *jump conditions*. These can be represented, depending on the mathematical model adopted, explicitly or with some closure laws. Specific numerical methods must be developed to treat the interface as a real discontinuity.
- *Phase changes* from liquid to vapor and vice versa. This means that we have to solve also an equation for the temperature (or internal energy) that cannot be decoupled from the other ones. In some cases these changes can involve the solid phase also.
- The *contact angle* which is the angle at which a liquid/vapor interface meets the solid surface (see Figure 2.1). The contact angle is specific for any given system and is determined by the interactions across the three interfaces. It

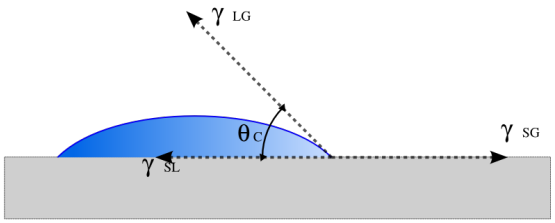

Figure 2.1: *The contact angle θ at a wetted wall.*

can be a crucial parameters when the surface tension is high (see simulation 9.2.3). Sometimes it is supposed to be constant or it can be calculated using the Young equation

$$\gamma_{\mathrm{SG}} - \gamma_{\mathrm{SL}} - \gamma_{LG} \cos\theta = 0 \tag{2.1}$$

where θ is the contact angle,$\gamma_{LG} = \sigma$ is the surface tension coefficient and γ_{SG}, γ_{SL} represent the interfacial energy (respectively solid-vapor and solid-liquid).

2.2 Mathematical models

There are many mathematical models for multi-phase flows ([2]). Most of them are based on the continuous approach meaning that the discontinuity between the phases are treated using averaged equations. Starting from the continuity and momentum equation of each phase, we can obtain the averaged equations multiplying them by phase indicator functions

$$\chi_k(\mathbf{x}) = \begin{cases} 1 \text{ if } \mathbf{x} \in \text{phase } k \\ 0 \text{ if } \mathbf{x} \notin \text{phase } k \end{cases} \tag{2.2}$$

and applying an averaging procedure. In literature several types of averaging are used including time averaging, volume averaging and ensemble averaging. All the averaging procedures yields averaged equations which formally have the same structure. At the end we obtain 2 equations for each phase.

With this approach we can use a continuous formulation of the equations but we lose information about the interaction between the phases that we have to

introduce in some way specifying additional closure laws. Depending on these additional "a priori" information, we can have different model:

- **Homogeneous model**, where we assume that phases move with the same velocity.
- **Diffusion model**, if we assume a constitutive algebraic relation for the relative velocity between fluids.
- **Drift-flux model**, in which we use a volumetric flux density $\mathbf{j}$ of phases mixture and specify a closure law for the so called *drift flux* velocity $\mathbf{v}_1 - \mathbf{j}$ of disperse phase 1.
- **Two-fluid model**, where are given closure laws for momentum transfer across the interface.

Other models have a Lagrangian formulation for the disperse phase (e.g. small drops in air or small bubbles in water) but they also need closure laws or additional balance equations or statistics for the size of the dispersed particles.

The alternative to the continuous formulation is to explicitly take into account the interfaces between the phases. To do this we introduce and focus on a different type of model described in the following section.

2.2.1 Single-fluid model

We consider in the following two incompressible Newtonian immiscible fluids (according to [5]). The whole flow fills a domain Ω. This domain may be decomposed into two sub-domains, each of them filled with one of the two phases. Each fluid component obeys the Navier-Stokes in its sub-domain, with proper boundary conditions at the interface, which for our problems are the continuity of the velocity field at the interface and a jump condition in the total stress tensor components that takes into account the discontinuity of the viscosity and the pressure across the interface together with a capillary force concentrated on the interface itself. The first boundary condition is a requirement that a point on the interface is also a boundary point for both phases, the second one is the balance of the singular

forces at the interface and it is required for its integrity. In the whole-domain formulation of the Navier-Stokes equations these two sets of equations are combined together and extended to the whole domain Ω. The velocity is still continuous across each sub-domain and satisfies the incompressibility condition

$$\nabla \cdot \mathbf{u} = 0 \tag{2.3}$$

where $\mathbf{u}$ is the velocity field.

The momentum balance equation includes now the surface tension term, i.e. the capillary force, which is no longer a boundary conditions across the two sub-domains and it is given by the following Navier-Stokes equation

$$\rho(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u}) = -\nabla p + \nabla \cdot (2\mu \mathbf{D}) + \sigma \kappa \delta_S \mathbf{n} + \mathbf{f} \tag{2.4}$$

where $\mathbf{f}$ represents any external *body* force, for example gravity, p is the hydrostatic pressure and $\mathbf{D}$ the rate-of-strain tensor

$$D_{ij} = \frac{1}{2}\left(\frac{\partial u_j}{\partial x_i} + \frac{\partial u_i}{\partial x_j}\right) \tag{2.5}$$

In Equation (2.4), the term $\sigma \kappa \delta_S \mathbf{n}$ represents the capillary force, here modeled with a constant surface tension coefficient σ, $\mathbf{n}$ is the unit vector normal to the interface, $\kappa = \nabla_S \cdot \mathbf{n}$ is the local interface curvature (various derivations are given in Section 2.2.2), where the divergence in this expression is restricted to directions tangent to the surface. A definition of the distribution δ_S is given in Section 2.2.2, in terms of the characteristic function H, where $H = 1$ in reference phase and $H = 0$ in the secondary phase. Thus, is basically a three-dimensional Heaviside function, constant in each phase with its value equal either to 1 or 0, but discontinuous across the interface.

The sub-domain of Ω occupied by each of the two fluids is thus completely specified by the function H, but also its time evolution. As a matter of fact the two fluids are immiscible, therefore each single fluid parcel does not change its identity in time and the function H can be considered as passive scalar advected by the flow, therefore its convective derivative is zero, as expressed by Equation (2.32) provided in Section 2.2.2. Note that in (2.4) physical properties may vary

in space because of H and in general may be discontinuous as well across the interface. For instance, the density ρ is written

$$\rho = \rho_1 H + \rho_2(1 - H) \tag{2.6}$$

where ρ_1 and ρ_2 are assumed to be the two constant density values of phase 1 and phase 2, respectively.

A discrete version of H is the scalar field C, which is used in Volume-of-Fluid (VOF) methods for two-phase flow simulations on a computational grid. C defines the fraction of each computational cell which is occupied by the reference phase, therefore we have $0 < C < 1$ in cells cut by the interface otherwise away from it $C = 1$ when the cell is filled with reference phase, $C = 0$ when the cell is occupied by the secondary phase or it is empty for a free-surface flow.

The whole domain formulation (2.4) does not require jump conditions at the interface as it is also called *single-fluid formulation*.

Conservation law form

We may also rewrite Equation (2.4) as a conservation law for the momentum $\rho\mathbf{u}$. Moreover, capillary effects may be represented by the *capillarity pressure tensor* $\mathbf{T}$ which is tangent to the interface

$$\mathbf{T} = -\sigma\left(\mathbf{1} - \mathbf{n} \otimes \mathbf{n}\right) H \tag{2.7}$$

where $\mathbf{1}$ is the Kronecker tensor δ_{ij}. One may then show that the capillary force can be written

$$\sigma\kappa\delta_S\mathbf{n} = -\nabla \cdot \mathbf{T} \tag{2.8}$$

and obtain an equation equivalent to (2.4) but in conservative form

$$\frac{\partial \rho\mathbf{u}}{\partial t} = -\nabla\left(p\mathbf{1} + \rho\mathbf{\Pi} + \mathbf{T} - 2\mu\mathbf{D}\right) + \mathbf{f} \tag{2.9}$$

where $\mathbf{\Pi} = \mathbf{u} \otimes \mathbf{u}$ is the nonlinear advection tensor.

Away from the interface $\mathbf{T}$ vanishes and the Equation (2.9) is the usual Navier-Stokes equation for a single fluid component.

Dimensionless form

To define dimensionless variables it is necessary to choose a reference length $\tilde{l}$ and velocity $\tilde{v}$. The choice depends on the typical problem under investigation. Once these values have been selected the following dimensionless variable may be defined

$$\begin{array}{lll} \mathbf{u}' = \frac{\mathbf{u}}{\tilde{v}} & p' = \frac{(p-p_0)}{\rho\,\tilde{v}^2} & t' = \frac{t\,\tilde{v}}{\tilde{l}} \\ x' = \frac{x}{\tilde{l}} & y' = \frac{y}{\tilde{l}} & \\ \nabla' = \tilde{l}\,\nabla & & \\ \frac{\partial}{\partial t'} = \frac{\tilde{l}}{\tilde{v}}\frac{\partial}{\partial t} & & \end{array} \tag{2.10}$$

where p_0 is a given reference pressure value.

A dimensionless form of Equation (2.4), is readily available, where we have omitted both the capillary and external volume forces

$$\frac{\partial \mathbf{u}'}{\partial t} = -\left(\mathbf{u}' \cdot \nabla'\right)\mathbf{u}' - \nabla' p' + \frac{1}{Re}\nabla' \cdot 2\mathbf{D}' \tag{2.11}$$

where μ and ρ have been considered to be constant, $\mathbf{D}' = \frac{\tilde{l}}{\tilde{v}}\mathbf{D}$ and $Re = \frac{\tilde{l}\,\tilde{v}\,\rho}{\mu}$ is the *Reynolds number*. The flow regime is described from a qualitative point of view by the dimensionless number Re: if $Re \ll 1$ we have the *Stokes flow* where the viscous term $\nabla' \cdot 2\mathbf{D}'$ is prevailing with respect to the convection one $-\left(\mathbf{u}' \cdot \nabla'\right)\mathbf{u}'$. Vice versa for high values of the Reynolds number the viscous term is negligible and the flow regime is said to be *turbulent*. If the capillary and the gravitational force are held in place, two new adimensional numbers are defined, the *Weber number* and the *Froud number*, respectively. The former represents the ratio of inertia forces to surface tension and is defined as $We = \frac{\rho\tilde{l}\,\tilde{v}^2}{\sigma}$, the latter $Fr = \frac{\tilde{v}^2}{g\,\tilde{l}}$ is the ratio of inertia and gravity forces.

2.2.2 Interfaces

In three-dimensional (3D) space the interface separating two fluid phases is a smooth surface S and in two-dimensional space (2D) is a line C. The study of two-dimensional and three-dimensional interfaces is very different but in order to keep the discussion simple in this chapter we will illustrate the representation of two-dimensional interfaces.

The simplest way to define a line in two-dimensional space is by using an equation of the form

$$y = h(x,t) \tag{2.12}$$

In this case the normal vector is

$$\mathbf{n} = \frac{1}{\sqrt{1+\left(\frac{\partial h}{\partial x}\right)^2}} \begin{bmatrix} -\frac{\partial h}{\partial x} \\ 1 \end{bmatrix} \tag{2.13}$$

and the curvature is

$$\kappa = \frac{1}{\sqrt{\left[1+\left(\frac{\partial h}{\partial x}\right)^2\right]^3}} \frac{\partial^2 h}{\partial x^2} \tag{2.14}$$

In a different coordinate system a line may be described by a vector function $\mathbf{x}(s)$ of a real parameter s that conserves length. The real parameter s is called *curvilinear abscissa* and it is chosen such that the tangent $\mathbf{t}$ is the unit vector (i.e. $|d\mathbf{x}(s)/ds| = 1$). The normal to the interface is defined as follows. Let us consider a drop of phase "1" surrounded by phase "2", as shown in Fig. 2.2. The curvilinear coordinate s has been chosen with counterclockwise orientation. The normal vector is defined by the unit vector orthogonal to $\mathbf{t}$ and is directed towards the outside of the drop (i.e. the unit normal points always from the "reference" phase 1 to phase 2). The relation between the unit normal and tangent is as follows

$$\frac{d\mathbf{t}}{ds} = -\kappa\mathbf{n} \tag{2.15}$$

where κ is the curvature having magnitude $|\kappa| = |d\mathbf{t}/ds|$. The sign is positive if the *inside* (phase 1) region is convex and negative otherwise (it depends on the direction chosen for the unit normal). Another definition of *curvature* is given by $|\kappa| = 1/R$ where R, the radius of curvature, is the radius of the circle that best approximates the curve.

Another way to define the interface is through a level-set function ϕ. Let $\phi(x,y)$ be any smooth function such that the interface is located on the level set

$$\phi(x,y) = 0. \tag{2.16}$$

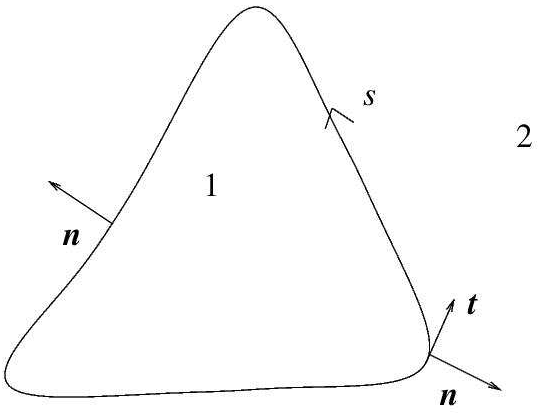

Figure 2.2: *2D interface notations. Figure taken from [5].*

The level-set definition provides a simple formula for the normal vector

$$\mathbf{n} = \frac{\nabla \phi}{|\nabla \phi|} \tag{2.17}$$

and for the curvature

$$\kappa = \nabla \cdot \mathbf{n} \tag{2.18}$$

In order to prove (2.18), we notice that the divergence of a vector $\mathbf{u}$ is

$$\nabla \cdot \mathbf{u} = \mathbf{t} \cdot \frac{\partial}{\partial s}\mathbf{u} + \mathbf{n} \cdot \frac{\partial}{\partial n}\mathbf{u}. \tag{2.19}$$

Also we notice that, since $\mathbf{n}$ is of constant length, it is orthogonal to its derivative, namely $\mathbf{n} \cdot \frac{\partial}{\partial n}\mathbf{n} = 0$. Applying (2.19) to $\mathbf{n}$ we get

$$\nabla \cdot \mathbf{n} = \mathbf{t} \cdot \frac{\partial}{\partial s}\mathbf{n} \tag{2.20}$$

which recovers (2.18).

Another useful tool for interface representation is the characteristic Heaviside function H such that $H = 1$ in phase 1 and $H = 0$ in phase 2. In this case the interface is represented by the discontinuous points of the Heaviside function H. We remark that H can be expressed in terms of an integral over the product of δ-functions

$$H(x, y, t) = \int_{A(t)} \delta(x - x')\delta(y - y')\, dA' \tag{2.21}$$

For instance an ellipse interface can be represented in equation form

$$y = \pm b\sqrt{1 - \frac{x^2}{a^2}} \tag{2.22}$$

in parametrization form

$$\begin{cases} x = a\cos(u) \\ y = b\sin(u) \end{cases} \tag{2.23}$$

in level set function form

$$\phi = \frac{x^2}{a^2} + \frac{y^2}{b^2} - 1 \tag{2.24}$$

and by using the Heaviside function

$$H = H_1(\frac{x^2}{a^2} + \frac{y^2}{b^2} - 1) \tag{2.25}$$

where a, b are the ellipse semi-axes and H_1 is the one-dimensional Heaviside step function, respectively.

It is possible to define a distribution δ_S that is concentrated at the interface as a Dirac δ-function is concentrated at a point. The surface distribution δ_S is conveniently related to the gradient of the characteristic function H. The gradient of (2.21) with respect to the variables (x, y) can be written as

$$\nabla H = \int_A \nabla[\delta(x-x')\delta(y-y')]\, dA' \tag{2.26}$$

Since δ-functions are anti-symmetric with respect to the primed and un-primed variables, the gradient with respect to the variables (x, y) can be replaced by the gradient with respect to the variables (x', y') and it becomes

$$\nabla H = -\int_A \nabla'[\delta(x-x')\delta(y-y')]\, dA' \tag{2.27}$$

where the prime on the gradient symbol denotes the gradient with respect to the variables (x', y').

The integral (2.27) can be transformed into a line integral by the divergence theorem as

$$\nabla H = -\int_S \delta(x-x')\delta(y-y')\mathbf{n}\, dS' \tag{2.28}$$

The circle on the integral is dropped because the contribution of the integral over the closed domain is zero in the most of the contour and the integration can be performed only over the non-vanishing part. This may be also denoted by

$$\nabla H = -\delta_S \mathbf{n} \tag{2.29}$$

We can obtain this same result using the theory of distribution. In fact $\forall \psi \in \mathcal{D}(\Omega)$ we can write

$$\langle \nabla H, \psi \rangle \doteq \int_\Omega H \nabla \psi = -\int_\Omega \nabla H \psi = -\int_{\Omega_1} \nabla H \psi = -\int_S \psi \mathbf{n} = \langle -\delta_S \mathbf{n}, \psi \rangle \tag{2.30}$$

The evolution of the interface S is defined by the specified velocity V on each point of S. This may be the fluid velocity itself, or a result of other effects, such as evaporation or condensation. In order to describe the motion of these points one may also introduce a level-set function ϕ (2.16) and extend the velocity field to the entire space. The idea that a particle on the interface follows the velocity field may be expressed by

$$\frac{D\phi}{Dt} = 0 \tag{2.31}$$

where $\frac{D}{Dt} = \frac{\partial}{\partial t} + \mathbf{u} \cdot \nabla = \frac{\partial}{\partial t} + u\frac{\partial}{\partial x} + v\frac{\partial}{\partial y} + w\frac{\partial}{\partial z}$ and $\mathbf{u}$ are the total derivative and the velocity field, respectively.

This description also hold for the characteristic function H in (2.25) although now the derivatives must be understood in a weak sense

$$\frac{DH}{Dt} = 0 \tag{2.32}$$

2.3 Interface resolving methods

The interface resolving simulation, or direct numerical simulation of it, aims to represent explicitly the deformation and topological evolution of the phase-interface separating the two fluids. We can use the single fluid models and we don't need any "a priori" knowledge of the shape of the bubbles or drops. Physically, the deformation of the phase-interface is intimately connected with the surface tension force. Therefore, any method for direct numerical simulation of gas-liquid or liquid-liquid flow relies on a reliable representation of the surface tension force.

The main difficulty of interface resolving simulations is the moving interface whose shape is part of the solution. Therefore physical quantities such as density, viscosity and pressure are discontinuous across it. Mathematically, the interface is treated as a surface of discontinuity. Loosely spoken it is "sharp". To keep the interface numerically sharp, i.e. to avoid any artificial smearing of the interface during the computation, special numerical methods have been developed. Nowadays mainly three methods are used. The Volume-Of-Fluid method (VOF) is the oldest one while the Level-Set method and the Front-Tracking method have been developed in the last decade. All these methods are formulated for incompressible fluids only and solve only one single momentum equation coupled with an additional equation for the interface.

In the following Subsections 2.3.1 and 2.3.2 some details about VOF and Front Tracking techniques are given, respectively, while in Section 2.4 the Level Set method is well described because of its importance for this work.

2.3.1 VOF

The Volume-of-Fluid method appear the first time in 1981 ([25]). The basic idea is the definition of a scalar quantity C which represents the fraction of the volume of a mesh cell occupied by one phase, say the liquid (phase 1). It is the discrete form of the function H. Thus, for $C = 1$ the mesh cell is entirely filled with liquid, while for $C = 0$ it is entirely filled with gas. In a mesh cell that instantaneously contains a part of the interface both phases coexist and $O < C < 1$.

The set of equations of the volume-of-fluid method is in principle equivalent to the set of equations of the homogeneous model because we are in some way doing an average over the mesh cell volume. It is further assumed that the mesh cell is so small that the interface curvature is well resolved and that the relative velocity in interface mesh cells is zero.

The mass conservation of both phases is expressed by a transport equation for the liquid volumetric fraction f

$$\frac{\partial C}{\partial t} + \nabla \cdot C\mathbf{v} = 0 \tag{2.33}$$

and by the condition of a divergence free velocity field

$$\nabla \cdot \mathbf{u} = 0 \tag{2.34}$$

As f has large gradients at the interface, Eq. (2.33) is not solved by standard difference schemes because this would result in numerical smearing of the interface. In the VOF method this equation is instead solved in a "geometrical" manner. For this purpose, by taking advantage of the Gauss divergence theorem, the convective term in Eq. (2.33) is written as a surface integral that depends on the velocity and the phase distribution at the faces of a mesh cell. The velocity is obtained by solution of the momentum equation while the phase distribution within a mesh cell is obtained by a reconstruction of the interface.

In general, there are two types of VOF reconstruction schemes. In SLIC methods (Simple Line interface Calculation) the interface is assumed to be orientated parallel to the face of a mesh cell. In a 2D case the interface is therefore orientated either horizontally or vertically. The actual orientation depends on the values of

f in neighboring mesh cells. In PLIC methods (Piecewise Linear interface Calculation) the interface is represented in 2D by a line and in 3D by a plane. The orientation of the line or plane can be arbitrary. A plane is uniquely represented by a point within the plane and by its unit normal vector. For evaluation of the unit normal vector a large number of different reconstruction schemes exists in literature. In general, the unit normal vector is approximated as a discrete representation of relation

$$\mathbf{n} = \frac{\nabla C}{|\nabla C|} \tag{2.35}$$

A disadvantage of current PLIC methods is that the planes representing the interface in two neighboring mesh cells are not continuous at the mesh cell face that both mesh cells are sharing. Once the unit normal vector is computed the point within the plane can be computed by parallel shift of the plane within a mesh cell so that the liquid volume fraction in the cell just equals the value of C in the respective mesh cell.

After completion of the interface reconstruction step, Eq. (2.33) can be solved by computing the liquid volume fluxes across all the mesh cell faces within a time step Δt. For this advection step two different approaches exist in literature. In *operator splitting methods* the reconstruction and advection is done for each coordinate direction separately. Thus, in 3D three consecutive reconstruction and advection steps are required per time step. In *unsplit methods* there is only one reconstruction step and the volume fluxes across the mesh cell faces are computed simultaneously. The disadvantage of the unsplit method is that the same liquid volume may be advected twice or even triply. Nevertheless, a key advantage of the volume-of-fluid method as compared e.g. to the Level Set and Front Tracking method is its excellent conservation of mass of the two phases.

2.3.2 Front Tracking

The Front Tracking method, developed in [24], is based on a local single-field formulation of the momentum equation which is discretized by finite differences. The key idea of the method is to compute the velocity field from the solution of the momentum equation on a fixed regular grid, while the phase interface is

represented by a set of marker particles. These marker particles are advected with the local velocity, which is interpolated from the fixed grid to the position of the marker particles. After the particle advection step is completed the phase interface is newly structured and marker particles are added or removed to ensure a locally adequate resolution of the phase interface. In a next step, the surface tension is computed using polynomial fits and is then interpolated from the front grid to the fixed grid. Also the density and viscosity field are updated to the new interface position. To avoid discontinuities, the surface tension force as well as the density and viscosity are smoothed over two to three mesh cells. Then, the velocity field for the next time step is computed from the single-field momentum equation.

Advantages of the front-tracking method are the accurate representation of the interface as continuous surface. A disadvantage is that topological changes of the interface such as break-up and coalescence are difficult to handle and require additional measures.

2.4 Level Set method

Another possibility than what discussed in subsections 2.3.1 and 2.3.2 is to model the moving front describing the interface by the zero level set of a level set function introduced in 2.2.2, denoted as ϕ. This means that the zero level set of the function is defined accordingly to the position of the interface $\{x \in \Omega : \phi(\mathbf{x}) = 0\} \doteq \Gamma$. At one side of the interface the level set function is defined larger than zero, fluid 1, $\{x \in \Omega : \phi(\mathbf{x}) > 0\} \doteq \Omega_1$ and less than zero, fluid 2, at the other side $\{x \in \Omega : \phi(\mathbf{x}) < 0\} \doteq \Omega_2$. Typically, the level set function is a signed distance function (or more generally a Lipschitz-continuous function). Motivated by the use of a level set as a description of the interface, this approach is known as the level set method.

The Level set method has been developed by Osher and Sethian in 1988 ([11]) and have been applied not only in multi-phase flows but also in many other fields (e.g. computer vision, image processing, seismic analysis, semiconductors, crystal growth, etc.) to tracking various type of discontinuity and it is implemented in many FEM (e.g. COMSOL Multiphysics) and FV (Finite Volumes) software. However, since it consists of a scalar advection equation, it could be applied to

many other fields and techniques. Recently this method was proposed also coupled with a discontinuous spectral element method ([23]).

In the level set method for multi-phase flows the phase distribution is computed by solving the advection equation (2.31) and, since ϕ is a smooth function, this equation can be solved by standard numerical schemes. However some particular schemes are usually used to prevent the mass loss (see **??**).

The Level Set method (like all the front tracking methods) is based on a fully local single-field formulation of the momentum equation described in 2.2. So there is no volume averaging of the governing equations. The surface tension term is given by

$$\mathbf{F}_\sigma = \int_\Gamma \sigma\kappa\mathbf{n}\mathrm{d}S = \int_\Omega \sigma\kappa\delta_\Gamma(\mathbf{x})\mathbf{n}\mathrm{d}\,V \tag{2.36}$$

where $\mathbf{n}$ is the unit normal on the interface pointing into fluid 1 and the curvature κ of the interface can be easily expressed in terms of $\phi(\mathbf{x}, t)$ using Eq. (2.17) as

$$\kappa = -\nabla \cdot \mathbf{n} = -\nabla \cdot \left.\frac{\nabla\phi}{|\nabla\phi|}\right|_{\phi=0} \tag{2.37}$$

while several proposals exist in literature to represent and approximate the Dirac delta function δ_Γ .

Level Set method has many advantages:

- In contrast to the Lagrangian formulation and coinciding with the volume of fluid method, no additional computational elements are used.
- Moreover, no explicit description (parametrization) of the interface is needed any more because it is implicitly captured on the Eulerian grid by the zero level set. This implies that the geometric characteristics of the interface are completed determined by the level set function.
- Therefore, complex interface structure and topological changes can be captured quite naturally because the interface is viewed as a level set. Hence, it avoids the complex techniques or interface reconstructions which plague front tracking methods and VOF.

- In addition to that, there are not many difficulties to use the method in three space dimensions.
- But the central advantage is that fixed-grid finite difference approximations may be used since no moving grid points are involved. Thus, no instability problems occur compared to front tracking methods.
- Finally, special properties of the interface such as local curvature or outer normal can be calculated easily from the level set function.

