Direct numerical simulation of two phase flows with adaptive mesh refinement

Davide Zuzio

Université de Toulouse

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :
Institut Supérieur de l'Aéronautique et de l'Espace (ISAE)

Discipline ou spécialité :

Dynamique des fluides

Présentée et soutenue par :
Davide ZUZIO

le : vendredi 17 décembre 2010

Titre :

Direct numerical simulation of two phase flows with adaptive mesh refinement

---

Ecole doctorale :
Mécanique, Energétique, Génie civil et Procédés (MEGeP)

Unité de recherche :
ONERA Toulouse, DMAE (Departement Modeles pour l'Aerodynamique et l'Energetique)

Directeur(s) de Thèse :

Jean-Luc ESTIVALEZES

Rapporteurs :
Alain BERLEMONT
Stéphane VINCENT

Autre(s) membre(s) du jury

Dominique LEGENDRE
Jean PIQUET
Stéphane ZALESKI

Dedicated with love to Emilie
and all my family

Acknowledgements

I would like to gratefully thank everyone who played a part in getting this work done, either directly or indirectly. The list of people who helped me is so *huge* that I'm sure I'm missing many of them, so my deepest apologies and gratitude to everyone.

That said, I would like to deeply thank my supervisor Jean-Luc Estivalezes, who discreetly showed me the professional and personal growth involved in a PhD thesis. His insightful advice, his iron will and his patience have been of great value to me. Working with him has been an extraordinary experience, based on reciprocal respect and trust. Thanks also to the readers of my thesis, Alain Berlemont and Stéphane Vincent for their careful reading of my dissertation, and to all the other jury's members, Stéphane Zaleski, Jean Piquet and Dominique Legendre.

A really big thank should include everyone at ONERA, engineers, researcher, technicians, PhD students, post-docs and whatsoever, starting from the department directors Jean Cousteix and Pierre Millan for having hosted me. Thanks to all my PhD comrades who I will not list here, because each of them has equally a place in my memory, and everyone of them should feel warmly involved in these acknowledgements. Finally, I would like to thank the DMAE department for allowing me to continue this adventure together!

I would like also to remember the CERFACS laboratory, which set up the ECCOMET project in which I took part, and which helped me a lot at my arrival in France.

Thanks a lot to my family, who courageously stood my decision to leave Italy, and continued to love and support me. I hope I can make them proud.

And finally, no words could describe my gratefulness and love to Emilie, who it at my side every day and night, and who always supported me. I'm sure my choice to be with her has been the best I have ever made in my life.

This research project has been supported by a Marie Curie Early Stage Research Training Fellowship of the European Community's Sixth Framework Programme under contract number MEST-CT-2005-020426.

The PARAMESH software used in this work was developed at the NASA Goddard Space Flight Center and Drexel University under NASA's HPCC and ESTO/CT projects and under grant NNG04GP79G from the NASA/AISR project.

Contents

List of Figures

List of Tables

Introduction

Contents

1 Motivations

The detailed numerical simulation of aeronautical combustion chambers has become a major topic in the last years, answering to the need to enhance the engine efficiency and to reduce the polluting agents. A key aspect in this field is the atomization process of the fuel (kerosene) injected into the chambers. The role of the atomization process is to generate, starting from an injection of liquid, a cloud of fine droplets on a very small distance. The aim is to maximize the air/fuel surface, easing the vaporization of the fuel and, finally, the combustion. This mechanism is very complex to simulate because of its multi scale aspect: a droplet diameter can be as small as some tenth of μm, the injection fence some mm and the whole combustion chamber tenth of cm. As the detailed simulation of the whole system is not possible, the problem can be split in two "moments". The first can be described by a separated phases model, suitable to describe the small scales of the atomization of the fuel jet. The second can, conversely, be described by a dispersed phase model, able to simulate the evolution of sprays, which follows the transport and the evaporation of the droplets generated by the fragmentation process.

The ONERA/DMAE/MH laboratory, where this thesis has been conceived, follows a certain number of research projects about the multi phase flows which occur in the turbo machines. The ECCOMET (Efficient and Clean COMbustion Early Training) project has been developed together with the IMFT (Institut de Méchanique des Fluides de Toulouse) and the CERFACS (Centre Européen de Recherche et Formation Avancée en Calcul Scientifique) to perform early research work on different combustion problems under different points of view, theoretical as well as experimental and numerical, in order to have a better knowledge and comprehension of the phenomena. This thesis is the ideal continuation of the work done by Couderc [18], which numerically investigated the primary disintegration of a liquid sheet, the physical process which take place in the aeronautical (or internal combustion engines) fuel injectors. All the knowledge in his vast and deep research has been made the most of, aiming to always better performances in numerical simulations.