On the other hand it is necessary to be aware of the disadvantages of the level set approach in order to find methods that reduce them as far as possible:

- High-order accuracy at the front, especially for incompressible two-phase flow with surface tension, is lost compared to front tracking methods.
- To keep the level set function as a signed distance, it must be "reinitialized" after each time step.
- The area of the fluid bubbles or drops must be conserved, which is not guaranteed in advance in numerical computations. To solve this problem there are some specific methods called "mass corrections".
- Another important drawback is that computational expense is introduced additionally caused by the extension of one more space dimension. Recall that a $(n-1)$-dimensional curve (representing the interface) is described by the zero level set of a n-dimensional function.

2.4.1 Reinitialization

Even if we use high-order schemes to solve the level set equation (see **??**) certain flow fields can cause the level set function to generate large gradients polluting the accuracy of the approximation. Thus, researchers have focused on methods to keep ϕ from developing large gradients. In fact, it turns out to be even more useful to force ϕ to approximate a signed distance function with $\nabla\phi = 1$. For example, the smoothness of the level set function is very important when we have

to approximate the interface curvature κ which depends on the second derivatives of ϕ that is very sensitive to numerical errors.

In the original paper [11], the level set function was initialized as $\phi = 1 \pm d^2$ where d is the distance to the interface and the ''$\pm$'' sign depends on which side of the interface a grid point is located on. This was later changed to be a signed distance function. After the advection of the interface, it is uncommon for the level set function to remain a signed distance function, which means that ϕ needs to be reinitialized at regular intervals in order to limit numerical dissipation. A simple and accurate technique is to calculate how far each grid point is from the zero isocontour of the level set directly. This technique is quite expensive in practice, and as such cannot be used in large real world examples or with schemes that require frequent reinitialization. A more practical alternative is to evolve the interface in the normal direction at unit speed keeping track of the amount of time that it takes for the interface to cross over each grid point. The crossing time gives the distance value for a grid point. If we evolve the interface in both the normal and negative normal direction at the same time, we obtain an equation of the form

$$\phi_t + S(\phi_0)(|\nabla\phi| - 1) = 0 \tag{2.38}$$

which is commonly known as the *reinitialization equation*. Here $S(\phi_0)$ is a smeared out sign function approximation of the H function.

One difficulty encountered when solving Equation (2.38) is that the interface moves by a small amount due to numerical truncation error. In an attempt to alleviate this difficulty, Ref. [22] proposed a reinitialization method that attempts to preserve the partial volume cut by the interface in each cell (see 7.6).

2.4.2 Narrow band Level Set

In the level set formulation, both the level set function and the speed are embedded into a higher dimension. This then implies computational labor through the entire grid, which is inefficient.Assuming N grid points in each space dimension of a three-dimensional problem, for a simple problem of straightforward propagation with speed $V = 1$, if it takes roughly N time steps for the front to propagate through the domain (here, the CFL condition is taken almost equal to unity), we

obtain approximately an $O(N^4)$ method. Considerable computational speedup in the level set method comes from the use of the *narrow-band level set method*. It is clear that performing calculations over the entire computational domain is wasteful. Instead, an efficient modification is to perform work only in a neighborhood (or "narrow band") of the zero level set. This drops the operation count in three dimensions to $O(kN^3)$, where k is the number of cells in the narrow band. This is a significant cost reduction; it also means that if we are interested only on the interface position, velocities need only to be constructed at points lying in the narrow band, as opposed to all points in the computational domain.

To use the narrow band method we have to mark in some way the cell in a region close to the interface and solve the advection equation and the reinitialization equation only for these cells.

2.4.3 Fast marching method

An alternative of the narrow band method is the *fast marching method*. Instead of solving the partial differential equation (2.38) for a number of time steps in fictitious times, one can instead start at the interface and march outwards creating a signed distance function in a single sweep through the data. In order to apply this algorithm, one first needs to initialize all the grid points adjacent to the interface with a ''correct'' ϕ value. These nodes are then considered "done", all their neighbors are considered "close" and all other nodes are considered "far". Next, all grid nodes in the "close" set have tentative values of ϕ calculated using only values from "done" points. Then the smallest of these is added to the done set, all its neighbors are added to the "close" set, and the tentative values of these neighbors are calculated or updated using the fact that there is now one more node in the done set. The algorithm is terminated when a thick enough band of points around the interface has been added to the "done" set.

Chapter 3

Hamilton-Jacobi Equations

From the mathematical point of view, the level set method consists in solving Hamilton-Jacobi (H-J) equations in two essential steps: evolution of the level set function accordingly to the underlying flow field and reinitialization. We discuss briefly their main properties, in particular the close relation to hyperbolic conservation laws and discretizations techniques, designed originally for hyperbolic conservation laws, adapted to H-J equations (in particular discretization of the reinitialization problem is explained).

3.1 Definition

An *Hamilton-Jacobi equation* is defined as

$$u_t + H(x, u, \nabla u, t) = 0 \tag{3.1}$$

where H is called the *Hamiltonian function.* In the following we consider his associated Cauchy problem

$$\begin{aligned} u_t + H(\nabla u) = 0 \quad & in\, \mathbb{R}^n \times (0, \infty) \\ u(x, 0) = u_0(x) \quad & in\, \mathbb{R}^n \end{aligned}$$

Basically, all the equations concerning Level Set method considered in this thesis are of this kind. Therefore, it seems to be convenient to summarize briefly their

main properties and some basic numerical approaches to compute approximate solutions. A more detailed and general discussion including existence, uniqueness, and stability proofs for smooth Hamiltonian H and smooth initial data u_0 is given by in [26].

3.2 Mathematical Properties

It is well-known that H-J equations do not have classical solutions so only weak solutions can be expected. Unfortunately, weak solutions are generally not uniquely defined. Therefore, the notion of *viscosity solution* is introduced in [26] which is a weak solution satisfying some entropy conditions aimed to single out the physically relevant solution among all weak solutions. Thus, viscosity solutions are uniquely defined. Moreover, they are proven to be stable with respect to small perturbations in the computational data.

Let us define the "super-differential" D^+u as

$$D^+u(x_0) = \left\{ p : \limsup_{x_1 \to x_0} \frac{u(x_1) - u(x_0) - p(x_1 - x_0)}{|x_1 - x_0|} \leq 0 \right\} \tag{3.2}$$

and the "sub-differential" D^-u as

$$D^-u(x_0) = \left\{ p : \liminf_{x_1 \to x_0} \frac{u(x_1) - u(x_0) - p(x_1 - x_0)}{|x_1 - x_0|} \geq 0 \right\} \tag{3.3}$$

A continuous function u is a "viscosity super-solution" of Eq. (3.1)

$$H(x, u(x), p, t) \leq 0, \forall x \in \Omega, \forall p \in D^+u(x), \forall t \tag{3.4}$$

or a "viscosity sub-solution" if

$$H(x, u(x), p, t) \geq 0, \forall x \in \Omega, \forall p \in D^-u(x), \forall t \tag{3.5}$$

Finally u is a *viscosity solution* if it is both a viscosity super-solution and a viscosity sub-solution.

The name, viscosity solution, refers to the "vanishing viscosity" approach, where

the limiting solution of the viscous equations for the viscous term tending to zeros is determined. Although the formal definition of a viscosity solution for H-J equations is different, it is proven to be this limit.

Typically solutions of H-J equations are continuous but with discontinuous derivatives even for smooth initial data u_0. This is similar to the analysis of hyperbolic conservation laws, where discontinuities (shocks) may develop from a C^∞ function as an initial condition. In addition to that, an analytical relation exists between H-J equations and hyperbolic conservation laws. In one dimension the H-J equation can be regarded as the "once integrated" conservation law

$$\tilde{u}_t + (H(\tilde{u}))_x = 0 \tag{3.6}$$

where $\tilde{u} = u_x$. This relation holds if the solution u is twice continuously differentiable. Although such a relation does not exist in the multi-dimensional case, it emphasizes the close connection between these kind of partial differential equations (see, e.g., [27]).

3.3 Numerical Methods

The first approach for solving H-J equations was to use monotone schemes that are proved to converge to the viscosity solution but, unfortunately, they are at most first-order accurate. On the other hand, traditional high-order methods introduce oscillations in the presence of discontinuous derivative. Based on the relation given above between hyperbolic conservation laws and H-J equations, high-order numerical methods, originally designed for hyperbolic conservation laws, have been adapted to H-J equations. We present here some of these methods. For details see [27].

Firstly, only the space derivatives are discretized while the time dependency is left continuous. Thus, an ordinary differential equation is obtained, which is called "semi-*discrete equation*". Finally any high-order ODE solver may be applied to discretize the temporal derivative.

Since the space and time dependencies are decoupled, it is much easier to obtain high-order accuracy. This approach, which is sometimes also called "*method of*

lines", is useful especially if we have two or more space dimensions.

3.3.1 Discretization in Space

ENO Schemes

In order to avoid oscillations in the presence of non-smooth data using higher order schemes, so-called essentially non-oscillatory (ENO) schemes have been constructed by Harten, Osher, and Shu ([29]). Originally, they have been designed for hyperbolic conservation laws. Osher and Sethian ([11]) extended ENO schemes in order to apply them to H-J with good results.

The main idea for approximating the partial derivatives $\phi_x(x_i)$ and $\phi_y(x_i)$ (to simplify the notation we use only two space dimensions) was to construct stencil interpolating polynomials $P^{\phi,r}(x_i)$ of degree r as an approximation of the function ϕ. Hence, the derivative $\frac{d}{dx}P^{\phi,r}(x_i)$ of this polynomial is used as an approximation of the partial derivative of the level set function $\phi_x(x_i)$. In particular, a forward and a backward approximation is computed for each point x_i. For the forward approximation a stencil interpolating polynomial $P^{\phi,r}_{i+\frac{1}{2}}(x_i)$ defined on the interval $[x_i, x_{i+1}]$ is computed and ϕ_x is approximated by $\phi^+_x(x_i) = \frac{d}{dx}P^{\phi,r}_{i+\frac{1}{2}}(x_i)$. Analogously $P^{\phi,r}_{i-\frac{1}{2}}(x_i)$ is computed on $[x_i, x_{i+1}]$ giving a backward approximation $\phi^-_x(x_i) = \frac{d}{dx}P^{\phi,r}_{i-\frac{1}{2}}(x_i)$.

The freedom in choosing additional stencil points for the forward and backward approximations (if the degree of the polynomial is bigger than one) is used to obtain information always from the region where the function is locally smoother. This is achieved by selecting the grid points where the approximated derivative has the smallest absolute value. A simple construction method can be built in the following way, assuming a uniform mono-dimensional grid.

First define undivided differences

$$\psi(j,0) = \phi(x_j) \tag{3.7}$$

$$\psi(j,k) = \phi(j+1,k-1) - \phi(j,k-1) \qquad k = 1,\ldots,r \tag{3.8}$$

Hence $\psi(j,0)$ approximate the function ϕ and $\psi(j,k)$ its k^{th} derivative in $[x_j, x_{j+k}]$. The stencil for the forward approximation of the partial derivative

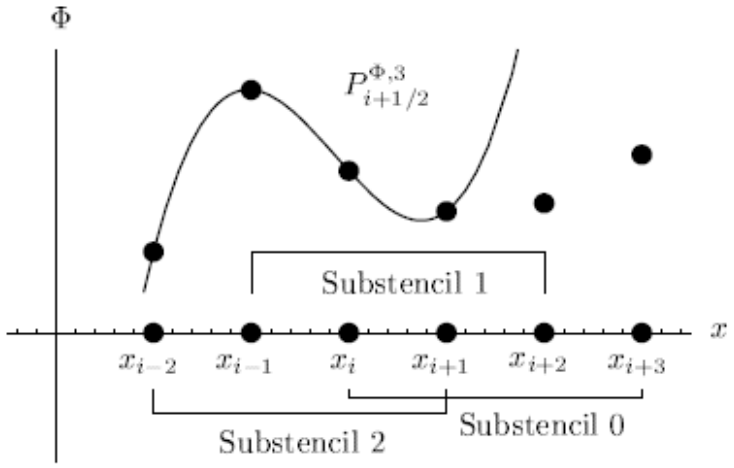

Figure 3.1: *Example of ENO scheme. Stencil interpolating polynomial $P^{\phi,3}_{i+\frac{1}{2}}(x)$ used for the forward approximation of the partial derivative $\phi^+_x(x)$ at $x = x_i$. Sub-stencil 2 is selected from three possible sub-stencils.*

is chosen, according to the smoothness property mentioned above, defining the leftmost point in the stencil for $P^{\phi,r}_{j+\frac{1}{2}}(x)$ as

$$i(j) = j \qquad \forall \text{stencil point } j \tag{3.9}$$

$$i(j) = i(j) - 1 \qquad \text{if } |\psi(i(j), k)| > |\psi(i(j) - 1, k)| \tag{3.10}$$

for $k = 2, \ldots, r$.

To show an example let us look at Figure 3.1.The sub-stencil 0 is selected by default (since $i(j) = j$). Then the absolute value of the second derivatives in x_{j-1} results smaller than the one in x_j (i.e. if $|\psi(i(j), 2)| > |\psi(i(j) - 1, 2)|$), so sub-stencil 1 will be selected ($i(j) = i(j) - 1$). This process continues until $k = r$. Finally, the derivative of the stencil interpolating polynomial is given by

$$\phi^+_x(x_i) = \frac{1}{\Delta x} \sum_{k=1}^{r} c(i(j) - j, k)\psi(i(j), k) \tag{3.11}$$

where

$$c(m, k) = \frac{1}{k!} \sum_{s=m}^{m+k-1} \prod_{\substack{l=m \\ l \neq s}}^{m+k-1} (-l) \tag{3.12}$$

is a matrix independent from ϕ. So it can be computed only once and may be stored for later use. The computation of the backward approximation is done in a similar way.

Weighted ENO Schemes

A further extension of ENO schemes are Weighted ENO (WENO) schemes, applied by Jiang and Peng ([28]) to H-J equations.

In contrast to the ENO scheme above, which selects one (the smoothest) sub-stencil to construct the interpolating polynomial, the WENO scheme uses a linear combination of all possible sub-stencils and weights them according to the local smoothness of the function. In other words, r different polynomials (for each sub-stencil, one polynomial) are computed for each grid point and a weighted average of all polynomials is used. Therefore WENO schemes may be considered as central schemes in regions where the solution is smooth. Thus, is much more accurate as the original ENO scheme. If there are singularities in the solution, the WENO scheme will adapt the stencil to avoid oscillations. In order to achieve these features the weights are defined according to the following two principles:

1. if ϕ is smooth on the whole stencil, information from all possible sub-stencil will be used to approximate the derivative;

2. if the stencil contains a singularity, the weights adaptively approach the ENO "1/0-weights" to avoid oscillations.

3.3.2 Approximation of the Hamiltonian

Consider the H-J equation

$$\begin{aligned} \phi_t + H(\phi_x, \phi_y) = 0 \quad in\, \mathbb{R}^2 \times (0, \infty) \\ \phi(x, y, 0) = \phi_0(x, y) \quad in\, \mathbb{R}^2 \end{aligned} \tag{3.13}$$

The semi-discrete equation is obtained by approximating the Hamiltonian H by a Lipschitz-continuous monotone flux function $\widehat{H}(\phi_x^+, \phi_x^-, \phi_y^+, \phi_y^-)$ consistent with

H (i.e. $H(u,u,v,v) = H(u,v)$). With monotonicity we mean that H is non-increasing in its first and third argument and non-decreasing in its other arguments. A variety of monotone flux functions is given in [11]. As pointed out in [28] the Godunov flux $\widehat{H}^G$ for the Level Set reinitialization equation 2.38 is

$$\widehat{H}^G = \begin{cases} s\sqrt{\left(\max\left\{\left(\phi_x^+\right)^-, \left(\phi_x^-\right)^+\right\}\right)^2 + \left(\max\left\{\left(\phi_y^+\right)^-, \left(\phi_y^-\right)^+\right\}\right)^2} - 1 & \text{if } \phi_0 \geq 0 \\ s\sqrt{\left(\max\left\{\left(\phi_x^-\right)^-, \left(\phi_x^+\right)^+\right\}\right)^2 + \left(\max\left\{\left(\phi_y^-\right)^-, \left(\phi_y^+\right)^+\right\}\right)^2} - 1 & \text{if } \phi_0 < 0 \end{cases} \tag{3.14}$$

where $s = \text{sign}(\phi_0)$, $(\cdot)^+ = \max\{\cdot, 0\}$ and $(\cdot)^- = -\min\{\cdot, 0\}$.

However, in [22] another approach is applied to choose the appropriate approximation of the partial derivative. The authors used a sort of upwind method because equation 2.38 can be written in the form of a convection equation with characteristics propagating outward from the zero level set. This method uses backward approximations if the characteristics are propagating from below to the point under consideration while it uses forward approximations if the characteristics are propagating from above. First we approximate derivatives as

$$\phi_x \approx \begin{cases} \phi_x^+ & \text{if } \phi_x^+ \text{sign}(\phi_0) < 0 \text{ and } (\phi_x^+ + \phi_x^-)\,\text{sign}(\phi_0) < 0 \\ \phi_x^- & \text{if } \phi_x^- \text{sign}(\phi_0) > 0 \text{ and } (\phi_x^+ + \phi_x^-)\,\text{sign}(\phi_0) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.15}$$

and ϕ_y is approximated in a similar way. Then the Hamiltonian is approximated by using the corresponding forward and backward approximations. It can be proved that the Hamiltonian approximated in this way is equivalent to $\widehat{H}^G$.

3.3.3 Discretization in Time

Runge-Kutta type procedures for the time-discretization are often used to discretize time derivatives in H-J equations. We explain here how to apply these schemes, we refer to [32] for detailed derivations.

Order	α_{kl}	β_{kl}
2	1 $\frac{1}{2}\quad\frac{1}{2}$	1 $0\quad\frac{1}{2}$
3	1 $\frac{3}{4}\quad\frac{1}{4}$ $\frac{1}{3}\quad 0\quad\frac{2}{3}$	1 $0\quad\frac{1}{4}$ $0\quad 0\quad\frac{2}{3}$
4	1 $\frac{1}{2}\quad\frac{1}{2}$ $\frac{1}{9}\quad\frac{2}{9}\quad\frac{2}{3}$ $0\quad\frac{1}{3}\quad\frac{1}{3}\quad\frac{1}{3}$	$\frac{1}{2}$ $-\frac{1}{4}\quad\frac{1}{2}$ $-\frac{1}{9}\quad\frac{1}{3}\quad 1$ $0\quad\frac{1}{6}\quad 0\quad\frac{1}{6}$

Table 3.1: *Coefficients for Runge-Kutta TVD schemes.*

To obtain ϕ^{n+1} from ϕ^n let us define

$$L_{ij}^{(0)} = -\Delta t \hat{H}\left(\phi_x^+, \phi_x^-, \phi_y^+, \phi_y^-\right) \tag{3.16}$$

$$\phi_{ij}^{(0)} = \phi_{ij}^n \tag{3.17}$$

where $\hat{H}$ is a numerical flux and compute

$$\phi_{ij}^{(k)} = \sum_{l=0}^{k-1}\left(\alpha_{kl}\phi_{ij}^{(l)} + \beta_{kl}L_{ij}^{(l)}\right)\ , k = 1, \ldots, r \tag{3.18}$$

where r is the order of the R-K scheme and α_{kl}, β_{kl} are suitable coefficients (see, e.g., Table 3.1). Finally the solution at time $n+1$ is computed as $\phi^{n+1} = \phi_{ij}^{(r)}$ where r denotes the level of the Runge-Kutta method.

Very important are the Total Variation Diminishing schemes. A numerical method is said to be Total Variation Diminishing (TVD) if

$$TV\left(\phi^{n+1}\right) \leq TV\left(\phi^n\right) \tag{3.19}$$

where TV is a function that represent the sharpness of the solution ($TV(\phi) = \int_\Omega |\nabla\phi| d\Omega$).

It can be proved that the method (3.18) is TVD under the CFL condition $\frac{\Delta t}{\Delta x} \leq c\lambda_0$ if the corresponding Euler forward version is TVD under the CFL condition $\frac{\Delta t}{\Delta x} \leq \lambda_0$ where c is a constant.

Chapter 4

Grid Generation

In the previous chapters we analyzed the main mathematical models for Multi-Fluid Dynamics and the mathematical properties of the Level Set equation. At this point is very important to introduce the different approaches for grid generation and mesh refinement because, especially in CMFD, the choice of the computational grid is deeply linked with the model. Many aspects have to be evaluated:

- Accuracy of solution.
- Computational effort to build the grid.
- Difficulty of implementing particular numerical methods.
- Adaptivity.

In the first section we give a general overview on grid generation, while in the second one we briefly discuss how to represent complex geometries in a computational domain. Finally in the third section we analyzed two different methods of Adaptive Mesh Refinement.

4.1 Grid generation techniques

Grid generation methods are typically grouped in one of three categories shown in figure 4.1: structured, multi-block, and unstructured. These categories describe the layout of the physical cells and the neighbor relationships between cells in

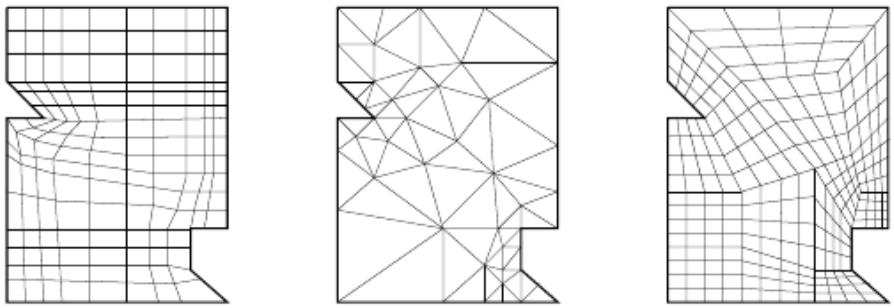

Figure 4.1: *Types of meshes: (a) structured, (b) unstructured, (c) block-structured*

a given grid. Structured grids are built with a rigid topological framework, unstructured grids have no (enforced) underlying structure and multi-block grids are usually an unstructured collection of structured blocks.

4.1.1 Structured grids

Structured grids are built with a repeating geometric and topological structure. They are usually formed from hexahedra (with eight nodes and six quadrilateral facets) or "bricks", as this is perhaps the simplest repeating topology to compute on (figure 4.2). Structured grids are often simpler to deal with, specifically in terms of application development, computation, and visualization. This simple structure often simplifies the computational connectivity of the grid, allowing for very efficient computation on modern computers. This rigid structure may favor the computation with finite difference method leading to simple and accurate computation of spatial derivatives and often cancellation of higher-order error terms. However, as it usually requires an inordinate amount of expert-type assistance in laying out a suitable structured grid around any complicated geometry, such as an airplane.

4.1.2 Block-structured grids

A block-structured grid is a collection of structured grids that together fill the domain, as shown in Figure 4.2. It inherits most of the computational efficiency of

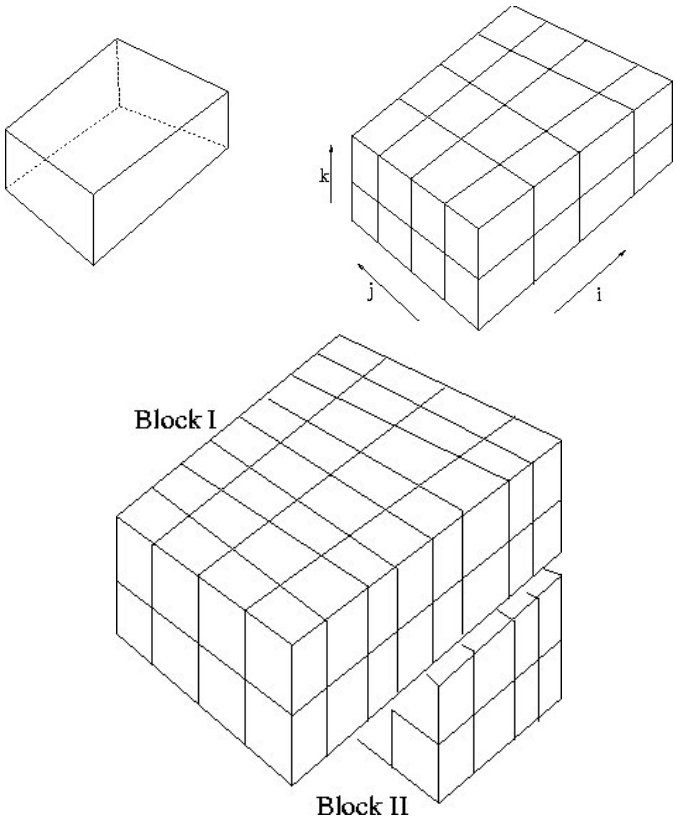

Figure 4.2: *Three-dimensional (a) structured grid and (b) block-structured grid.*

a structured grid, but the difficulty now becomes the communication between the blocks. Since geometric components can be dealt with nearly independently, it is easier to grid a complicated geometry with a multi-block then a structured grid, but filling in complex geometry intersections and building blocks that properly share boundary surfaces usually require a large amount of expert assistance and partially offset the benefits of the multi-block approach.

Because of their generality, multi-block codes have been very successful at computing aerodynamic problems. Research continues, however, in generating multi-block grids automatically. Until an automatic method is accepted, the setup time for a new configuration still dominates the simulation. For example in many real application like aerospace industry, using commercial multi-block grid generators, the grid generation can take several days. This could be often much more than the time needed to solve Reynolds-Averaged Navier-Stokes (RANS) equations on

this grid.

4.1.3 Unstructured grids

Unstructured grids are characterized by no such repeating geometry and structure that can be controlled only very locally. Unstructured grids are typically formed from simplexes such as tetrahedral, and the fact that they have no repeating structure can make it very difficult to create and compute the necessary cell-to-cell connectivity. They are used mainly with finite elements methods, which are generally no more complicated on unstructured grids than on structured ones but can be also used for finite volumes methods. The real advantage is its simple automatic generation and refinement (see figure 4.3).

4.2 Meshing complex geometries

4.2.1 Body fitted coordinates

For meshing complex geometry and curved-shaped structures a common approach is to use Body Fitted Coordinates (BFC). They can be employed either in structured or unstructured meshing frameworks, under H-, C- and O-structures. The technique has some advantages, but necessitates a large overhead (some times several weeks to generate and re-generate the grid). It induces furthermore discretization errors (of the gradients) due to the quality (aspect ratio) of the cells that can be non-negligible as compared to what might be the contribution of Sub-Grid Scales (SGS) in Large-Eddy Simulations (LES), for example. A typical example of a curvilinear C-O grid of a stack of blade turbines is shown in Figure 4.3. If the core body of the blades is well represented, there is an uncertainty on the quality of the grid when it comes to representing the critical tip-leakage area as shown in the bottom panel. Here it is clear that structured BFC grids will not be sufficient, and the numerical errors induced by grid skewness are expected to be very large there, compromising the CFD solution and convergence. These flow zones are known to be the source of viscous dissipation, a grid such as the one shown in Figure 4.3 may simply provide the wrong picture of what might be expected in reality.

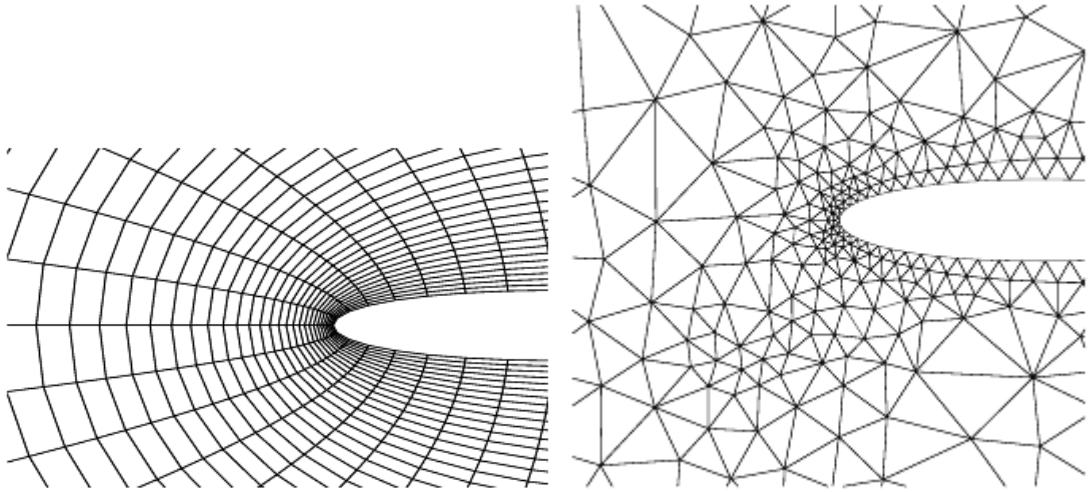

Figure 4.3: *Example of a structured body fitted grid (a) compared with an unstructured grid (b).*

4.2.2 Cartesian grid methods

Cartesian methods are a class of methods developed to avoid the difficulties of body fitted grids and which are subject of increasing interest in the last years. They use a base Cartesian grid (i.e. a structured grid where each grid plane is parallel to a pair of coordinate axes) and the geometry is "immersed" in the grid as an internal interface. While the grid exhibits much of the same regularity that makes structured grids attractive, the grid is not tied to the structure itself. This can be an advantage when building the grid, but flow features that are caused by structural features may not be well resolved. Also, regions of smooth flow may be allocated more grid (and the computational resources this implies) than they need. This can be solved using a mesh refinement strategy (see 4.3) while the first problem can be solved with special treatment for the cells near the boundaries.

The first possible approach is the "cut-cell" method ([34–36]). The grid is Cartesian in all the domain with all solid boundaries included, the cells inside the solids are neglected and the cells that intersect the boundaries are cut (see Figure 4.4). This leads to simple schemes and numerical methods in all the cells far from the boundaries but these *cut cells* have no specific orientation or shape, and they often slow the computation, degrade the accuracy, and inhibit computation of the higher-order derivatives and need special treatments. However, this approach,

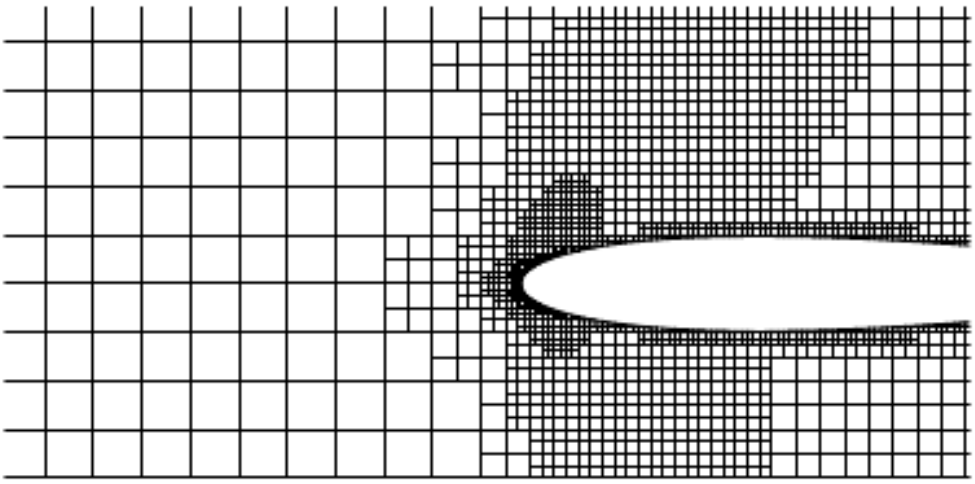

Figure 4.4: *Example of Cartesian grid refined near the surfaces. The closest cells are cut by the surface.*

together with mesh adaptation can preserve boundary fidelity and can be competitive with block-structured and unstructured (tetrahedron-based) methods. In figure

4.2.3 Immersed boundaries

Another possible approach to represent complex geometries without BFC is the Immersed boundary (IB) technique. It has been introduced by Peskin in 1972 and developed in the late nineties for the simulation of flow interacting with solid boundary (see for review [46]), under various formulations. The mostly known formulation for instance employs a mixture of Eulerian and Lagrangian variables, where the solid boundary is represented by discrete Lagrangian markers embedding in and exerting forces to the Eulerian fluid domain. The interactions between the Lagrangian markers and the fluid variables are linked by a simple discretized delta function. The IB methods are all based on the direct momentum forcing (penalty approach) on the Eulerian grids, with various forcing formulations on the Lagrangian marker. The forcing should be performed such as it ensures the satisfaction of the no-slip boundary condition on the immersed boundary in the intermediate time step. This forcing procedure involves solving a banded linear

system of equations whose unknowns consist of the boundary forces on the Lagrangian markers, thus, the order of the unknowns is one-dimensional lower than the fluid variables. Numerical experiments show that the stability limit can in general be altered by the proposed force formulation, in that the second-order accuracy of the adopted numerical scheme is degraded to first order, and in the best conditions to 1.5 but this loss can be compensated using fine grids near the boundaries.

This IB approach is also very interesting because introduces boundary conditions directly inside the discrete equation and partially decouples the grid generation from the geometry involved so that the same grid can be used for many different geometries reducing drastically the time of meshing.

4.3 Adaptive Mesh Refinement

Mesh refinement is very important for those fields in which there are high gradients in the solution or when it is impossible to solve the flow in all the domain with approximately the same resolution. In fact with usual single-block or multi-block grid we have to keep a good mesh quality which means that the mesh size gradient cannot be to large.

In some applications this refinement must be done dynamically in time because the "critical" zones where we need a better accuracy varies with time. For example, this is the case of shock hydrodynamics and interface tracking for multi-phase flows where *Adaptive Mesh Refinement (AMR)* is needed. On the other hand, in many aerodynamically and wind engineering simulations we already know at the meshing step, which zones must be refined so a fixed grid with refined zones can be enough.

Presently, unstructured mesh approaches have been adapted with success to perform adaptive mesh refinement in a finite volumes (face-based) or finite element (edge-based) framework. In particular the Adaptive Finite Element Method (AFEM) has very important mathematical properties. However in nearly all of these implementations, mesh connectivity data is constructed from the mesh by numbering the different element types (nodes, edge, faces, cells), and storing them in lists, typically integer arrays. These different mesh elements are then cross indexed by integer numbers, resulting in what is commonly referred to as mesh

connectivity. So for this type of storage and addressing of the data, the addition and deletion of new cells into the mesh to locally resolve flow phenomena causes global changes to the data structures. When performing mesh refinement for steady state problems, the cost of performing the small number of refinement passes and subsequent data re-ordering operations is tolerable when compared to the overall cost of performing the steady state calculation. On the other hand, for unsteady problems where mesh refinement must occur at every time step, this cost would be prohibitive.

An alternative of unstructured AMR is the *Cartesian AMR* (also called *Hierarchical Adaptive Mesh Refinement HAMR)*. Cartesian grid refinement could be done mainly in two different way: a block-based refinement or a tree-based refinement. Both these methods are very powerful coupled with methods like IB because this gives the possibility to ignore the geometry and use a Cartesian grid that cover all the domain as the "father" grid and then refine it where is needed.

There are several studies on Cartesian AMR for both compressible ([33; 35–37; 39]) and incompressible flows ([14; 15; 21]), with or without phase tracking methods. We deal only with Cartesian refinements for incompressible flow although many concepts are common in all the methods.

4.3.1 Block based refinement

This type of AMR (also called *Patch based AMR*) was developed by Berger and Oliger ([38; 39]) in 1984 and had a good success in the last years. This AMR-method follows a patch-wise refinement strategy (see fig 4.5). The algorithm begins with the entire computational domain covered with a coarsely resolved base-level regular Cartesian grid. Cells being flagged by various error estimators are then clustered into rectangular boxes of appropriate size. They describe refinement regions geometrically and sub-grids with refined mesh spacing in space and time are generated according to them. Refined grids are derived recursively from coarser ones and an entire hierarchy of successively embedded grid patches is therefore constructed. All grid patches are logically rectangular and only a specific integration method for single rectangular grids is required. The adaptive algorithm calls this application dependent routine automatically. Further on it uses conservative

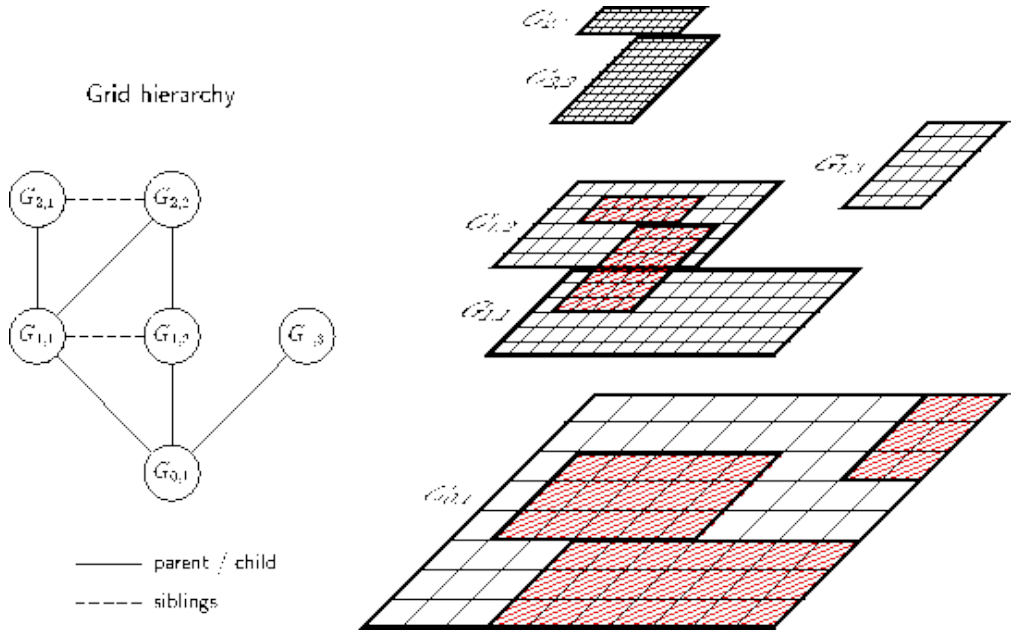

Figure 4.5: *The block-structured refinement strategy creates a hierarchy of properly nested sub-grids.*

interpolation functions to transfer cell values between refined sub-grids and their coarser parents appropriately.

It is important to note, that refined grids overlay the coarser sub-grids from which they have been created. The numerical solution on a particular level is first of all advanced independently. Values of cells covered by refined sub-grids are overwritten by averaged fine grid values subsequently. The resulting extra work is usually negligible compared to the computational costs for integrating the superimposed refinement grids.

Replacing coarse cell values by averaged fine grid values modifies the numerical stencil on the coarse grid and in general the conservation property is lost. A flux correction replacing the coarse grid flux at the affected side of a neighboring cell by accumulated fine grid fluxes is necessary to ensure conservation. This so called conservative fix up is usually implemented as a correction pass (see 6.1.2) . In two and three space dimensions hanging nodes additionally have to be considered.

An important difference of the AMR approach in comparison to tree based adaptive strategies (see next section 4.3.2) is the application of refined time-steps

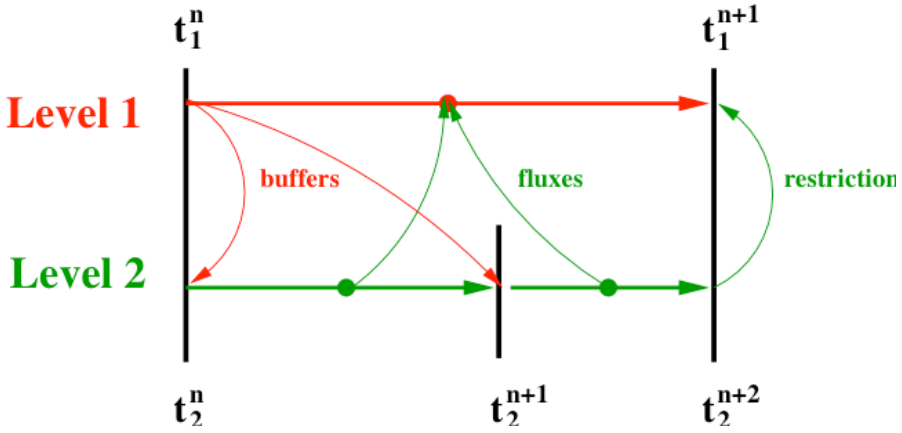

Figure 4.6: *AMR iterative sub-stepping*

on finer sub-grids. The algorithm follows a recursive integration procedure that allows the construction of internal boundary conditions for refined sub-grids by time-space interpolation. By applying the same refinement factor in time as in space the Courant number in principle remains unchanged on refined sub-grids. Figure 4.6 shows this time-stepping procedure and the information that must be passed between the different level.

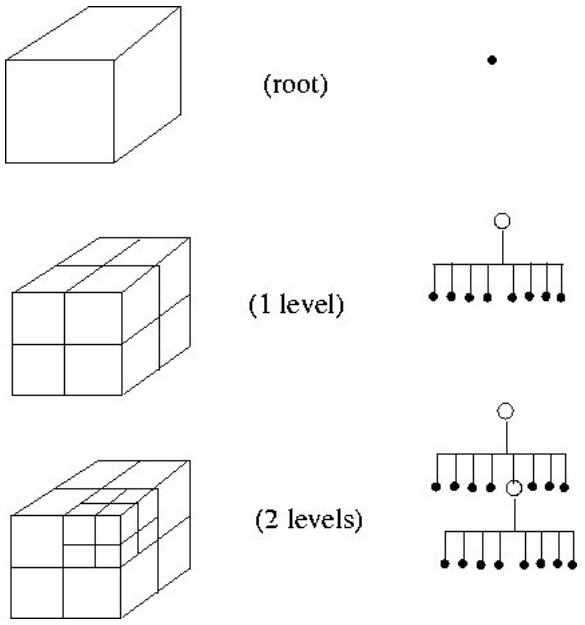

Figure 4.7: *Octree grid.*

4.3.2 Tree-based refinement

A tree-based grid, like the octree (Figure 4.7), falls somewhere in between structured and unstructured grids. Octree structure has been used since 1970 for image processing and computer graphics applications and started to be adopted also for Euler and Navier-Stokes equations later. This approach has been use by many authors ([14; 15; 33]) coupled with "cut-cells" method or with different types of IB methods.

Tree-based grids have repeating geometric structure, but there is no rigid connectivity. This produces a combination with some of the best (and of the worst) properties of both structured and unstructured grids. Since an octree is built from axis-oriented cubes, the benefits of accurate spatial derivatives and high-order cancellation are often found. The repeating structure of the tree yields simple connectivity computation.

In Figure 4.8 we can see an example of how these two methods can cover a curved shape. The tree-based approach is more flexible and can use less cells to cover the interested region.

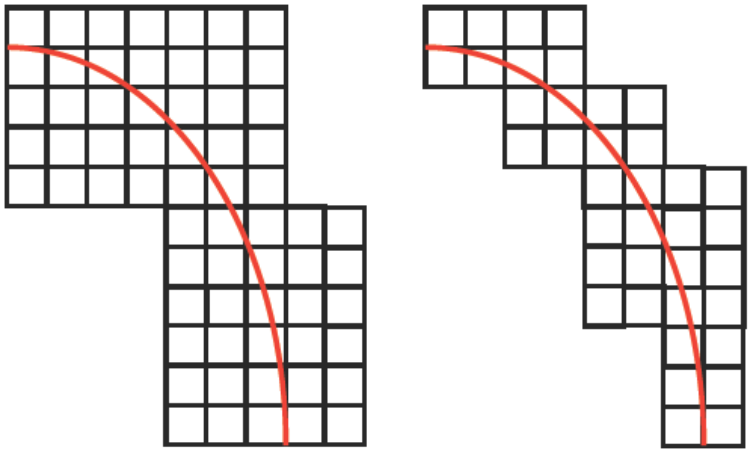

Figure 4.8: *Block based versus Tree based refinement.*

4.3.3 Refinement criterion

Various choices are possible for the refinement criterion in both block based and tree based AMR. The best option is to use Richardson extrapolation to obtain a numerical approximation of the truncation error of the whole method used. A simpler criterion is based on the norm of the local vorticity vector. Specifically, a cell could be refined whenever

$$\frac{h||\nabla \times \mathbf{u}||}{\max ||\mathbf{u}||} > \tau \tag{4.1}$$

where the maximum is evaluated over the entire domain. The threshold value τ can be interpreted as the maximum acceptable angular deviation (caused by the local vorticity) of a particle traveling at speed $\max ||\mathbf{u}||$ across the cell.

Chapter 5

TransAT

Before going on with the specific work done and the results obtained, in this chapter we analyze the CMFD (Computational Multi-Fluid Dynamics) software *TransAT*, developed and distributed by ASCOMP GmbH and in particulare the *Immersed Surfaces Technique* used to represent solid boundaries.

5.1 TransAT

TransAT (Transport Phenomena Analysis Tool) is a structured, curvilinear grids, finite-volume, Navier-Stokes solver for incompressible and weakly compressible multi-phase flows developed by ASCOMP GmbH. It has the following physics modeling capabilities:

- Mass, momentum, heat transport modeling in single and two-phase flow conditions.
- Passive concentration transport.
- Axis-symmetric flows. (Swirl flow in upcoming release)
- Reynolds averaged turbulence modeling: two equation $k - \epsilon$ models with specialized near-wall modeling.
- Unsteady turbulence modeling: Large eddy simulation, Direct numerical simulation.

- Non-Newtonian transport property variations.
- Buoyancy driven flows.
- Two-phase flows with surface tension, and dynamic contact angle modeling (density ratios up to $\approx 10,000$).
- Liquid-vapor flows with phase change models.
- Dispersed two-phase flows with particles, drops, bubbles.
- Conjugate heat transfer with solid boundaries or obstacles using Embedded Interface method.

The following numerical methods are available in TransAT:

- Higher-order explicit time stepping: Runge-Kutta $2-4^{th}$ order.
- Implicit time stepping: 1^{st} order and 2^{nd} order Crank-Nicholson.
- Various monotonic convection schemes: HLPA, Quick, Hybrid.
- Solvers: Stone's Implicit Procedure (SIP), Geometric Multigrid (GMG) using SIP, Preconditioned (SIP/GMG) GMRES, Preconditioned Bi-Conjugate Gradient Squared.
- Lagrangian particle tracking.
- Interface tracking with Level Sets and Volume of Fluid (VOF).
- Level Set reinitialization using 3^{rd}order WENO scheme.
- Inflow, Outflow boundary conditions: velocity or pressure specified.

TransAT can run with single-block curvilinear grid that can be created either with commercial mesh generator or with specific software of ASCOMP. One of these is *TransAT-Mesh*, a Cartesian mesh generator that will be described in details in 6.2.

One of the main characteristics of TransAT is the capability to solve multiphase flow problems within complex geometries. This is achieved with the *Immersed Surfaces Technique*.

5.2 Immersed Surfaces Technique

The Immersed Surfaces Technique (IST) is a special IB method developed by ASCOMP GmbH, although other similar approaches have been developed in parallel. The underpinning idea is inspired from Interface Tracking techniques for two-phase flows (VOF and Level-Set), where free surfaces are described by a hyperbolic convection equation advecting the phase color function. In the IST the solid is described as the second "phase", with its own thermo-mechanical properties. The technique differs substantially from the IB method discussed above (see 4.2.3), in that the jump condition at the solid surface is implicitly accounted for, not via direct momentum forcing (using the penalty approach) on the Eulerian grids. This has the major advantage to solve conjugate heat transfer problems, in that conduction inside the body is directly linked to external fluid convection.

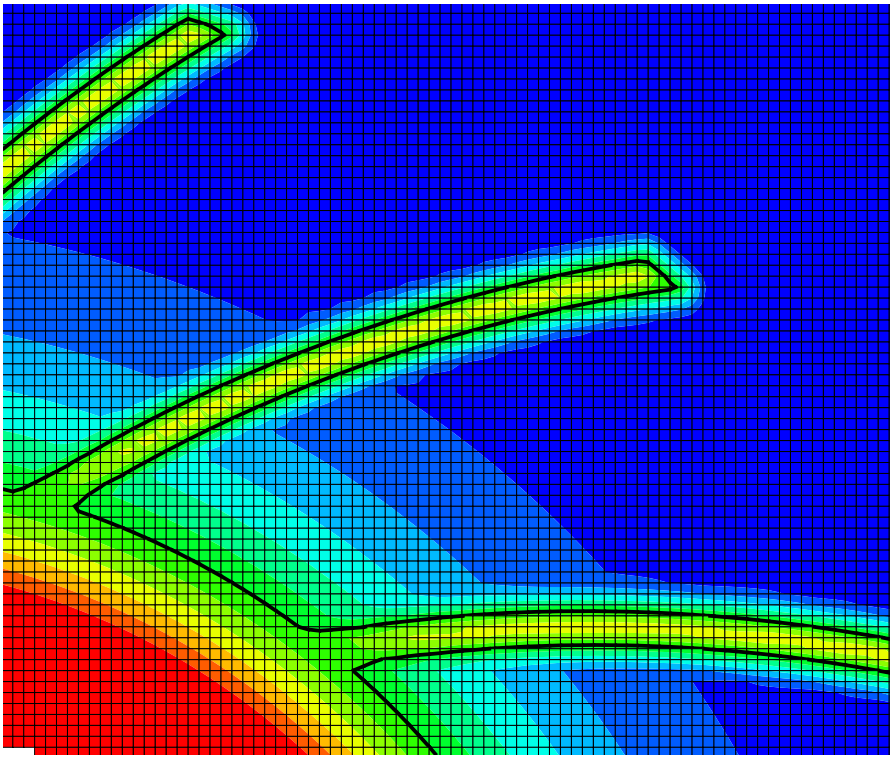

Figure 5.1: *Representation of the blades of a centrifugal compressor with the Immersed Surfaces Technique. In this section the black line represents the solid boundaries ($\phi = 0$) and each cell is colored according to the value of ϕ.*

We proceed by showing the example reported in Figure 5.1, consisting of the flow inside a centrifugal compressor. The solid is first embedded into a Cartesian grid and for each cell we define a distance function ϕ. The external boundaries, exactly like we do for multi-phase flow, will be described by the zero level set of this distance function. The only difference is that, except for moving wall, this level set function is constant and we do not solve any equation for it. The treatment of viscous shear at the solid surfaces is handled very much the same way as in all CFD codes, where wall indices are calculated using the level set function.

All that is needed to run a simulation with IST in TransAT is a Level Set input file containing for each cell the distance from the closest objects with the index of that object.

5.3 TransAT-MB

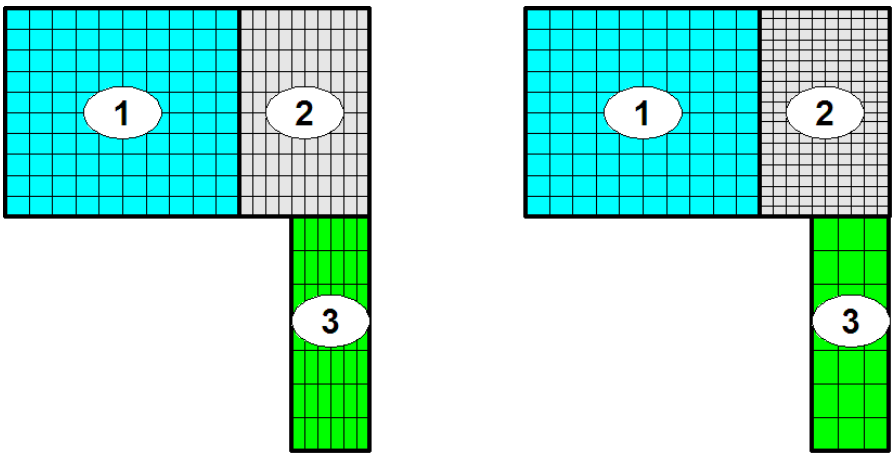

Figure 5.2: *Mesh with three blocks connected with (a) matching connections (b) non-matching connections*

TransAT-MB is the multi-block version of TransAT (developed in [3]). It has the same features of TransAT but it can handle multi-block grids and it uses MPI to distribute blocks between many processors.

The first version of TransAT-MB was created to manage only "matching connections" between blocks. Figure 5.2 shows the difference between matching and non-matching connection. In the second one we can have a ratio of refinement

between two adjacent blocks meaning that the each cell of a block can be linked to two or more cells of the other block.

In the first part of this thesis we worked on TransAT-MB to manage also non-matching connections (see 6.1).

Chapter 6

TransAT BMR

In this chapter we summarize the first part of my work developing a *Block-based Mesh Refinement (BMR)* and improving the mesh generator *TransAT-Mesh*. In the first section we will explain the idea of the BMR and his implementation on the multi-block solver TransAT-MB. The features and the implementation details of the mesh generator are then presented in the second section while in the third section we show some applications of BMR.

6.1 Block-based Mesh Refinement

The idea of BMR was to extend the multi-block solver TransAT-MB in a way similar to the method explained in 4.3.1, using grid with a hierarchy of blocks to have local refinements in some zones (e.g. the zone near the objects or in the regions where we want a better accuracy). This requires to

1. find a numerical method to solve NS equations on grids with "non-matching" connections between blocks and with refined blocks inside;
2. modify TransAT-MB to manage this new type of grid;
3. implement an automatic, fast and simple way to create the grid for every complex geometry.

The last issue will be analyzed and solved implementing new features in the mesh generator TransAT-Mesh (see 6.2) so let us focus on the first two.

6.1.1 Composite grids

A *composite grid* is the union of several grids (generally uniform) with different grid sizes that cover different part of the domain. In our case we use non-uniform orthogonal grids but to simplify the notation we suppose to have uniform grids.

We start from a global coarse grid Ω^H over the domain Ω, with meshsize H, chosen in agreement with the relatively smooth behavior of the solution outside the refined zones. In a region $\Omega_l \in \Omega$, such that it contains the part(s) of Ω in which relatively high resolution is needed, we have a local fine grid Ω_l^h with meshsize h. We chose the meshsizes so that $\zeta h = H$ where the *refinement factor*

$$\zeta = \frac{H}{h} \tag{6.1}$$

is an integer. The interface between the global coarse grid and the local fine grid will be denoted by $\Gamma \doteq \partial\Omega_l \setminus \partial\Omega$.

This refinement stategy is known as local grid refinement and we want to find an approximation solution on the composite grid using the finite volume formulation of TransAT. To do this we use a method similar to the *Local Defect Correction (LDC)* method, developed in a finite difference formulation for elliptic problems in [10].

6.1.2 Local Defect Correction

In this method, which is an iterative process, a basic global discretization is improved by by local discretizations defined in the subdomains. This update of the coarse grid solution is achieved by putting a defect correction term in the right hand side of the coarse grid problem. At each iteration step, the process yields a discrete approximation of the continuous solution on the composite grid. the discrete problem that is actually being solved is an implicit result of the iterative process.

Basically, the LDC iteration consists of the following steps:

1. Solve a *global coarse* grid problem with given right hand side.

2. Solve a *local fine* grid problem with artificial Dirichlet boundary conditions on the interface Γ.

3. Compute a *defect correction* term for the right hand side of the coarse grid problem and go to 1.

Suppose we want to solve the following problem, written in an integral formulation

$$\frac{\partial}{\partial t}\int_V \psi d\Omega + \int_{\partial V} \mathbf{f}\cdot\mathbf{n} d\gamma = \int_V s d\Omega \tag{6.2}$$

with the desired boundary conditions on $\partial\Omega$ and initial condition. $\mathbf{f} = \psi\mathbf{u} + \lambda\nabla\mathbf{u}$ is the sum of diffusive and convective fluxes, $\mathbf{n}$ is the outward normal, $V \subset \Omega$ is a generic control volumes and ψ is a generic quantity (for example velocity in a specified direction).

After the discretization on Ω^H, in a finite volume, we find an approximate solution ψ_H so that

$$\nabla^t T^H(\psi_H) + F^H(\psi_H) = S^H(\psi_H) \tag{6.3}$$

where T^H, F^H and S^H are respectively the approximation of the integral of ψ, of the fuxes and of the source term and ∇^t is the time discretization operator. T, F and S without indices represents the exact integrals.

In the first step we advance in time problem (6.3) and we obtain a corse solution ψ_H^n at time n. Then, in step 2, we solve the same problem in the refined region Ω_l using as Dirichlet boundary condition the coarse solution restricted to the boundary $\psi_H^n|_{\partial\Omega_l}$. In this way we obtain a fine solution ψ_h^n.

At this point we have three different solution: the *coarse* solution ψ_H^n, the *fine* solution ψ_h^n defined only on Ω_l^h and the c*omposite* solution

$$\psi_{H,h}^n = \begin{cases} \psi_H^n & \text{on} \quad \Omega^H \setminus \Omega_l^h \\ \psi_h^n & \text{on} \quad \Omega_l^H \end{cases} \tag{6.4}$$

The composite solution is used to calculate the *defect correction term* $\tilde{d_H}$. This is an approximation of the residiual of (6.3) for the coarse grid that is

$$d_H = \nabla^t T^H(\psi) + F^H(\psi) - S^H(\psi) \tag{6.5}$$

where ψ is the exact solution of (6.2). This can be rewrited as

$$d_H = \nabla^t(T^H(\psi) - T(\psi)) + (F^H(\psi) - F(\psi)) - (S^H(\psi) - S(\psi)) \tag{6.6}$$

Now, we approximate the exact solution with the composite solution and the exact operators (for example F) with

$$F^{best} = \begin{cases} F^{sum} & \text{on} \quad \Omega^H \setminus \Omega_l^h \\ F^H & \text{on} \quad \Omega_l^h \end{cases} \tag{6.7}$$

where F^{sum} is the sum of the fine grid approximation fluxes. This yields

$$\begin{aligned} d_H \approx \nabla^t(T^H(\psi_{H,h}) - T^{best}(\psi_{H,h})) + (F^H(\psi_{H,h}) + \\ - F^{best}(\psi_{H,h})) - (S^H(\psi_{H,h}) - S^{best}(\psi_{H,h})) = \tilde{d_H} \end{aligned} \tag{6.8}$$

Finally we can go back to step 1 adding the defect correction term in the right hand side of equation. This three steps must be iterated k times to reach a convergence. There are many theoretical results on the convergence of LDC method (for example [10]). What is important to note is that, if the process converges, the limit solution $(\bar{\psi}_H, \bar{\psi}_h)$ satisfies

$$\bar{\psi}_H|_{\Omega_l^H} = \bar{\psi}_h|_{\Omega_l^H} \tag{6.9}$$

meaning that the coarse grid solution in the refined domain is equal to the fine grid solution.

When the BMR levels (the levels of refinement) are more than two it's sufficient to repeat recoursively step 2 until we reach the last level and to calculate the composite solution taking into account all the levels. In most situations the process converges after a number of time step equal to the number of levels plus one.

6.1.3 Fine/coarse grid interplay

In the process briefly described above, we have to pass information from the coarse grid to the fine (in step 2) and then going back to the coarse grid (when we

calculate the composite solution). The first passage can be done with interpolation and the second one with averaging.

Figure 6.1 shows a coarse grid and a fine grid (with refinement factor of two in x-direction and three in y-direction) with the stencils needed to compute the numerical fluxes. When we solve the fine grid problem we need the fluxes at the boundary cell (blue cell number 2). These values are calculated using the stencils represented by the triangles. Coloured triangles represents the points in which it is needed the interpolation using the coarse values. To calculate the composite solution instead we have to average over all the fine cells forming a coarse cell (in figure 6.1 the averaging is done for the coarse cells marked with coloured circles using 6 fine cells).

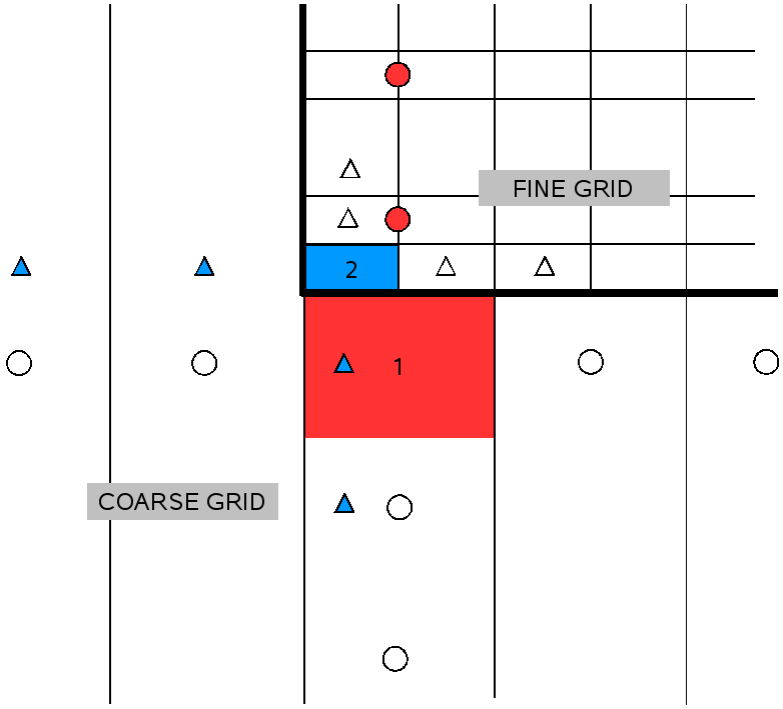

Figure 6.1: *Stencils to calculate convective fluxes in the coarse (cell 1) and in the fine (cell 2) grid. Coloured triangles are the interpolation points, coloured circles are averaging points.*

6.2 TransAT-Mesh

I started working on this application in October 2007 continuing the work done by Marcel Birrer ([4]). The result was an improved version of TransAT-Mesh, now also called TMB (TransAT-Mesh BMR).

This program has a graphical user interface (GUI) and a mesh generator that could read geometry files (GTS or STL format) and create the files that have to be given to the TransAT solvers as input. The main improvement between TransAT-Mesh and TMB is the possibility to create automatically multi-block and BMR grids, modify manually them if necessary and then write the output files needed by the multi-block solver TransAT-MB which are different from the ones for the single-block solver TransAT.

For single-block grids four files are created: the grid file containing information about the cell corners, the boundary conditions file, the properties file containing the physical properties of the model and the level set file needed for the IST method (see 5.2). For multi-block grids two more files are needed: the connectivity file containing the connections between the blocks and the blocks distribution file that assign each block to a processor. In both cases the program can also create an input file for Tecplot to visualize the mesh and the level set function for IST.

When the program starts the user has to choose if he wants to create a multi-block (and eventually BMR) or single-block mesh. If the user chooses to work in single-block mode, then the BMR tab is deactivated. In the following pages we will explain all these features putting in evidence especially the ones implemented from scratch by me during my internship (for example all functions concerning BMR and multi-block).

Figure 6.2 shows the GUI of TransAT-Mesh, which contains a 3D-viewer showing a preview of the mesh on the right side and on the left there are five tabs, where all the properties for the grid can be set.

Libraries used

This application is written in C++ with help of several libraries. Below all libraries used in the code are given.

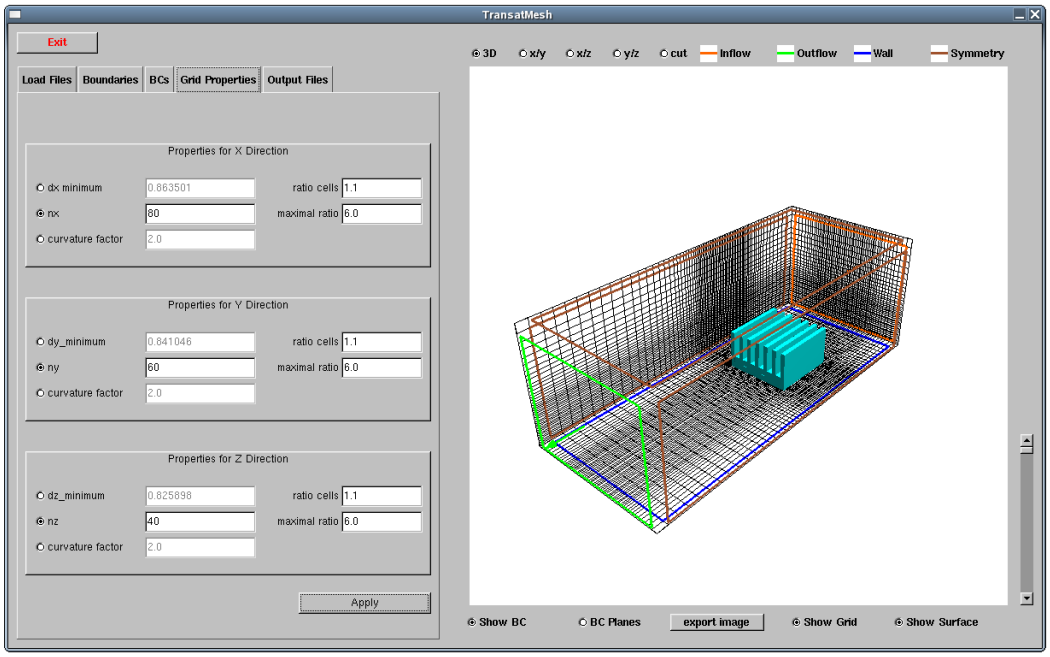

Figure 6.2: *TransAT-Mesh.*

- **GTS :** the core functions of this application include surface handling. There exist many file formats to represent CAD geometries. The most used is the STL format. TransAT-Mesh uses GTS (GNU Triangulated Surface Library), a library that contains useful structures to represent surfaces and functions to deal with these surfaces, including the possibility to read from STL files. A GTS surface is just a structure containing various faces (triangles), which not necessarily need to be connected together. A group of connected triangles is a "connected surface". Instead, a closed surface is a surface in which each edge of the surface belongs to two triangles. A three dimensional surface of a solid, which contains no wholes could still be a not closed surface.

- **GLIB and Standard Template Library:** basic libraries used for data storage and manipulation.

- **Fltk :** the GUI was created with fltk, which stands for “fast and light toolkit”. Fltk provides also a connection to OpenGL.

- **OpenGL :** 3D library used for the preview window.

- **Libtiff/Libjpeg :** below the main preview window is a button to export the preview as tiff or jpeg file. Therefore these two libraries were linked to the code.
- **GNU Make** : used to compile the source code.

Load Files

In the first tab there are all the functions to load files and projects. When a file is loaded it checks if it is a valid STL, or GTS file and reads the data to a surface. Now the surface gets split up into connected surfaces, if necessary, and checks each one of them, if it is closed. In case this check fails, there are the options to load the file as it is, try a simple closing algorithm or to "thicken" the surface to obtain a new closed surface. This thickening option is necessary when you want to create a grid for a plate or a thin object represented just as a surface in the input file. After the loading, a grid with default values is created and then it can be modified in a next step. In the first tab there is a button to open the properties window.

Properties window

The physical properties (density, heat capacity, thermal conductivity, surface roughness and internal heat source) of each object can be specified. For problems not involving heat transfer default values are used. Each object can be set to invisible or “disable”. The first feature is useful to show objects that are one inside the other. Disable objects instead are not taken into account for the refinement (see Boundary tab and BMR tab) and when the objects are exported into a new gts file.

Boundaries

This second tab gives the possibility to set the positions of the domain boundaries and choose the type of refinement (around edges or over the whole body, see figure 6.3) for the father grid.

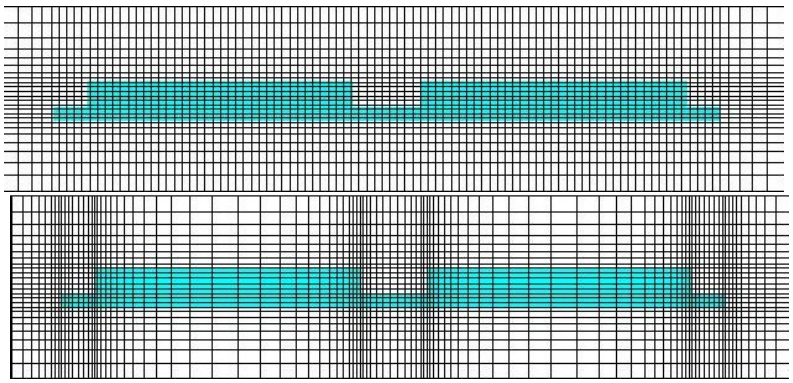

Figure 6.3: *Father grid for an electronic device. (a) refinement over whole objects (b) refinement over edges.*

Boundary Conditions

TransAT can handle a variety of different boundary conditions and set multiple conditions for each boundary. These are defined in a separate file with the suffix ".bc". The current version of TransAT-Mesh writes this boundary conditions file together with other files used as input for TransAT (see Output files tab). Currently only one condition for each boundary can be set. TransAT-Mesh offers a choice of four of the conditions that TransAT can handle, which are inflow, outflow, wall and symmetry. Inflow and wall need additional settings to be done which are temperature, type of profile and bulk velocity for the inflow, and temperature or heat flux at the wall. at an inflow there is also the special option to set a fan, used for electronic device cooling.

Grid Properties

In this tab you can choose some properties concerning the father grid. First you have to choose between setting the minimal cell size, the number of cell corners or an option to set your minimal cell size depending on the maximal curvature of the surface. The function to calculate this curvature is taken from the GTS library, but does not give the desired results for every file. Therefore this option might be modified in a later version. Further you can set the ratio between two neighboring cells and the maximal ratio between smallest and biggest cell.

Output Files

The tab "Output Files" contains some options to save a project or to save a grid, build the output file and run TransAT. When you write the files you can choose to reverse the level set value. This is necessary when we want to solve an internal flow (e.g. to create a pipe you can load a full cylinder and reverse level set to obtain an empty pipe. Another important new feature is the possibility, when you have created more than one blocks, to delete the first grid and keep only the refined grid around the objects ([7]). There is also an option to write all the surfaces loaded (except those disable) to a GTS file, this is useful also to convert an STL file to GTS format.

BMR

In this tab, shown in Figure 6.4, we can split the grid into a multi-block one and set up block-based mesh refinement (BMR). For a non-BMR multi-block grid you can only split (and eventually shrink it around the objects) the initial father grid and distribute the blocks into many processors optimizing the processor load. Instead for a BMR grid there are other options. You need some starting parameters for the initialization and then there are some tools to modify the resulting grid as needed.

The starting parameters are:

- *BMR levels:* the number of refinement levels.
- *Ratio level 2:* in the BMR the refinement is done starting from the existing grid and splitting his cells into n parts for each direction (i.e. n^3 new cells for each old one). Here you can specify the ratio n between the initial grid and the second level grid. This ratio can be also modified (also in a different way for each direction) after the initialization.
- *Ratio level 3-4-5:* same as above for the following levels.
- *Reynolds number:* is the approximate Re of the simulation used to calculate the dimensions of the new grids so that the last level grid covers the boundary

layer approximately calculated using the following expression:

$$\frac{\delta}{L} \approx 4,92Re^{-\frac{1}{2}} \quad \text{for} Re < 500 \qquad \frac{\delta}{L} \approx 0,37Re^{-\frac{1}{5}} \quad \text{for} Re > 500$$

where is δ is the boundary layer length and L is the length scale of the object. These are just the initial dimensions of the grids that can be modified later on.

- *Factor:* you can specify a factor to modify the size of the grid calculated using the boundary layer estimation.

- *Automatic ratio* and *Max ratio:* if Automatic ratio is active the last level refinement ratio is not given by the number specified before but automatically calculated so that there are enough cell covering the boundary layer. This number can be limited by the Max ratio.

- *Initialize BMR:* this button initialize the BMR framework and build automatically the BMR grids. If there is at least one object loaded, TransATMesh will create new refined grid around it. To exclude an object from the BMR refinement or to reduce the refinement levels just for a single object disable it or specify a maximum BMR level in the properties window.

Once the BMR is initialized, all the other tools can be used. In particular:

- *Add refinement zone:* usually the refinement is needed just around the objects so the BMR grids are placed by default just around them. Here you can add a refined grid where is needed. When the button is pressed, a new window with the coordinate of the 6 corners of the zone will appear. You can specify in a simple way selecting a projection view in the 3d-viewer (xy, yz or xz), pressing the *Set with mouse* button and clicking twice on the 3D viewer to select the minimum and the maximum of the 2 selected coordinates. To add in this way also the third coordinates, repeat the same procedure with a different plane. For each refinement zone added also a maximum level of refinement must be specified. When all is done press *Save* to build the new grids for the selected zone.

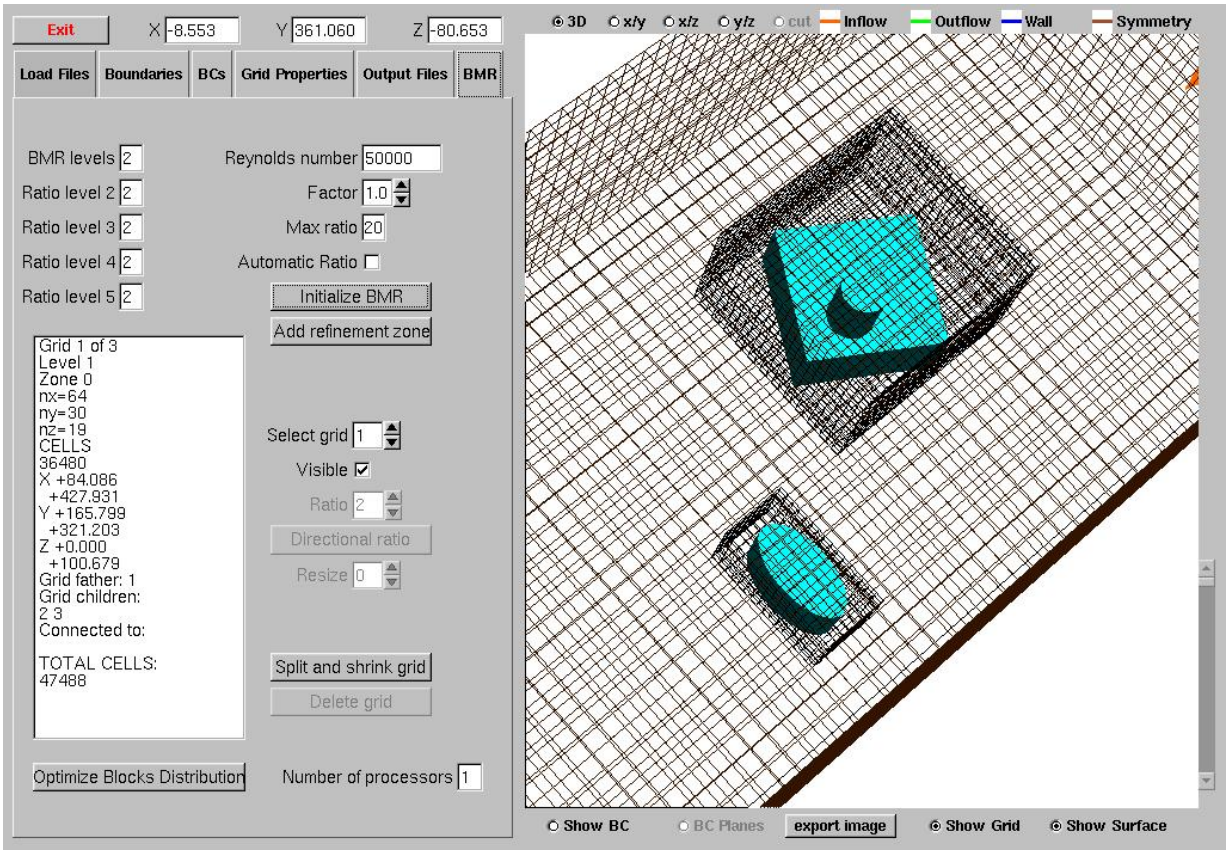

Figure 6.4: *BMR tab in TransAT-Mesh*

- *Select grid:* go through all the grids (all are identified by an index), the selected one is showed in a different color. The selected grid can be modified in different way as explained below
- *Visible:* makes the selected grid visible/invisible
- *Ratio:* modifies the refinement ratio
- *Directional ratio*: pressing this button one can choose a different ratio for each direction
- *Resize*: using the arrows the grid is resized
- *Add son grid:* add a new grid inside the selected one by-passing all the restriction imposed in the main parameters
- *Split and shrink grid:* using this button one can split a grid in r parts where $r = r_1 r_2 r_3$ and r_i is the number of parts in the $i-th$ direction. Then one can choose to shrink this new grids keeping only the cells closest to the object (all the cells inside the approximate boundary layer). In this case a factor that can modify this length must be specified. If $r_1 = r_2 = r_3 = 1$ the grid is not split but can be just shrunk
- *Delete grid:* delete the selected grid

At the bottom of the page there are some tools to distribute the grid on different processors. *Optimize blocks distribution* button takes the number of processors as input and gives each grid to a processor to optimize the speed of the simulation. If the solution is not good enough (the ratio between the most loaded and less loaded processor is bigger than 1.2) the grids are automatically split. A better result is obtained if the grid is split using the Split and shrink button according to the number of processors (e.g. for an 8 processor parallel job and a mesh with only one grid, one can use a 4 by 2 by 1 splitting or 2 by 2 by 2).

3D Viewer

The 3D graphics viewer on the right shows the mesh and the surfaces in 3D perspective or alternatively in 2D planes chosen by the user. An option called

cut mode let the user view just the intersection between the grid planes and the objects. An image of the mesh can be exported in either JPG or TIFF formats.

Batch Mode

TransAT-Mesh can be also run in batch mode (without initiating the GUI). An existing project file is needed for this. If using a single–block mode project file, one can execute as:

```
prompt> transatmesh project.stt
```

For a multi-block or BMR project file, use:

```
prompt> transatmesh project.stt -mb
```

6.3 Examples and applications

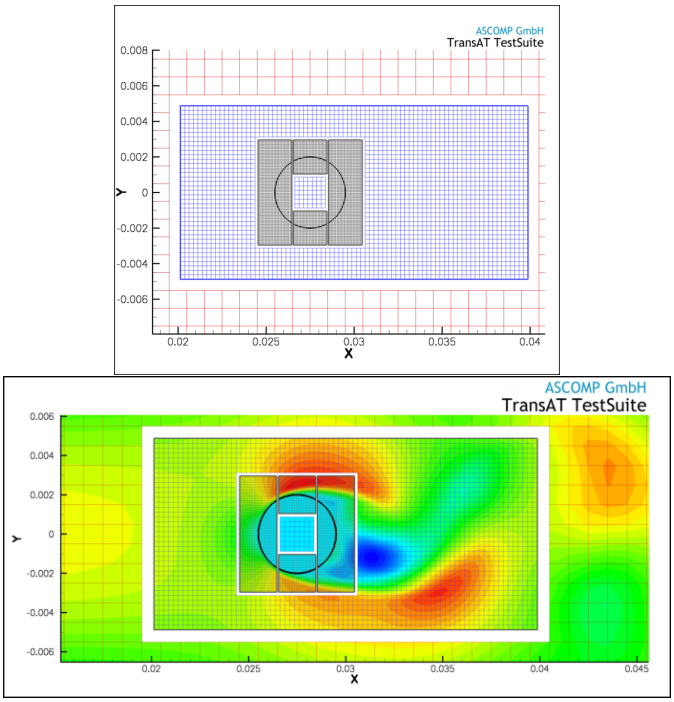

Figure 6.5: *Example of a BMR mesh for circular cylinder and flow solution*

BMR has been tested through many test cases and applications. Here we present some examples and we refer to chapter 9 for detailed simulations and analysis.

An example of a BMR mesh for a circular cylinder is given in Figure 6.5. This simulation use 6 blocks and 3 level of refinement. A similar grid for an unsteady LES simulation of flow past a three-dimensional sphere is given in Figure 6.6. In this case there are 29 blocks and 800.000 grid points. Unfortunately while I am writing this thesis, this simulation is still running and the results are not completed.

In [7] we show some microfluidic applications where the refined blocks can be used in a simulations of a micro-reactor to have a better accuracy where the fluids are mixing.

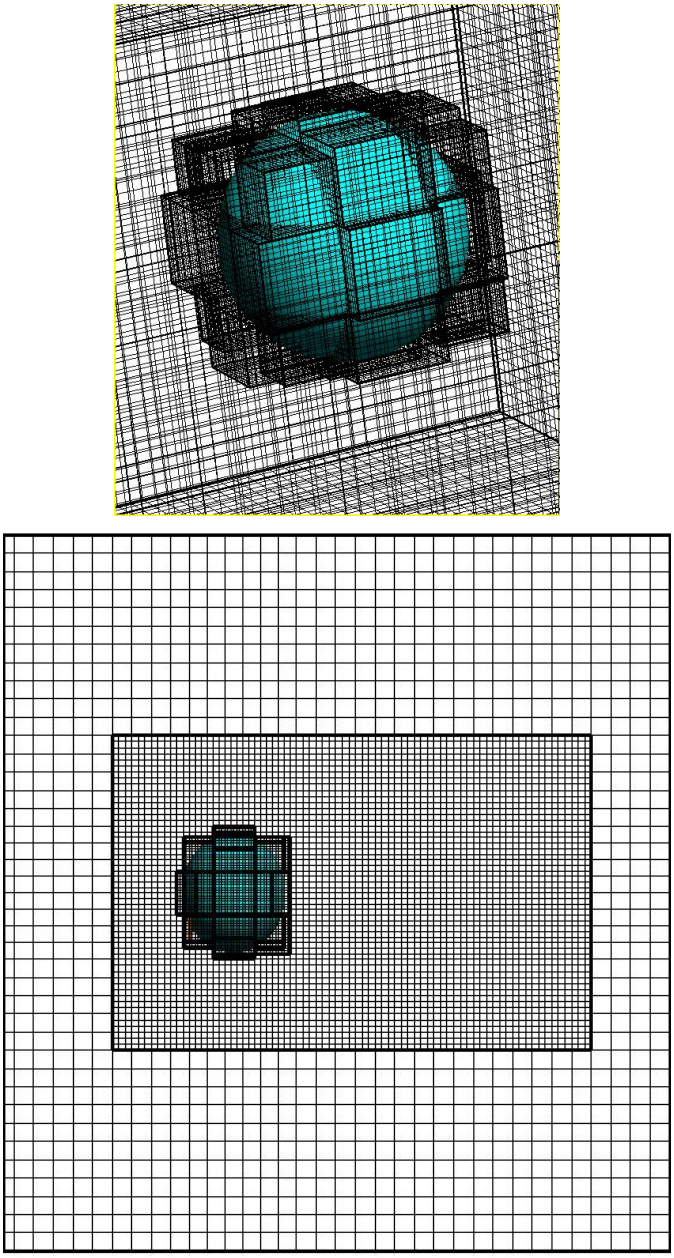

Figure 6.6: *Example of a BMR mesh for the flow past a sphere.*

A more common field of application of BMR is wind engineering. In Figures 6.7 and 6.8 we can see an example of grid and the resulting flow (LES calculation) for a geometry composed by three buildings of Mulhouse (France).

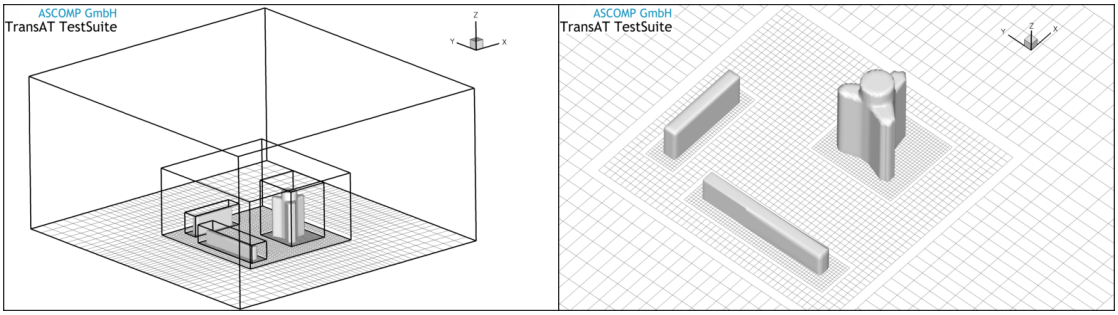

Figure 6.7: *BMR applications in wind engineering. Mesh with 3 level and 6 blocks.*

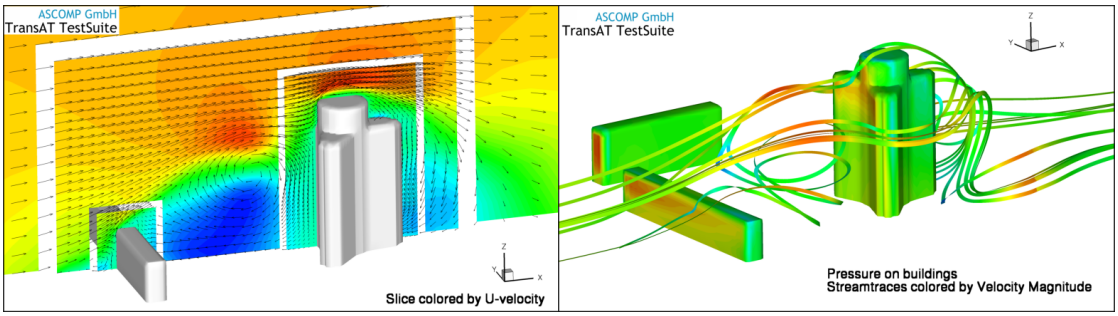

Figure 6.8: *BMR applications in wind engineering. Resulting velocity and stream lines.*

The numbers of blocks can be also very high to cover complicated geometry with refined blocks as in the example of Figure 6.9 where we built a mesh for a racing car. In this example, with approximately four millions grid points and more than four hundred blocks, we can see that with this kind of approach (BMR coupled with the Immersed Surfaces Techniques, for detail on IST see 5.2) we can use Cartesian grid also for complex geometry and save more than half cells with respect to a single block grid without losing the accuracy and the simplicity of Cartesian grids. In this case the number of blocks is very high because we want to preserve a good resolution for the boundary layer close to the car without having many cells inside the car where they would be completely useless. TransAT-Mesh is able to build automatically grids with the "Split & Shrink" option.

It is important to note that what we can see in Figure 6.9 is the effective resolution of the car boundaries using IST. In fact the surface representing the car is the zero level set of a distance function ϕ defined in all the domain. This

function calculated by TransAT-Mesh on each cell will be used by TransAT-MB to impose the boundary conditions on the car surface. However in this example the resolution is still not sufficient because we can see some holes in the aileron surface due to the difficulty of capturing thin surfaces. This can be achieved for example refining more the blocks containing these surfaces (for example modifying the ratio of refinement or adding other blocks).

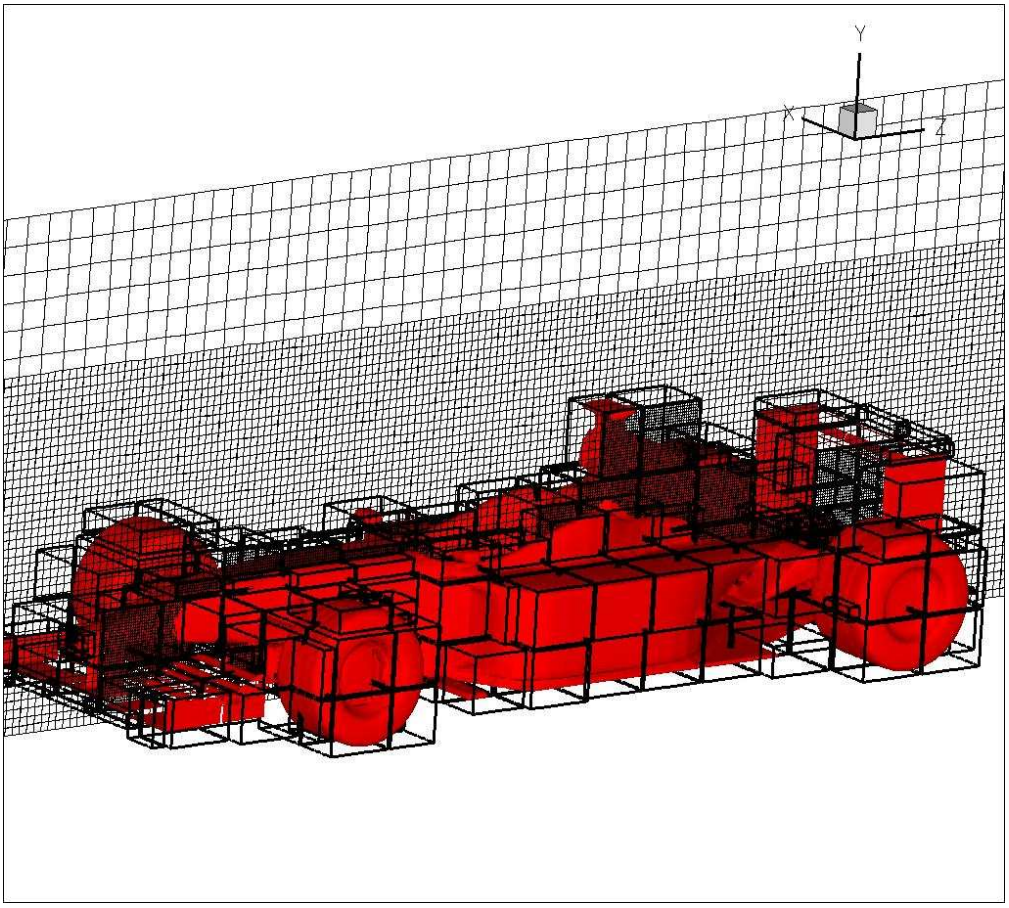

Figure 6.9: *Three-level BMR grid for a Formula 1 car.*

Chapter 7

TransAT AMR

In this chapter we present the AMR implementations and algorithms implemented in TransAT during the second part of my work. I started working on this second subject in February 2008 and the development is still in progress.

7.1 Refined Level Set Method

As we explained in chapter 2, the drawback of the level set method is that it does not inherently preserve the volume of the fluids on either side of the interface. This error results in many important applications and can cause big problems to the final result of multi-phase flow simulations. This volume error is proportional to the employed grid spacing so if we want to cut down the volume error we should refine the grid everywhere or with AMR techniques. However in many applications (e.g. when we have primary atomization of liquid jets and sheets) we don't really need a refined grid for all the equations because it would not be efficient in cases in which a fine grid is required over large portions of the interfaces.

So the first aim of our work on AMR is focused on solving only the level set equation on a separate, refined grid, so that we can cut down the volume error without modifying the other equations and adding too much computational efforts. This method, used in [13] and [17], can be summarized up in the following points:

- locate the interface and refine the grid around it using some criterions and satisfying some specific constraints;

- interpolate all the variables needed using the coarse grid to obtain all the values in the refined grid;
- solve the level set equation (the advection and the reinitialization) in the refined grid;
- calculate the new values in the coarse cell averaging the refined cells values;
- unrefine the unnecessary cells

This procedure must be done for each time step although the refinement and derefinement could be done every k time steps (where k can be constant or can varies depending, for example, on the velocity of the interface).

The level set values obtained with AMR can be then passed to TransAT solver and used inside the other equations, for example in the momentum equation to get a better approximation of the surface tension. Another possibility is to calculate the surface curvature in the AMR grid and directly pass it to TransAT.

Before going on let us look deeply at the method implemented.

7.2 Octree grid

A generalized binary tree structure (called quadtree in 2D and octree in 3D) is used to refine the mesh starting from the original TransAT coarse grid. To simplify notation we suppose a uniform Cartesian grid, however the AMR implementation could work also for non-uniform Cartesian grids.

We start from the level set value ϕ on the coarse grid (level 1) and refine this grid splitting recursively each cell into 8 cells. At the l^{th} level of the tree, each node i, represents a cube cell with sides of length $dx_i = h/2^l$ where h is the cell distance in TransAT original mesh (which can be eventually different for each coordinates). We assume, according to TransAT code and finite volume formulation, that the level set value is stored at the centers of mass of the nodes at the finest level of the tree, also known as the leaves. An alternate storage location could be the vertices of the cells. We choose the centers because of the simplicity of implementation (e.g. there are no storage point belonging to multiple cells), and the fact that local interpolation can be done in a dimension independent way.

We do not allow side length ratio > 2 or < 0.5 between neighbor cells. We consider also the neighbor in the diagonal direction to simplify the interpolation and flux calculation. This restriction results in what is known as a *balanced octree*. This balancing can be obtained by following the criterion of refining cell whose distance to the interface is less than a constant times its edge length. In practice the criterion used to refine, similar to that used in [16], is given by

$$|\phi| < \epsilon + (1 + \sqrt{3}/2)dx \tag{7.1}$$

whereϵ is a reference fixed distance that can be used to increase or decrease the refinement band around the interface. We can include in the criterion also information about the advection scheme or the advection velocity so that the advection equation is solved with an higher accuracy just where is needed. This can be done taking into account ϵ as a dependent variable.

If we use some some other criteria (for example given by other properties of the flow) we have to impose externally the constraints to have a balanced octree (see for example figure **??**).

Starting from level 1, when the criterion is satisfied the cell is split like in figure 7.2 creating eight 2nd level cells and in this cells we need to interpolate all the variables. In this case the most important one is the level set value ϕ. This value is calculated using the interpolation methods described in the following section.

When this procedure ends all the split cells forms the 2nd level grid while the unsplit cells are still the original 1st level cells. This refinement procedure is repeated *l-1* times where l is the maximum level of refinement that can be specified as a parameter or can be estimated in different ways trying to control the accuracy of the solution and the computational efficiency.

The derefinement procedure is governed by the same criterion. When eight refined cells forming a cell in the upper level don't satisfy the criterion, they are deleted and they're averaged value is copied to the father cell.

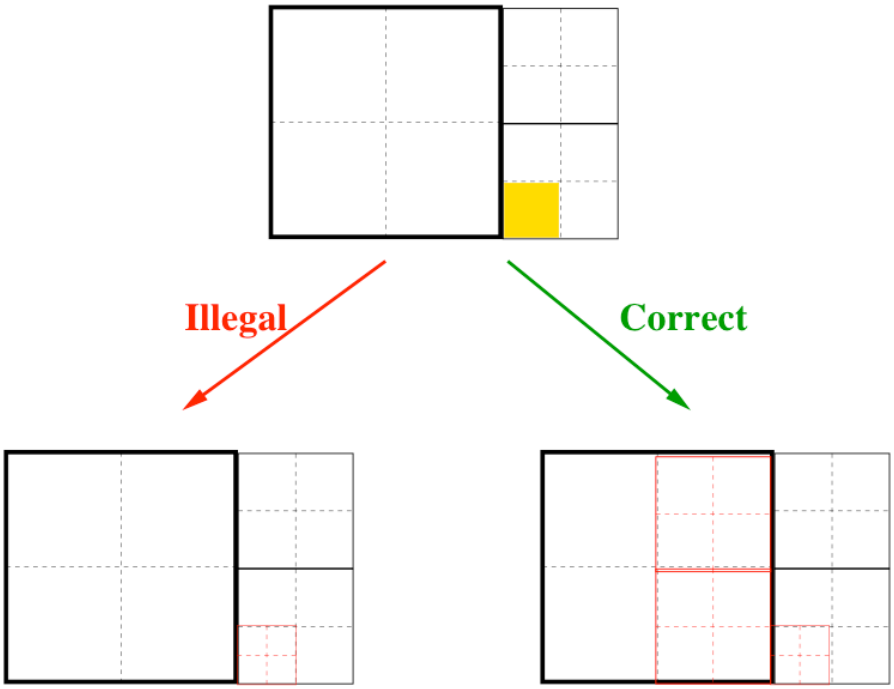

Figure 7.1: *Constraint for refinement.*

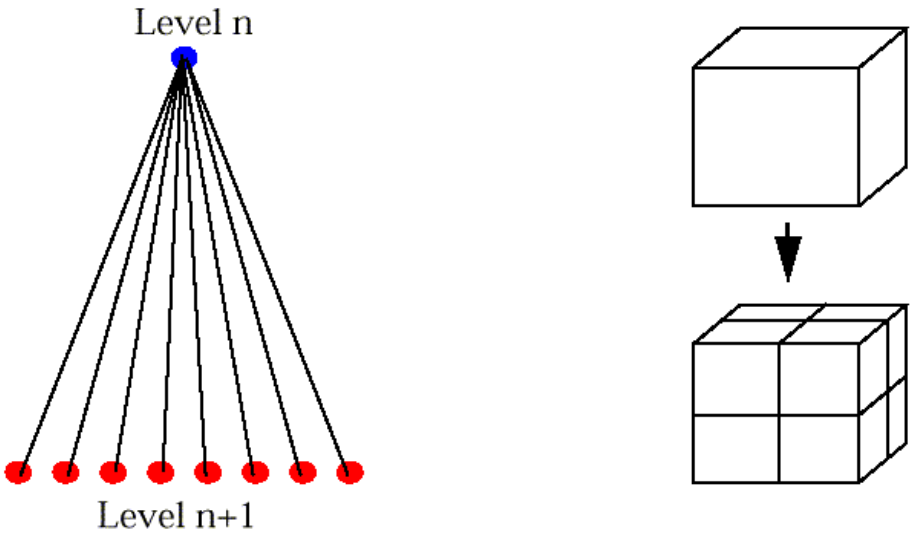

Figure 7.2: *Example of cell splitting*

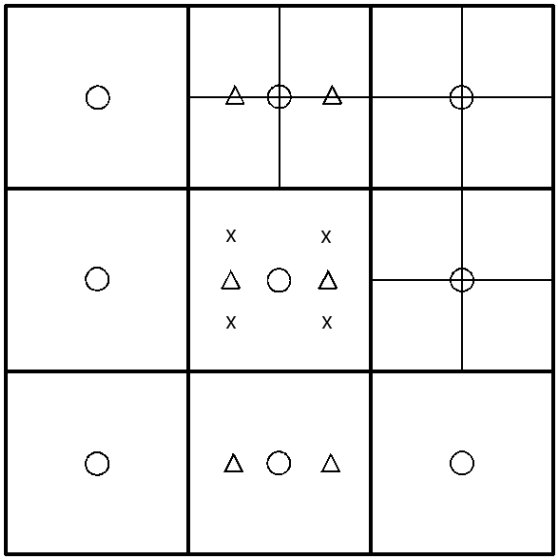

Figure 7.3: *Bi-quadratic interpolation in 2D.*

7.3 ϕ interpolation

When a cell is split the level set value of the new cells must be recovered in some way starting from the existent cell. There are two possibilities: interpolating using the upper level grid values or interpolating using the neighbor cells whether they are of the same level or of a different level.

Since all the cells of a certain level form a Cartesian grid, the first choice for interpolation are some simple schemes like the tri-linear (bi-linear in 2D) or tri-quadratic (bi-quadratic in 2D) interpolation where each direction is treated separately. Figure **??** shows the bi-quadratic interpolation in 2D, the extension in 3D is trivial.

The interpolation in each direction can be easily computed through Neville algorithm which can used to obtain 1D interpolation of the desired order.

Another possibility, is to recover information from the finest level possible. This means that sometimes we have to use cells of different levels so we have to interpolate in a unstructured way. This can be done linearly or with quadratic precision solving each a linear system. Figure This gives a better result but is more expensive than the previous one.

When the father grid is uniform is possible to perform a third type of interpo-

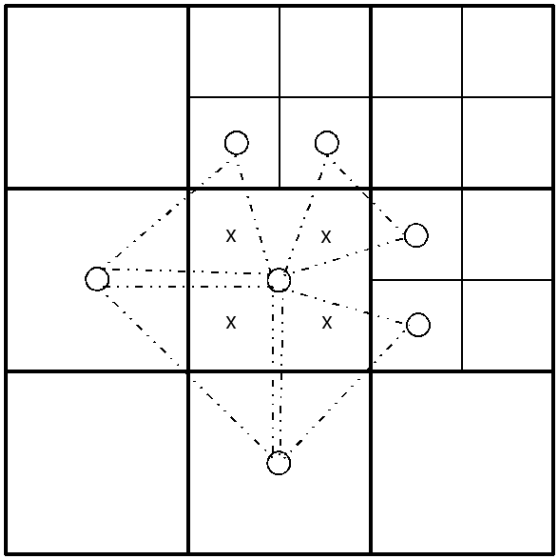

Figure 7.4: *"Unstructured" interpolation in 2D (linear).*

lation (as proposed by [18]) using the diagonal direction.

The last very simple solution is the *straight injection* which means that we simply copy the values from the father cell. In our code is possible to choose the desired interpolation procedure modifying an input parameter.

7.4 Velocity interpolation

In the Refined Level Set Method we store and solve on the adaptive grid only the level set equation. So only the level set values will be available in every cells while the velocity must be recovered from the first level grid. This same interpolation however is needed also for fully AMR code (with all the variable stored and solved in the octree grid) when a cell is refined. In our case instead, we have to recursively repeat this procedure for all the levels starting from level 1. The velocity obtained are necessary to calculate the fluxes of ϕat the cell faces and can be done in four steps:

1. interpolating the external velocity (i.e. the fluxes relative to a face which is at the boundary of the father cell);

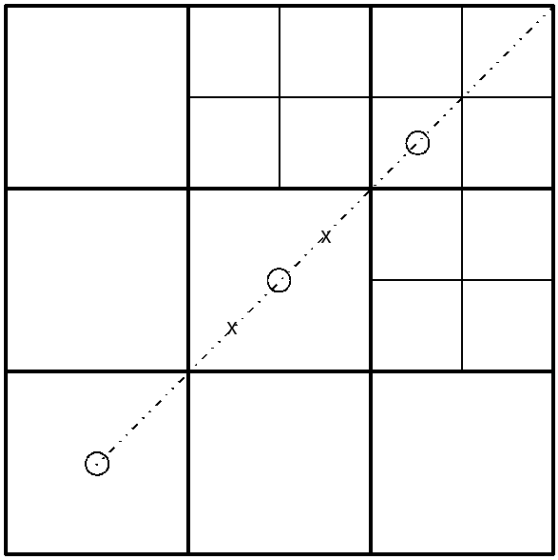

Figure 7.5: *"Diagonal" interpolation in 2D.*

2. correcting the external velocity to conserving the mass (if the flow is incompressible);

3. interpolating the internal velocity;

4. correcting the internal velocity.

In the first step the velocity coming from the father cell are interpolated to the external faces of the new octal structure. After the interpolation these new velocities must be corrected to satisfy the following relation on each face f, split into $f_1, ..., f_p (p = 2$ in 2D and $p = 4$ in 3D)

$$\sum_{i=1}^{p} V_{f_i} - pV_f = 0 \tag{7.2}$$

which means that the coarse velocity must be the mean of the fine velocity. If this expression is not null, to correct the fine velocity, we simply subtract the residual divided by p to each fluxes. This correction is necessary to keep the field solenoidal but can be avoided if the velocities are simply copied to the fine grid but this reduces the accuracy level set advection.

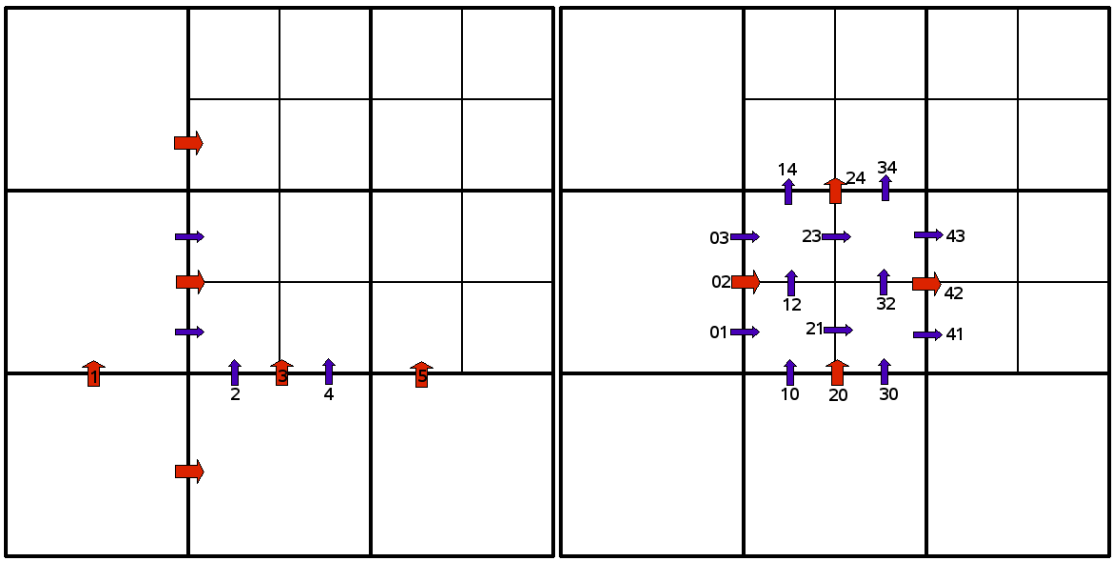

Figure 7.6: *Velocity interpolation. (a) external velocity (b) internal velocity.*

The third step consists in interpolating these velocities to the internal faces of the octree. Also after this interpolation is necessary another correction step to keep the field solenoidal i.e.

$$\sum_{i=1}^{s} V_{f_i} = 0 \tag{7.3}$$

where s is the number of the faces in each cell (6 in 3D and 4 in 2D).

We can see a 2D example of these steps in figure 7.6. The first two steps are equivalent to solve the following constrained minimum problem

$$\arg\min_{v_2,v_4}(v_2 - \bar{v}_2)^2 + (v_4 - \bar{v}_4)^2 \text{ with } \frac{(v_2 + v_4)}{2} = v_3 \tag{7.4}$$

where the indexes are indicated in the left part of figure 7.6 and $\bar{v}_i$ are the velocities obtained from quadratic interpolation using v_1, v_3, v_5.

The last two steps instead are equivalent to (see the right part of figure 7.6)

$$\arg \min_{v_{12},v_{32},v_{21},v_{23}} (v_{12} - \bar{v}_{12})^2 + (v_{32} - \bar{v}_{32})^2 + (v_{21} - \bar{v}_{21})^2 + (v_{23} - \bar{v}_{23})^2$$

$$v_{01} - v_{21} + v_{10} - v_{12} = 0$$

$$v_{21} - v_{41} + v_{30} - v_{32} = 0$$

$$v_{03} - v_{23} + v_{12} - v_{14} = 0$$

$$v_{23} - v_{43} + v_{32} - v_{34} = 0$$

where this time $\bar{v}_{ij}$ is obtained with linear interpolation using the previously calculated external velocity (e.g. $v_{12} = \frac{v_{10}+v_{14}}{2}$).

7.5 Level set advection

The level set equation is a passive advection equation of a scalar quantity ϕ. In a finite volume formulation, using the first order Euler approximation for time derivative yields

$$\phi_{i+1} = \phi_i + \frac{\Delta t}{\Delta V} \sum_k f_k \tag{7.5}$$

where V is the volume of the cell, $f_k = \int_{S_k} V_k \phi_k dS_k$ is the numerical fluxes of ϕ on the faces k (east, west, north, south, top or bottom), V_k and S_e are respectively the normal velocity and the surface area of face k. ϕ_k can be calculated in many different ways depending on the advection scheme used.

7.5.1 Advection schemes

According to the schemes used in TransAT we implemented the following advection schemes:

- Central scheme.
- 1^{st} order upwind.
- QUICK scheme.

- HLPA scheme.

To build these schemes we need at most to move two cells away from the cell i in which we want to calculate the fluxes. So there are three possible situations:

1. The neighbor cells $i \pm 2$, $i \pm 1$ are of the same level of i. In this case we simply use the value of these cells.
2. The neighbor cells are split and contain finer cells. In this case we take an average of his 8 son cells.
3. The neighbor cells are one level coarser. In this case we have to interpolate ϕ in the point where it would be a cell of the same level of i.

7.5.2 Time scheme

For the time stepping scheme we implemented:

- $1^{st}$and 2^{nd} order Euler[1].
- $3^{rd}$and 4^{th} order explicit TVD Runge-Kutta.

7.6 Reinitialization

Reinitialization is needed because after each time step this function represents no more a signed distance. This is done, according to [21; 31] trying to set the norm of the gradient of ϕ equal to one solving the following equation

$$\frac{\partial d}{\partial \tau} = S(\phi)(1 - |\nabla d|) + c\Lambda_{i,j,k} f(\phi) \equiv L(\phi) + c\Lambda_{i,j,k} f(\phi) \tag{7.6}$$

with the initial conditions

$$d(x, y, z, \tau = 0) = \phi(x, y, z) \tag{7.7}$$

where $S(\phi)$ is a sign function, $\Lambda_{i,j,k}$ is given as

[1]With 2^{nd} order Euler we mean that, if $\frac{d\phi}{dt} = f(\phi)$, we use $\phi_{t-\Delta t}$and ϕ_t to calculate $f_{t+\Delta t/2}$ and then $\phi_{t+\Delta t}$

$$\Lambda_{i,j,k} = \frac{\int_{\Omega_{i,j,k}} H'(\phi) L(\phi, d) d\Omega_{i,j,k}}{\int_{\Omega_{i,j,k}} H'(\phi) f(\phi) d\Omega_{i,j,k}} \tag{7.8}$$

$$H'(\phi) = dH/d\phi \tag{7.9}$$

$$f(\phi) = H'(\phi)|\nabla\phi| \tag{7.10}$$

and $\Omega_{i,j,k}$is the grid cell. The last term in equation 7.6 is a mass correction to keep the volume of the two phases constant. This equation is solved until reaches the steady state and $|\nabla d| = 1$ near the interface and then ϕ is replaced by d^{steady}. For each pseudo-time step, approximately one cell away from the interface is set to the exact distance value.

The derivatives are calculated using a WENO (Weighted Essentially Non Oscillatory) Scheme ([28]) while Runge-Kutta scheme is used to advance in τ (pseudo-time). When we have to calculate the derivative for a cell which is at the boundary between fine and coarse grid, as explained in 7.5.1, we need interpolation or averaging.

7.7 Coupling with TransAT

The level set values calculated in the AMR grid are then passed to TransAT. This is done averaging over all the cells forming a "father" cell. TransAT uses this value in the momentum equation to calculate the surface tension term and the physical properties (e.g. density and viscosity) that depend on ϕ(see 2.2).

However a better approach would be to calculate these quantities directly in the finer grid and then passing the averaging quantities. In fact, there are many important variables that depend on ϕ.

- H function $H = (\phi > 0)$ or a smoothed version of it. Many choices are possible to smooth the interface discontinuity over a width ϵ.
- Density (and the other physical properties) $\rho = H\rho_l + (1 - H)\rho_g$.
- Normal $\mathbf{n} = \frac{\nabla\phi}{|\nabla\phi|}$.

- Curvature $\kappa = \nabla \cdot \mathbf{n}$.

To calculate these variables we can distinguish two different approaches:

1. Average ϕ^{fine} to the coarse cells obtaining an averaged $\overline{\phi^{fine}}$ and then calculate $f\left(\overline{\phi^{fine}}\right)$, where f is a generic function of the Level Set function.
2. Calculate $f\left(\phi^{fine}\right)$ and then average obtaining $\overline{f(\phi^{fine})}$.

when the interface has a complex shape with high curvature gradient inside a "father" cell this two approaches can be very different. In fact in general $\overline{f\left(\phi^{fine}\right)} \neq f\left(\overline{\phi^{fine}}\right)$.

7.8 Algorithms and data structures

The choice of a data structure to represent the tree is conditioned by the following requirements:

1. for any given cell, efficient access to neighboring cells.
2. for any given cell, efficient access to cell level and spatial coordinates.
3. efficient traversal of: all leaf cells and all cells at a given level

Khokhlov ([19]) presented a data structures with pointers, the so called *fully-threaded octree* which allows (1) and (2) to be performed in $O(1)$ operations (versus $O(\log N)$ for a standard pointer-based structure). Operations (3) are performed in $O(N \log N)$ using the standard pointer-based tree description (N is the number of cells traversed). In this structure, each cell has pointers to the sons, the father and the neighbors.

However, most of TransAT functions are coded with *FORTRAN90* so, to have a better compatibility, we prefer to use a pointer-less data structures. A good example of a linear octree implementation without pointers can be found in [20] where each cell is defined by an integer and a complex way of numbering cells is proposed so that the operations (1)-(2)-(3) can be done simply adding or subtracting a number to the cell index.

We implemented our data structure in similar way but we added more information to each cell to simplify debugging. Each cell in our octree is represented by two numbers: the level l and the index i. Furthermore it contains information about the father cell, his position in the father cell and his sons cells. So, at present operations (1)-(2)-(3) are not done very fast (although we do not have exact results) because in the first steps of the project, we was more interested on the numerical methods rather than speed.

Whatever structure may be used for the octree, many operations, like calculating the cell centers or the cell corners can be done recursively. In our implementation, we just have to call the desired function with the couple (l, i) as parameters (e.g. `get_center(l,i)`) and the function recursively calls itself until the first level is reached. At this point the value can be evaluated from TransAT father grid. If we use quadratic interpolation (see 7.3) recursion is necessary also to interpolate ϕ value.

The main characteristic of our algorithms is that, using recursion and interpolating function, we can use numerical schemes for Cartesian grid. Suppose we want to calculate the x-derivative in a the cell $(2, 5)$ using upward difference. We first have to calculate the east neighbor, calling a specific function. Then we do not care about the level of this neighbor because the algorithm automatically select the right operation: interpolation if $l < 2$, averaging if $l > 2$ or simply get the value from the cell if $l = 2$.

7.9 AMR for NS equations

A similar approach can be used also for the other equations of fluid dynamics with some modifies. For example to solve the NS equations on an octree grid, the refinement criterion (see 4.3.3), the interpolation procedures and the advection scheme must be chosen with particle care to obtain the best results.

Furthermore particular care must be put on the solution of the pressure solver. Some specific projection method and Poisson solver have been developed for octree grid ([14; 15]). The biggest difficulties are the high-order calculation of the pressure gradient at the cell faces and the solution of the linear system obtained which is usually for Octree grid non-symmetric.

Chapter 8

Interface Curvature

In this chapter we explain the problem of parasitic currents and a new method to calculate curvature to prevent them.

8.1 Parasitic currents

Parasitic currents are unphysical velocities generated by numerical methods for front tracking where surface tension is approximated as a continuous variable. Let us consider a two-dimensional simulation where a viscous liquid drop is initially centered in a square cavity of characteristic length L, full of air. The radius of the circular shape drop is R. A zero-gravity field is imposed, the flow is assumed isothermal and the surface tension coefficient is constant. The test can be easily extended to three dimensions. The problem consists in verifying the equilibrium of the drop initially at rest. The pressure jump between the inside and outside drop pressures p_l and p_g is given by the *Laplace equation*

$$p_l - p_g = \frac{(d-1)\sigma}{R} \tag{8.1}$$

where d is the spatial dimension (in our case 2).

At the equilibrium, pressure forces should be balanced by the surface tension term keeping a zero velocity field. In Figure 8.1 we can see that, after 400 time steps this equilibrium is not conserved. This is because, the surface tension term in

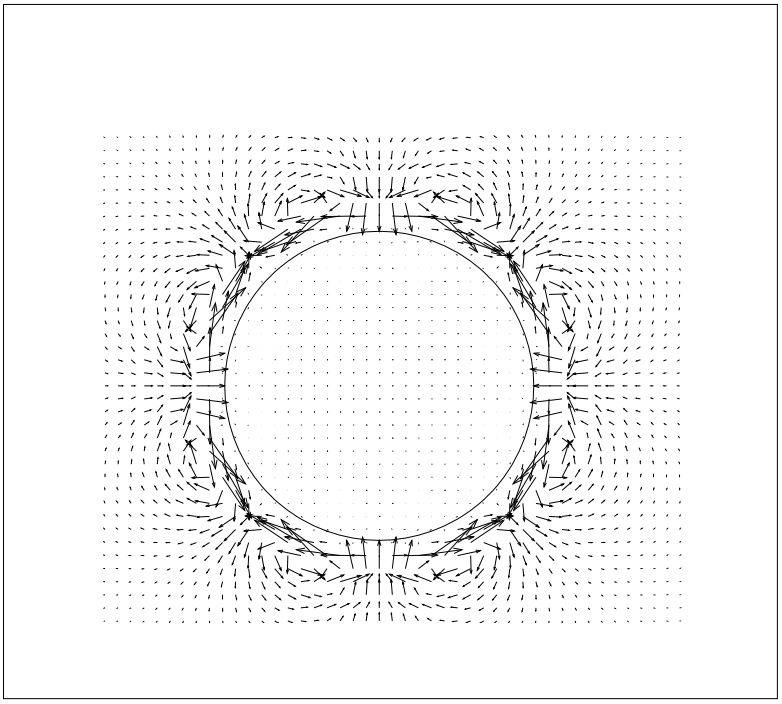

Figure 8.1: *Parasitic currents.*

the discretized equation is not perfectly isotropic. We tried to solve this problem using a better approximation of the curvature. In the following sections we explain it. Unfortunately this was not enough to prevent the parasitic currents because also other terms like the normal and the density contributes to the surface tension term.

8.2 Discretization of Curvature

To calculate the surface tension term in the momentum equation, we need an approximation of the curvature in each cell. In a finite volume formulation (for a 2D uniform grid) we can write the curvature as

$$\kappa_{ij} = \frac{1}{\int_V dV} \int_V \nabla \cdot \frac{\nabla \phi}{|\nabla \phi|} dV = \frac{1}{\int_V dV} \int_{\partial V_{ij}} \frac{\nabla \phi}{|\nabla \phi|} \cdot \mathbf{n} dS \tag{8.2}$$

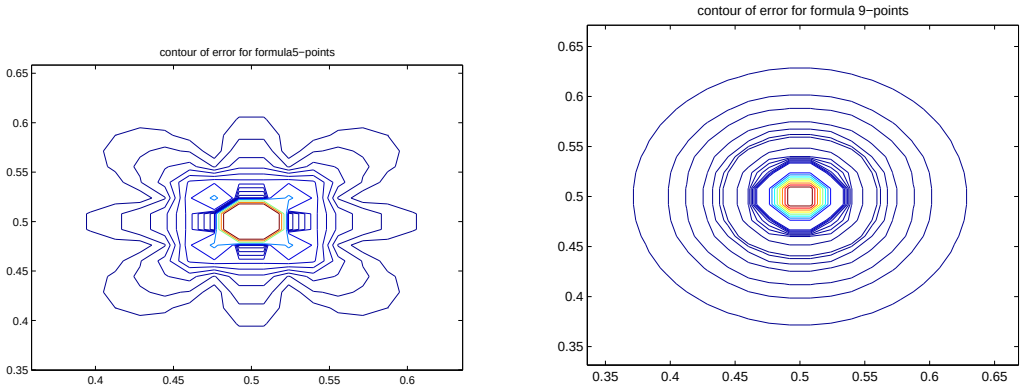

Figure 8.2: *Comparison of the discretization error for the Laplacian using 5-point stencil (left) and 9-point stencil (right).*

where V is the finite volume (or the computational cell). TransAT approximates the integral using a mid-point quadrature rule so that for a square cell of size h

$$\kappa_{ij} \approx \frac{1}{h}\left(f(\mathbf{x}_{i-\frac{1}{2},j}) + f(\mathbf{x}_{i,j-\frac{1}{2}}) - f(\mathbf{x}_{i+\frac{1}{2},j}) - f(\mathbf{x}_{i,j+\frac{1}{2}})\right) \tag{8.3}$$

where $f = \frac{\nabla\phi}{|\nabla\phi|}$. The gradient at the mid-point of the face is then calculated using a central difference. For example $\frac{\partial\phi}{\partial x}(\mathbf{x}_{i-\frac{1}{2},j}) = \left(\phi(\mathbf{x}_{i-1,j}) - \phi(\mathbf{x}_{i,j})\right)/\Delta x$. So the complete formula of curvature uses only five neighbor cells, neglecting the diagonal neighbor and this results in an anisotropic approximation error.

8.3 Nine points stencil

We developed a new formula to calculate the curvature that use 9 point to extrapolate the curvature in a cell (i, j). To do this we approximate the integral of 8.2 along a face using the Simpson rule

$$\int_{i-\frac{1}{2},j-\frac{1}{2}}^{i-\frac{1}{2},j+\frac{1}{2}} f \approx \frac{1}{6}(f_{i-\frac{1}{2},j-\frac{1}{2}} + 4f_{i-\frac{1}{2},j} + f_{i,j+\frac{1}{2}}) \tag{8.4}$$

where we have taken as example the integral over the west face. The derivatives $f_{i\pm\frac{1}{2},j\pm\frac{1}{2}}$ are then calculated using central difference so that the terms coming from diagonal neighbors $\phi_{i\pm1,j\pm1}$ appears. We obtain a *compact 9-point stencil.*

In Figure 8.2, we tested the 9-point stencil obtained using this method to approximate the Laplacian $\nabla \cdot \nabla \phi$ of a function $\phi = x^2 + y^2$. This is very similar to the discretization of the curvature for a circular drop. We plotted the level set lines of the discretization error. We can see that the 9-point stencil is perfectly isotropic.

Note that the order of the two methods is the same. In fact, even if the Simpson rule has 2^{nd} order accuracy, when we approximate the derivatives in the face center $(i - \frac{1}{2}, j)$ or in the cell corner $(i - \frac{1}{2}, j \pm \frac{1}{2})$ using linear interpolation between (i, j) and one of the neighbor (diagonal, vertical or horizontal), we lose the 2^{nd} order accuracy.

Chapter 9

Simulations

In this chapter we present and analyze some simulations with TransAT BMR and TransAT AMR. We compare some of our multi-phase simulations with the solutions obtained with the FEM software COMSOL Multiphysics 3.4. It also uses the Level Set Method for tracking interface although in a slightly different manner and inside a finite element framework.

9.1 TransAT BMR

9.1.1 Driven cavity

We set up this test case to validate the BMR method comparing it with two uniform grids.

- Computational domain: $\Omega = [0, 1] \times [0, 1]$
- $Re = 1000$
- Wall boundary conditions (no-slip) at south, east and west boundaries
- $u = 1$ and $v = 0$ at north boundary
- 3 different meshes: 20×20, 40×40 and a BMR grid with 1205 cells. Another reference simulation with a grid of 60×60 was also set up

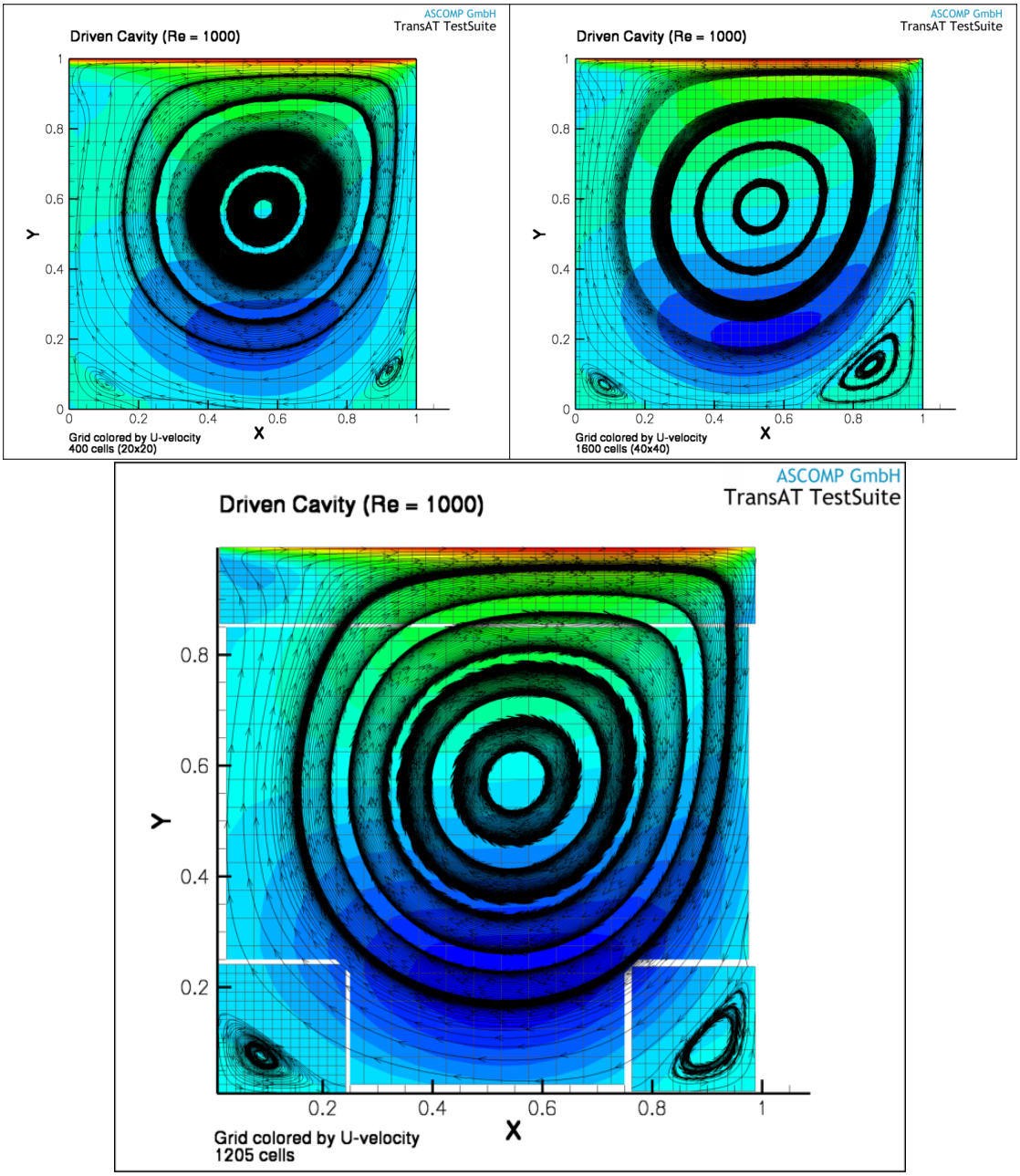

Figure 9.1: *Driven cavity. Stream lines with* 20×20 *grid,* 40×40 *grid and a BMR grid with 1205 cells.*

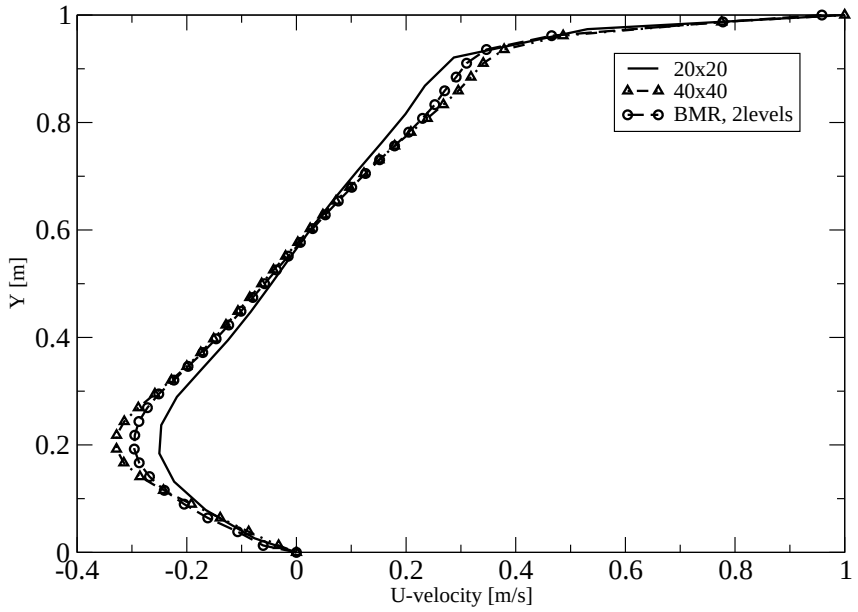

Figure 9.2: *Driven cavity. Comparison of u-velocity between the three different grids of figure 9.1*

9.2 TransAT AMR

9.2.1 Level set advection

This first simulation presented is the one we used to test the software and to compare it with TransAT without AMR. We set up a simple test case in 3D in which there is a water drop pushed by a constant inflow inside a channel of infinite width and length. We do not consider gravity and surface tension. The domain is $\Omega = [0, 5] \times [0, 1] \times [0, 1]$, the boundary condition are all symmetric except for an inflow $u = 0.1$ in the east boundary and an outflow condition at the west boundary and the initial condition are in all the domain $v = w = 0$, $u = 0.1$.

We ran this simple test case for all the schemes and the methods implemented and we present here a simulation with the following methods and parameters:

- uniform father grid of 640 cells ($40 \times 8 \times 8$),

- explicit 1st order Euler for the time stepping,
- constant time step $\Delta t = 0.1$ for all the levels,
- no mass correction after level set reinitialization,
- QUICK scheme for advection,
- three AMR levels
- ϕ is interpolated using tri-quadratic interpolation,
- velocities are copied from the coarse grid to the fine,
- octree refreshed (de-refined or refined where is necessary) every time step.

In the finest grid the Courant number is $C = u\Delta t/\Delta x_3 = 0.32$ and for the 1st level grid $C = u\Delta t/\Delta x = 0.08$ where of course $\Delta x_3 = \Delta x/4$.

In the first picture of figure 9.3 we have the evolution of the drop without AMR. Only for this case we used a $\Delta t = 0.4$ so that we have the same Courant number of the finest grid in AMR calculations. The second picture is the AMR solution without Level Set reinitialization: the drop shape is completely lost and from this is evident the importance of reinitialization that we have introduced in the third picture. The solution without AMR is obtained also without reinitialization, otherwise the drop would disappear after few time steps because the grid is too coarse. In the last picture is represented the solution with TransAT without AMR using a grid fine as the AMR grid in all the domain. For our point of view this is the "reference solution" because it tells us the upper limit of accuracy that we could get with AMR. Looking at the last two picture and at the volume of the drop plotted in picture 9.4 we can say that, although the drop shape is not well captured in anyone of these simulations, the AMR solution is very close to the "best" solution.

The choice that compromises more this result is the 1st order explicit time scheme. Using a better time scheme like a Runge-Kutta method the results are very different. With a 3^{rd} order scheme the shape is preserved but in Figure 9.5 we can see that there are some mass loss, especially if we decrease the number of levels

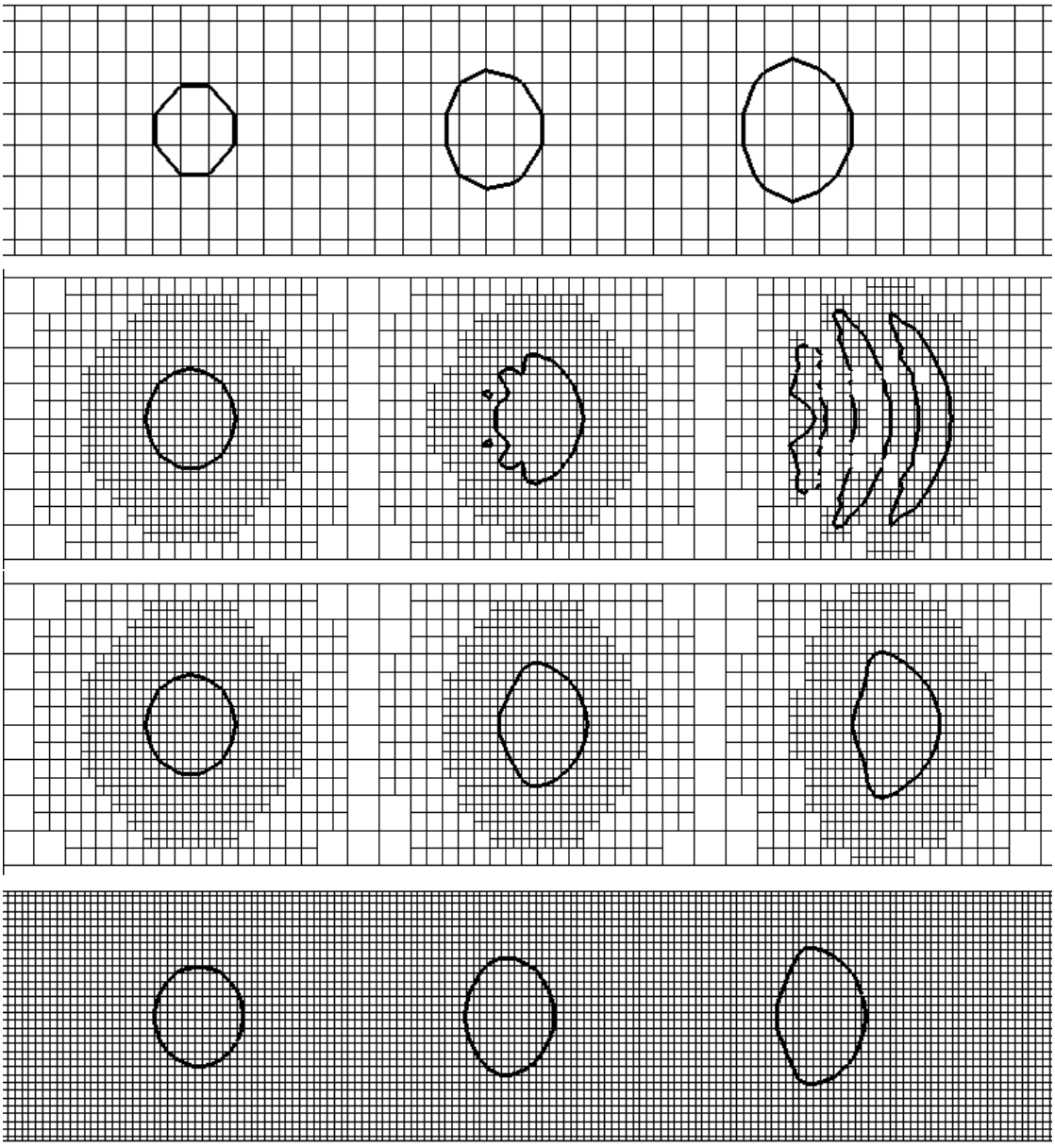

Figure 9.3: *Drop passive advection. Shape comparison for $t = \{0; 14; 28\}$ between, from top to bottom: TransAT without reinitialization, TransAT AMR without reinitialization, TransAT AMR with reinitialization and reference solution of TransAT with a grid refined everywhere.*

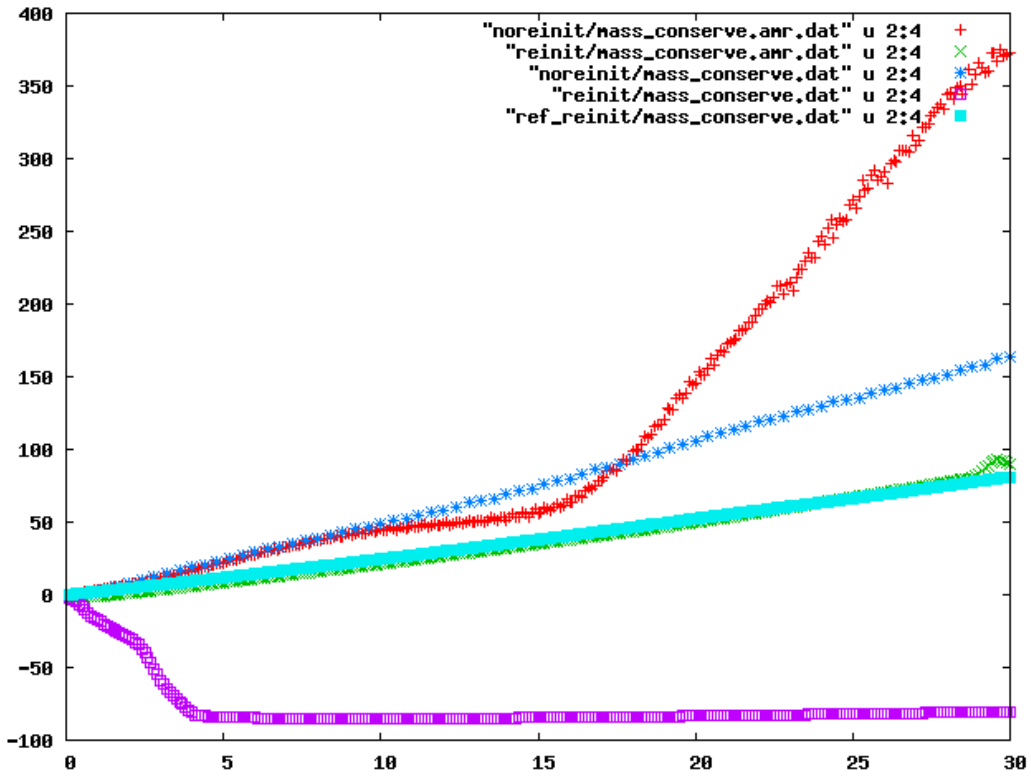

Figure 9.4: *Drop passive advection. Rate of mass gained or lost. The red line is AMR solution without reinitialization which in the first part is close to the solution without AMR and reinitialization (dark blue). The violet line is with reinitialization without AMR, in this case the drops disappears. The reference solution (light blue) and the AMR solution with reinitialization (green) are very similar.*

to 2. This same simulation with an high order scheme for time discretization and with the mass correction term in the reinitialization equation is exactly resolved. In Figure 9.6 we can see that the shape is well captured and the mass is conserved (mass gain is $< 1\%$).

In this simulation, with AMR approach we approximately the same result of the "reference" simulation with a fully refined grid but with only 16.420 cells instead of 163.840. So we could save approximately the 90% of cell points and CPU time. Unfortunately, all the AMR functions are still under development and not yet optimized. Furthermore we still do not use the parallel protocol openMP that is used by TransAT for parallel computation so, for the AMR solution with reinitialization we get approximately the same CPU time as the "reference"

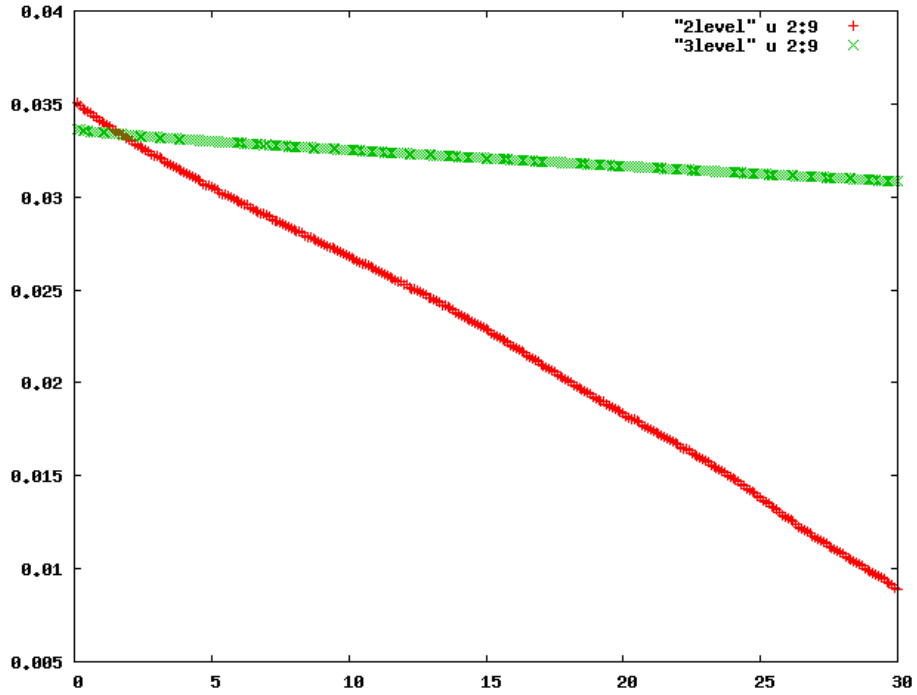

Figure 9.5: *Drop passive advection. Mass of the drop with* 3^{rd} *order Runge-Kutta. Comparison between 2 and 3 levels.*

solution. Also the choice for the interpolation and the octree refreshing are not the best ones in term of CPU time. If we use the diagonal interpolation with a refreshing rate of 3 we can halve this time and we could do much more after the optimization and with parallelization.

9.2.2 Droplet breakup

Emulsions consist of small liquid droplets immersed in another liquid and are widely used, for instance in the production of food, cosmetics, and pharmaceutical products. The quality of the products typically depends on the size of these droplets. This simulation shows how droplets in an emulsion can be created. From the results, you can determine the size of the created droplets and the rate with which they are produced. The setup of this simulation is taken from a a model provided in the COMSOL Multiphysics 3.4 Model Library ([6])

Figure 9.7 shows the geometry of the T-shaped micro-channel. The fluid to be

dispersed into small droplets, Fluid 2, enters through the vertical channel. The other fluid, Fluid 1, flows from the right to left through the horizontal channel.

This simulation has the following parameters:

- Constant density $\rho = 1000\ kg/m^3$ for both fluids.
- Dynamic viscosity $\eta_1 = 0.00195$ for fluid 1 and $\eta_2 = 0.00671$ for fluid 2.
- Parabolic inlet profile $u = -u_1 s(1-s)$ for fluid 1 and $v = -v_1 s(1-s)$ for fluid 2. s is the normalized distance from the wall (i.e. 0 at the wall and 1 in the center of the pipe). $u_1 = 0.0166\ m/s$ and $v_1 = 0.0083\ m/s$.
- Wetted wall boundary condition at all solid boundaries with a constant contact angle of 135 degree.

We solved this problem with TransAT AMR using

- 3 AMR levels.
- Quadratic interpolation of level set value and fluxes.
- 3^{rd} order Runge-Kutta scheme for time discretization.
- QUICK scheme for level set advection and HLPA scheme for velocity.
- Adaptive time step.
- Level Set reinitialization with 3^{rd} order WENO schemes and mass correction.

Figure 9.8 shows the fluid interface (i.e. the isocontour of the level set function) at various times. After 0.08 s the first droplet is formed. In COMSOL Multiphysics 3.4 the interface is represented by a certain level set or isocontour of a globally defined function, the level set function, ϕ that is a smooth step function that equals zero in a domain and one in the other. Across the interface, there is a smooth transition from zero to one. So the interface is defined by the 0.5 isocontour, or the level set, of ϕ. For more details see Chapter 5 of the Chemical Engineering Module User's Guide ([6]).

Instead in TransAT the level set function is a signed distance function from the interface and the reinitialization step reset ϕ to be a distance function in a region close to the surface. However in the last picture we can see that the interface is not smooth and unstable. This is because we did the reinitialization every 10 time steps and we solved the reinitialization equation with only 15 pseudo-time steps. So, when the fluids are moving and mixing faster, it is not enough to converge and Level Set function does not represent always a distance function. Then, the solution became inaccurate and going on with time small drops can appear or disappear in the pipe because of numerical errors. Furthermore also the octree refinement does not cover exactly the region close to the interface because we cannot have information about the true distance from the interface. Another possible cause is that we don't calculate the surface tension force in the finer mesh (see Chapter 8).

In COMSOL Multiphysics 3.4 the diameter of the produced droplet at $t = 0.08\,s$ can be calculated from

$$d = 2\sqrt{\frac{\int_\Omega (\phi > 0.5) d\Omega}{\pi}} \tag{9.1}$$

where Ω is the leftmost part of the horizontal channel, where $x < -310^{-4}m$. The results show that $d \approx 1.4510^{-4}m$.

To calculate the drop diameter in TransAT we use a smoothed discretized H function (see 2.2.2 and 7.7) obtained from the level set function ϕ in the cell (i, j) as $H_{ij} = g(\phi_{ij})$ where $g(\phi) = \frac{1}{2}\left(1 + \tanh\left(2\frac{\phi}{\epsilon}\right)\right)$ and ϵ is a reference distance representing the interface width. Also other choices are possible for g (e.g. linear, sinusoidal or sharp step). In this case we calculate the diameter as

$$d = 2\sqrt{\frac{1}{\pi}\sum_{i,j} H_{ij}} \tag{9.2}$$

and the result is $d \approx 1.4810^{-4}m$.

From this we can understand the importance of Level Set reinitialization. Initially which, (see Figure **??**). In this case the solution can be inaccurate and going on with time small drops can appear or disappear in the pipe because of numerical

errors. Furthermore also the octree refinement does not cover exactly the region close to the interface because we cannot have information about the true distance from the interface.

9.2.3 Capillary filling

We take this example from the COMSOL Multiphysics 3.4 Model Library ([6]) and we run the same simulation with TransAT AMR to show the importance of surface tension and wall adhesion in particular real problems.

In fact, surface tension and wall adhesive forces are often used to transport fluid through micro-channels in MEMS (Micro-electromechanical systems) devices or to measure, transport and position small amounts of fluid using micro-pipettes. Multi-phase flows through a porous medium and droplets on solid walls are other examples where wall adhesion and surface tension strongly influence the dynamics of the flow. This example studies a narrow vertical cylinder placed on top of a reservoir filled with water. Because of wall adhesion and surface tension at the air/water interface, water rises through the channel. The geometry (shown in Figure 9.9) consists of a capillary channel of radius $0.15\,mm$ attached to a water reservoir. Water can flow freely into the reservoir. Because both the channel and the reservoir are cylindrical, we use an axis-symmetric 2D simulation. Initially, the thin cylinder is filled with air. The wall adhesion causes water to creep up along the cylinder boundaries. The deformation of the water surface induces surface tension at the air/water interface, which in turn creates a pressure jump across the interface. The pressure variations cause water and air to move upward. The fluids continue to rise until the capillary forces are balanced by the gravity force that builds up as the water rises in the channel. In the present example, the capillary forces dominate over gravity throughout the simulation. Consequently, the interface moves upwards during the entire simulation.

- Initially, the reservoir is filled with water and the capillary channel is filled with air. The initial velocity is zero.

- The hydrostatic pressure $p = \rho g z$ gives the pressure at the inflow boundary.

- At the outlet, the pressure is equal to zero, that is, equal to the pressure at the top of the inflow boundary.
- The wetted wall has a static contact angle of 65 degrees.

We solved this problem with the following parameters:

- 3 AMR levels.
- Quadratic interpolation of level set value and fluxes.
- 3^{rd} order Runge-Kutta scheme for time discretization.
- QUICK scheme for level set advection and HLPA scheme for velocity.
- Adaptive time step.
- Level Set reinitialization with 3^{rd} order WENO schemes and mass correction.

We obtained the same results of COMSOL Multiphysics 3.4 (see Figure 9.10) in terms of interface shape even if the water fills the capillary at a lower velocity. This different behavior can be due the different way boundary conditions are implemented. In TransAT the velocity is imposed using cell-centered variables, so there is no variable defined exactly on the wall. In COMSOL Multiphysics 3.4 it is used the Finite Element Method (FEM) to solve the Navier-Stokes equations, so the equations are multiplied by test functions and then integrated over the computational domain. Adding capillary forces results in an additional integral, involving the contact angle, to the main one. If no-slip boundary condition are applied the boundary term vanishes because test functions are equal to 0 on that boundary, so the contact angle cannot be specified and the interface remains fixed on the wall. A small amount of slip has to be allowed in order to specify the contact angle. This is achieved using the wetted wall boundary condition that adds the above mentioned additional integral allowing to set the contact angle, the velocity component normal to the wall to zero and adding a frictional boundary force by specifying a slip length (so allowing a small tangential velocity $v_t \neq 0$). For more details see Chapter 4 of the MEMS Module Model Library ([6]).

Initially, the shape of the interface changes dynamically and after about $0.6\,ms$ ($0.9\,ms$ with TransAT) it remains constant and forms a rising concave meniscus. Note that the slip velocity at the walls is almost zero, except close to the fluid interface/wall contact point.

Figure **??** shows the pressure profile when the interface has reached his final shape. Consistently with the difference of velocity, also the pressure jump at the fluid interface is slightly different. We obtain a a pressure jump of roughly $200\,Pa$, while with COMSOL we have $300\,Pa$. This pressure jump is caused by the surface tension, and forces water and air to rise through the channel.

9.2.4 Bubble rise

Hnat and Buckmaster studied the different shapes and terminal velocities of bubbles rising in incompressible liquids [43]. Their data have served as benchmark for various numerical codes for interfacial or free surface flows. These are used here to validate the treatment of surface deformation due to external flow. In particular, we will use the case of Fig. 1a in Hnat and Buckmaster [43], with the following fluid parameters:

- density $\rho_l = 875.5\,kg/m^3$
- dynamic viscosity $\eta_l = 0.118\,Pa\,s$
- surface tension coefficient $\sigma = 3.22 \times 10^{-2}\,J/m^2$,
- bubble radius $R = 6.08 \times 10^{-3}\,m$
- gravitational acceleration $g = 9.8\,m/s^2$

For the gas we use a arbitrary values $\rho_g = \rho_l/1000$ and $\eta_g = \eta_l/1000$. The experimentally measured terminal rising velocity of the bubble is $u = 0.215\,m/s$ so we take this value as inflow condition simulating in this way the movement of the bubbles.

To simulate this problem we use a 2D axis-symmetric domain $\Omega = [0\,;\,0.061] \times [0\,;\,0.0305]$ with gravity directed in x-direction and with rotations around the x-axis. Because of axis-symmetry, only the right half of the meridian plane is calcu-

lated. Symmetry conditions are used on the y-axis, inflow and outflow conditions are imposed on the x-boundaries. The initial conditions are shown in Figure 9.12.

We solved this problem with the following parameters:

- 3 AMR levels.
- Quadratic interpolation of level set value and fluxes
- 3^{rd} order Runge-Kutta scheme for time discretization
- QUICK scheme for level set advection and HLPA scheme for velocity.
- Constant time step $\Delta t = 0.05\, ms$.
- Level Set reinitialization with 3^{rd} order WENO schemes and mass correction.

In Figure 9.13 we can see the result of the AMR solution compared with the final shape obtained with a reference grid refined everywhere. We can clearly see a different shape that is probably due to a inaccurate approximation of the surface tension term in the AMR solution. In fact, in this simulation the interface is captured using the refined grid but the curvature and the normal vector are still calculated in the coarse grid so the representation of surface tension is not coherent.

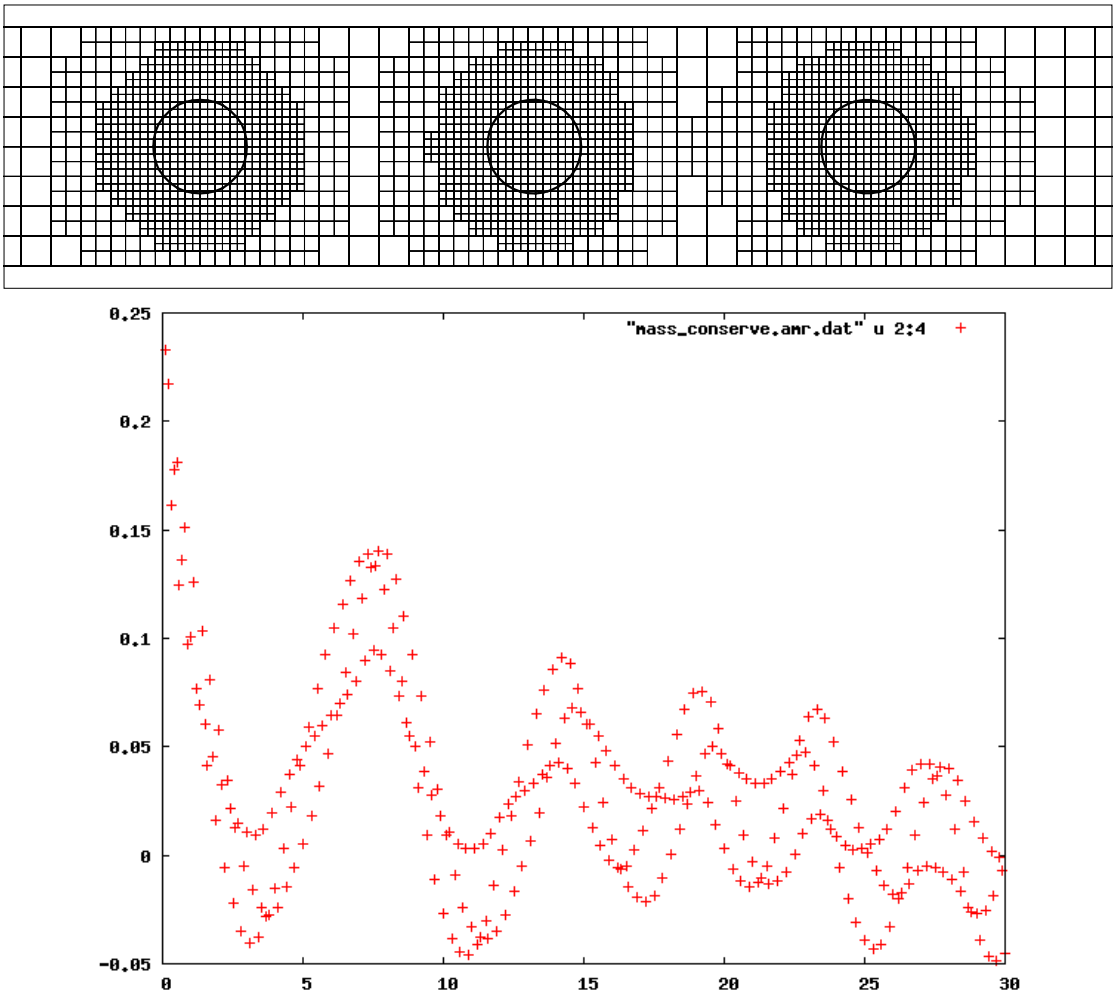

Figure 9.6: *Passive drop advection with higher order time scheme and mass correction. Shape comparison and mass loss.*

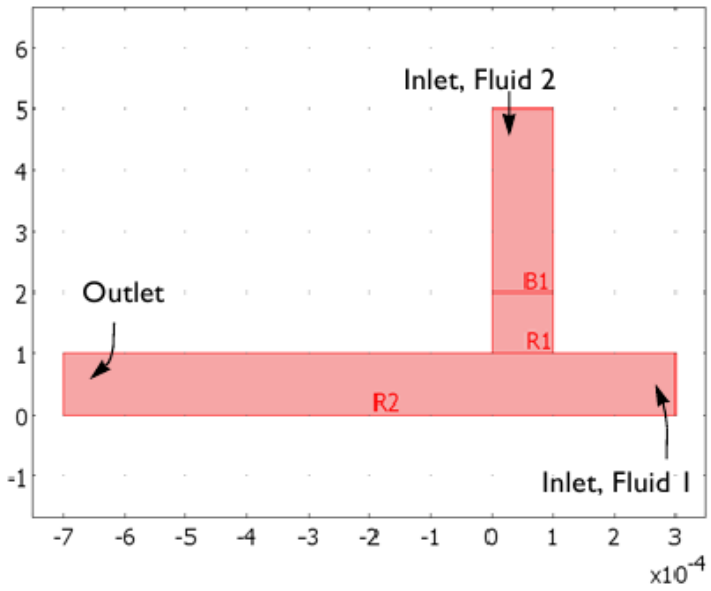

Figure 9.7: *T-junction geometry of the simulation 9.2.2 (picture taken from [6]).*

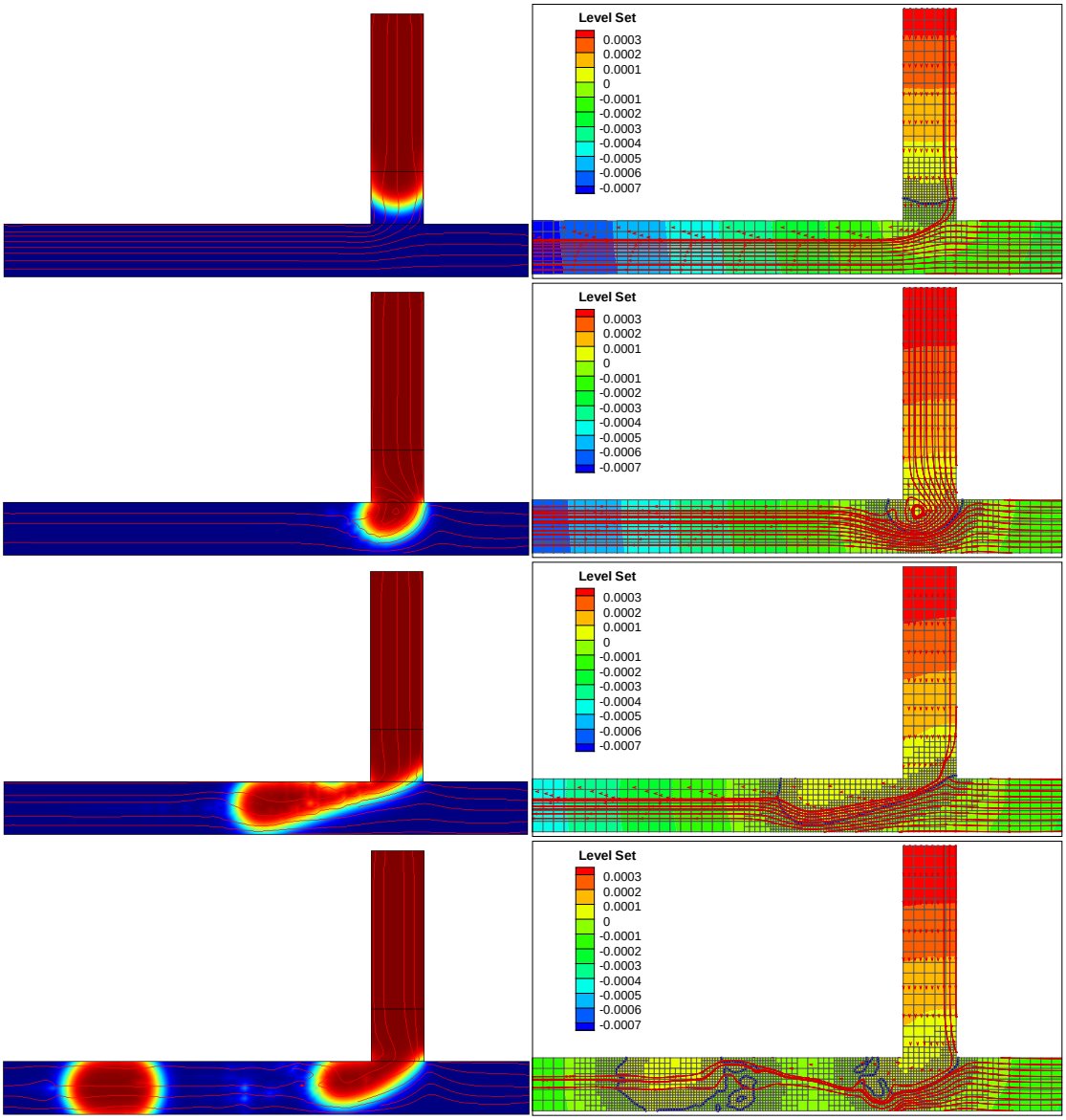

Figure 9.8: *Droplet breakup. Level set function and stream lines. Comparison between COMSOL Multiphysics 3.4 (left) and TransAT AMR (right) at* $t = 0.02$,$t = 0.04$,$t = 0.06$,$t = 0.08$.

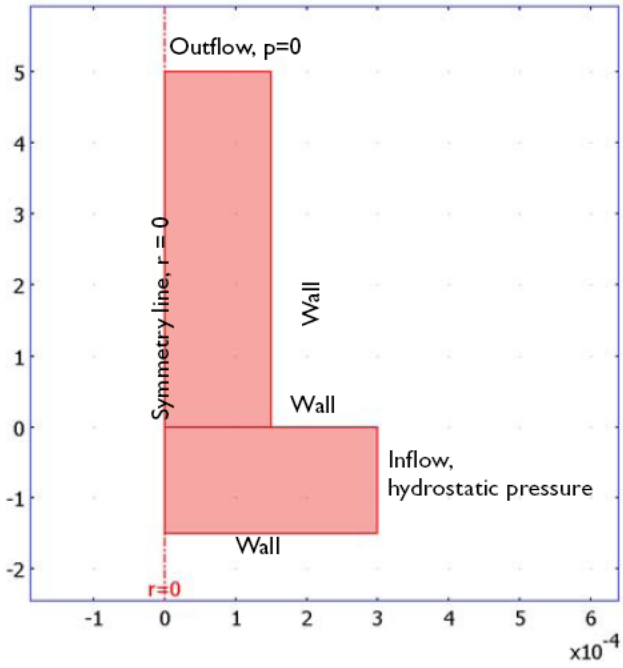

Figure 9.9: *Geometry description for the capillary filling. Picture taken from [6].*

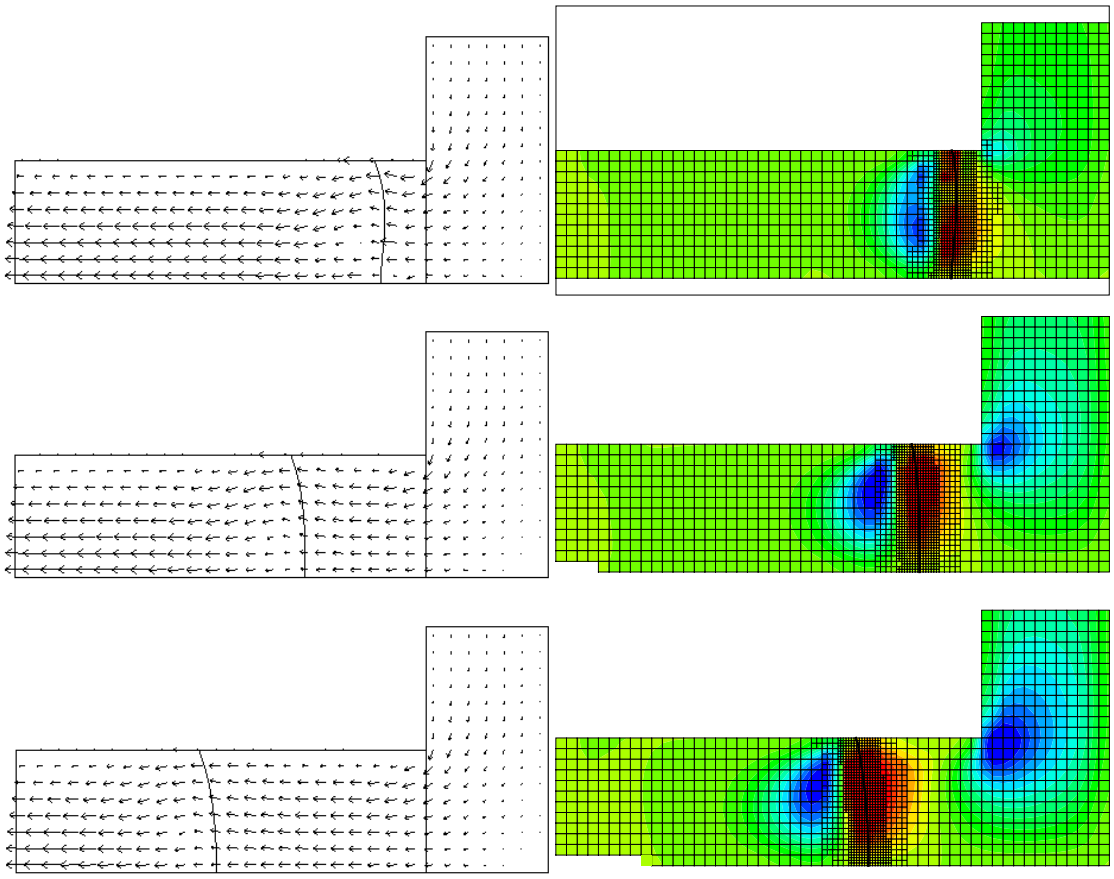

Figure 9.10: *Interface and velocity field with COMSOL Multiphysics 3.4 at* $t = 0.2\,ms$,$t = 0.4\,ms$,$t = 0.6\,ms$ *(left) and TransAT AMR at* $t = 0.3\,ms$,$t = 0.6\,ms$,$t = 0.9\,ms$ *(right).*

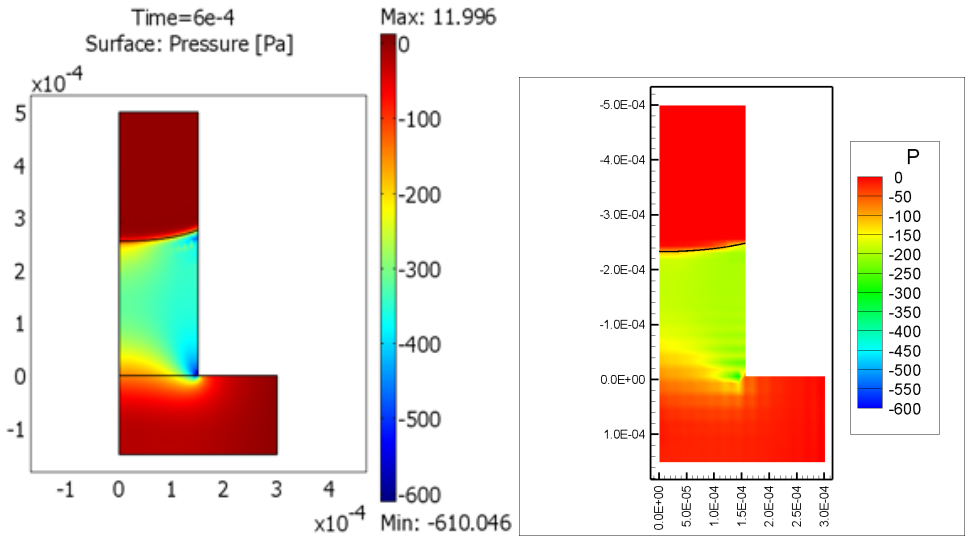

Figure 9.11: *Pressure calculated with COMSOL Multiphysics 3.4 at $t = 0.6ms$ (left) and with TransAT AMR (right) at $t = 0.88ms$.*

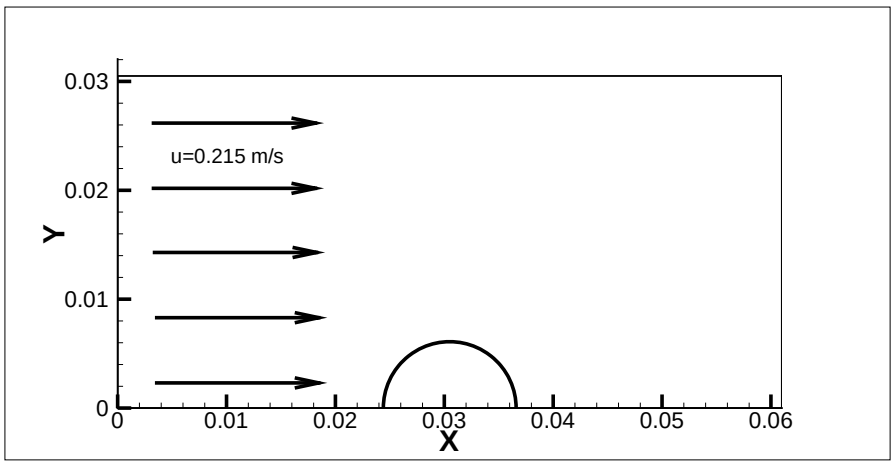

Figure 9.12: *Initial conditions for the bubble rise test case.*

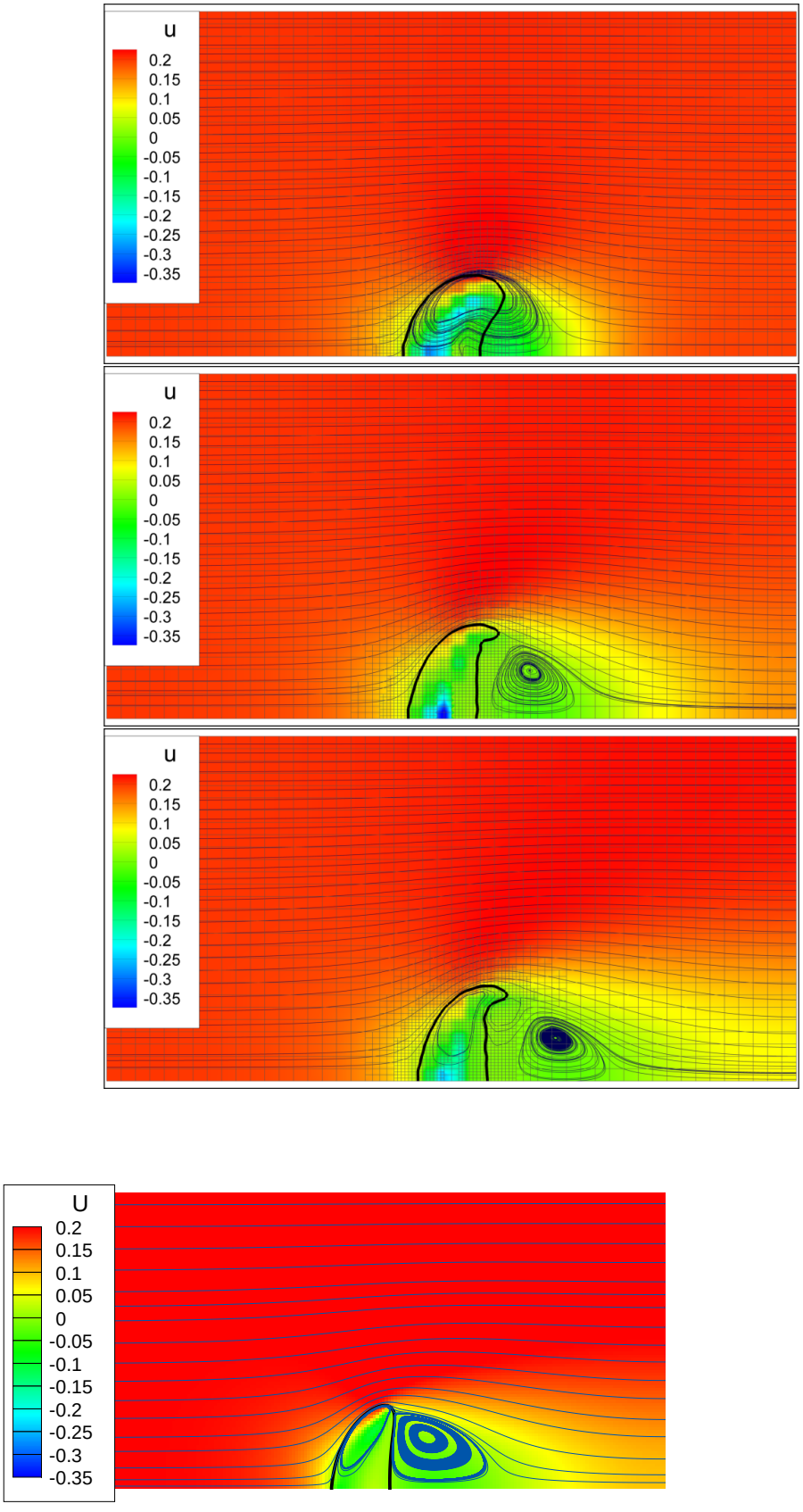

Figure 9.13: *Bubble rise. Shape, u velocity and stream lines at, from top to bottom, $t = 0.05\,s$, $t = 0.15\,s$, $t = 0.3\,s$ with AMR and $t = 0.3\,s$ with a fully refined reference solution.*

Chapter 10

Conclusions

This final chapter deals with some critical comments and analysis of the work done and the future development

10.1 Critical comments

We developed and implemented two different types of mesh refinement. BMR, although is not adaptive (meaning that the refined grid are fixed in time), resulted to be very useful in many situations where we know in advance the zones where we need a better resolution of the flow like in the examples shown in Chapter 6. BMR method and TransAT-Mesh is now widely used by ASCOMP in many real problems. However, this method still need to be further tested and validated with other test cases (for example coupled with phase tracking).

For the front tracking of fluid interface we used instead a dynamic adaptive mesh refinement and although we are only at a beginning stage and there are still much work to finish, correct, optimize and develop new schemes (see 10.2), we can already see the potentialities of our approach. For example we can suppose that the AMR could be very useful for applications where we want to track, follow and make statistics on small drops or bubbles that do not have a strong influence on the velocity field but that would disappear if we use for the Level Set equation the same grid of the NS equations. The simulations showed in Chapter 9 do not cover this kind of applications but they were useful to validate the AMR, check

his capabilities and analyze the CPU time needed.

Finally we can summarize the main achievements of this work in the following points:

- a Local Defect Correction method to solve advection-diffusion equation on composite grids,
- an automatic multi-block mesh generator to build composite grids for complex geometries,
- a method to refine automatically a Cartesian grid with a binary tree structure,
- a method to solve Level Set and Hamilton-Jacobi problems on tree-based grid,
- a method to calculate the curvature with isotropic error.

10.2 Further developments

All the applications explained in this work are under constant development so there are always additions and new ideas that can be implemented in them. In this section we try to summarize the most important additions, corrections and developments to be made.

TransAT AMR

Many improvements could be done to improve this method. First of all we have to improve the coupling with TransAT, calculating all possible information in the finer grid (e.g. curvature, normal, density, viscosity) and passing it to TransAT.

Then much work still remains to speed up and optimize all the functions and the data-structure. In particular, the next step of the work will be to implement the structure proposed in [20] to perform efficient traversal method in the octree.

To speed up our AMR, we can also implement the narrow band method (see 2.4.2), solving the Level Set problem only in a region close to the interfaces. However, this is not trivial because generally we know the distance from the interface only if we solve the reinitialization equation everywhere.

Finally, although it was not one of our aim, the AMR could be extended to the solution of the NS equations.

TransAT BMR

For BMR we could implement a different time stepping procedure, using subcycling (see Figure 4.6) to advance in time the coarser grid faster and independently from the finer ones. Furthermore we need to validate our method with other test cases, in particular for low Reynolds number, when the Local Defect Correction is more important because diffusion dominates the N-S equations.

TransAT-Mesh

Currently TransAT-Mesh is a pre-processing tool that generates input files for TransAT. The post-precessing still needs to be done with external tools (e.g. Tecplot). A possible improvement is to link directly TransAT-Mesh to TransAT and see the results of the simulation inside TransAT-Mesh. Furthermore it could very useful making TransAT-Mesh capable of including a feature for multi-phase flow. For example we could include in the projects and in the output files also information about the phase of every object. Finally a big work could be done to improve the user-friendliness and the graphical aspect.

Bibliography

[1] *TransAT User Manual,* Ascomp GmbH, Zürich (2008).

[2] G. Yadigaroglu, D. Lakehal, S. Zaleski, S. Banerjee et al., *Lecture Notes*, 25th Short Courses on Modelling and Computation of Multiphase Flows, ETH Zürich (2008). 11

[3] D. Caviezel, *Parallelization of the Multiblock CMFD-Code TransAT-MB*, Master Thesis, ETH Zürich (2006). 51

[4] M. Birrer, *Heat Transfer Management in Electronic Systems*, Master Thesis, ETH Zürich (2007). 58

[5] V. Marra, *Optimization of Volume-of-Fluid (VOF) Methods and Two-Phase Flow Simulation*s, Ph.D. Thesis, University of Bologna (2006). 12, 17

[6] COMSOL Multiphysics® 3.4, *Documentation Set*, COMSOL AB, (October 2007). 94, 95, 97, 98, 102, 104

[7] M. Icardi, D. Caviezel and D. Lakehal, *The Immersed Surfaces Technology for Reliable and Fast Setup of Microfluidics Simulation Problems,* NSTI Nanotech Conference, Boston (2008). 62, 67

[8] M. J. H. Anthonissen, B. van 't Hof, A. A. Reusken, *A finite volume scheme for solving elliptic boundary value problems on composite grids*, Computing 61, 285-305 (1998).

[9] R. Minero, H.G. Morsche and M.J.H. Anthonissen, *Convergence properties of the local defect correction method for parabolic problems*, CASA Report No. 05-40, Technische Universiteit Eindhoven (2005).

[10] W. Hackbusch, *Local defect correction method and domain decomposition techniques*, Computing 5, 89-113 (1984). 54, 56

[11] S. Osher and J. A. Sethian, *Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations*, Journal of Computation Physics 79, 12–49 (1988). 23, 26, 31, 34

[12] J. A. Sethian, *Level Set Methods and Fast Marching Methods*, Cambridge University Press (2006).

[13] M. Herrmann, *Refined Level Set Grid Method for Tracking Interfaces*, Annual Research Briefs, Center for Turbolence Research, Stanford University (2005). 71

[14] S. Popinet, *Gerris: a tree-based adaptive solver for the incompressible Euler equations in complex geometries*, Journal of Computational Physics 190, 572-600 (2003). 43, 46, 83

[15] C. Min and F. Gibou, *A second order accurate level set method on non-graded adaptive cartesian grids*, Journal of Computational Physics 225, 300-321 (2007). 43, 46, 83

[16] J. Strain, *Tree Methods for Moving Interfaces,* Journal of Computational Physics, 151, 616-648 (1999). 73

[17] F. Losasso, R. Fedkiw and S. J. Osher, *Spatially adaptive techniques for level set methods and incompressible flow,* Computers & Fluids 35, 995-1010 (2006). 71

[18] T. C. Cecil , S. J. Osher , J. Qian, *Simplex free adaptive tree fast sweeping and evolution methods for solving level set equations in arbitrary dimension*, Journal of Computational Physics 213, 458-473 (2006). 76

[19] A. M. Khokhlov, *Fully threaded tree algorithms for adaptive refinement fluid dynamics simulations*, Journal of Computational Physics 143, 519-543 (1998). 82

[20] L. Balmelli, J. Kovacevic, M. Vetterli, *Quadtree for embedded surface visualization: constraints and efficient data structures,* Proceedings of IEEE Int. Conf. Image Processing (ICIP) 2, 487-491 (1999). 82, 109

[21] M. Sussman, A. Almgren, J. Bell, P. Colella, L. Howell, and M. Welcome, *An adaptive level set approach for incompressible two-phase flows*, Journal of Computational Physics 148, 81-124 (1999). 43, 80

[22] M. Sussman, E. Fatemi, P. Smereka, and S.J. Osher, *An improved level set method for incompressible two-phase flows*, Journal of Computers and Fluids 27, 663-680 (1998). 26, 34

[23] H. Touil, M.Y. Hussaini and M. Sussman, *Tracking discontinuities in hyperbolic conservation laws with spectral accuracy*, Journal of Computational Physics 225, 1810-1826 (2007). 24

[24] S. O. Unverdi and G. Tryggvason, *A front-tracking method for viscous, incompressible, multi-fluid flows*, Journal of Computational Physics 100 (1992). 22

[25] C. W. Hirt, B. D. Nichols, *Volume of fluid method for the dynamics of free boundaries,* Journal of Computational Physics 39, 201-225 (1981). 21

[26] M. G. Crandall, L. C. Evans and P. L. Lions, *Some Properties of Viscosity Solutions of Hamilton-Jacobi Equations*, Transactions of the American Mathematical Society 282, 487-502 (1984). 29

[27] L. Corrias, M. Falcone, R. Natalini, *Numerical schemes for conservation laws via Hamilton-Jacobi equations.* Math. Comp. 64 (1995). 30

[28] G.-S. Jiang, D. Peng, *Weighted ENO schemes for Hamilton-Jacobi equations*, SIAM J. Sci. Comp. 21, 2126-2143 (2000). 33, 34, 81

[29] S. Osher and C. Shu, *High-order essentially nonsocillatory schemes for Hamilton-Jacobi equations*, SIAM Journal of Numerical Analysis 28, 907-922 (1991). 31

[30] J. U. Brackbill, D. B. Kothe, and C. Zemach, *A continuum method for modeling surface tension*, Journal of Computational Physics 100, 335-354 (1992).

[31] H. Takahira, M. Takahashi and S.Banerjee, *Numerical Analysis of Three Dimensional Bubble Growth and Detachment in a Shear Flow*, 5th International Conference on Multiphase Flow, ICMF 04, Yokohama, Japan (2004). 80

[32] S. Gottlieb and C.W. Shu, *Total-variation-diminishing Runge-Kutta schemes*, Math. Comp. 67, 73-85 (1998). 34

[33] E. F. Charlton, and K. G. Powell, *An octree solution to conservation-laws over arbitrary regions (OSCAR)*. AIAA Paper 97-0198 (1997). 43, 46

[34] M. J. Aftosmis, M. J. Berger and J. E. Melton, *Adaptive Cartesian Mesh Generation*, in Handbook of Grid Generation , CRC Press, Florida (1999). 40

[35] D. De Zeeuw and K. Powell, *An Adaptively- Refined Cartesian Mesh Solver for the Euler Equations*, Journal of Computational Physics 104, 56-68 (1993). 43

[36] W. J. Coirier, *An Adaptively Refined, Cartesian, Cell-based Scheme for the Euler and Navier-Stokes Equations*, Ph.D. Thesis, Aerospace Engineering Department, University of Michigan (1994). 40

[37] J. Quirk, *An Alternative to Unstructured Grids for Computing Gas Dynamic Flows Around Arbitrarily Complex Two-Dimensional Bodies*, Computers and Fluids 23, 125-142 (1994). 43

[38] M. Berger and J. Oliger, *Adaptive mesh refinement for hyperbolic partial differential equations*, Journal of Computational Physics 53, 484-512 (1984). 43

[39] M. Berger and P. Colella, *Local adaptive mesh refinement for shock hydrodynamics*, Journal of Computational Physics 82, 64-84 (1988). 43

[40] W. Rodi, S. Majumdar and B. Schönung, *Finite Volume Methods for Two-Dimensional Incompressible Flows with Complex Boundaries*, Computer Methods in Applied Mechanics and Engineering 75 p. 369-392 (1989).

[41] W. Majumdar, W. Rodi and J. Zhu, *Three-Dimensional Finite-Volume Method for Incompressible Flows With Complex Boundaries*, Journal of Fluids Engineering 114, p. 496-503 (1992).

[42] F. Gibou, L. Chen, D. Nguyen, and S. Banerjee, *A level set based sharp interface method for the multiphase incompressible Navier-Stokes equations with phase change*, Journal of Computational Physics 222 (2007).

[43] J. G. Hnat and J. D. Buckmaster, *Spherical cap bubbles and skirt formation*, Physics of Fluids 19 (1976). 99

[44] J.H. Ferziger and M. Peric, *Computational Methods for Fluid Dynamics*, Springer-Verlag, Berlin, 3rd edition (2002).

[45] Thompson J.F., Soni B.K. and Weatherill N.P., *Handbook of Grid Generations*, CRC Press, Florida (1999).

[46] R. Mittal and G. Iaccarino, *Immersed Boundary Methods*, Ann. Review Fluid Mech. 37:239-261 (2005). 41

[47] A. J. Chorin and T. J. E. Marsden, *A Mathematical Introduction to Fluid Mechanics*, Springer-Verlag, New York (1979).

[48] G. K. Batchelor, *An Introduction to Fluid Dynamics*, Cambridge University Press, Cambridge (1970).

[49] S. Chandrasekhar, *Hydrodynamic and Hydromagnetic Stability*, Clarendon, New York (1961).

[50] W. L. Briggs, A *Multigrid Tutorial*, SIAM, Philadelphia (1987).