If the progresses in computer's performances are steadily increasing, so are the needs of numerical researchers whose goal is to perform always more accurate and fast simulations of physical processes. Accurate and fast are two attributes which immediately make the characteristics of the numerical simulation to diverge; several tools can help to get the best of both worlds. Among these are the parallel computation, which ideally should make (given enough resources) a large domain computation to take as much time as a small one, and the mesh refinement, which should lessen the computational weight of a fixed size domain by distributing the nodes in a more efficient way than a uniform disposition. It is in this direction that this work is aimed, given the physical problem (the atomization), how can it be possible to find a good numerical solution when the resources are still limited face to the size of the problem.

2 Atomization of a liquid jet

The purpose of an injector is to introduce the liquid fuel into a combustion chamber and, at the same time, to favour the mixing of the combustive agent and the combustible, in order to optimize the conditions for combustion. This is achieved by an aerodynamic "destructive" effect on the shape of the liquid injection: the original shape possibly being a cylinder or a sheet, the final form of the fuel should be in the form of a droplet cloud, of which characteristics like density or droplet size are function of the injection parameters and geometry. At least three main configuration of injector can be distinguished:

- Pressure atomizers: in these devices a high pression force the liquid to flow through a small opening, generating a conical spray of which the droplet diameter decreases as the surface of the orifice becomes smaller. Several geometry of the inlet are possible, such as planar, annular, possibly with a swirl chamber. The main limit of this design is the maximum flow rate that can be imposed by the flow pressure, which makes this kind of injector poorly suited for aeronautical engine applications.

- Rotary atomizers: being a mechanism similar to the pressure atomizer, it forces the liquid to flow inside a rotating device (a disk for example) before entering the combustion chamber. At low liquid velocities, beads are generated at the edge of the rotating device; at higher velocities, ligaments are formed and disintegrate themselves into droplets, their size being inversely proportional to the liquid velocity. If these devices can control very tightly the final diameter of the droplet and generate very fine clouds, they are however too brittle and complex to be effectively employed in aeronautical combustion chambers.

- *Airblast* atomizers: these devices exploit the shear effect of an accelerated air flow parallel to the fuel, often enclosing the fuel flow into the air injection veins (this is called assisted disintegration). In principle these mechanisms work at low relative speed and high air flow, as happens for aircraft engines. A complex interaction between the air and fuel provokes the atomization of the fuel and the formation of the droplet cloud. The configuration of the injection streams can be planar or annular, both sharing the same general behaviour.

As the last type corresponds to the most used in aeronautical application, the study of the physical phenomena related to the assisted disintegration will have priority in this work. The plane configuration gives the advantages of a simplification of the observation and measurements, and it is well documented in literature. As the planar or axial geometry of the injector share the same atomization mechanism, the first will be used in both the description and the simulations inside this thesis. The reference experimental configuration is illustrated in figure 1. The liquid is injected between two parallel walls, forming a very thin (respect to the jet width) liquid film. Above and beyond, two parallel gas streams are injected as well, shearing the liquid flow as

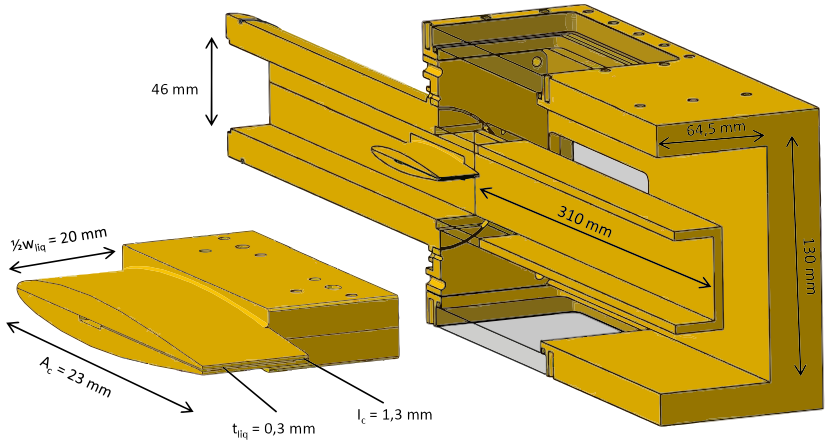


Figure 1: ONERA Lacom experimental installation, from Fernandez [31]: *airblast* planar injector. The liquid is injected through the horizontal fence situated into the profile shaped structure. The gas flows inside the rectangular channel.

they enter the chamber. Following the notation in Trontin [121] and Couderc [18], $2a$ represents the thickness of the liquid sheet. The air flow presents a turbulent velocity profile with a δ thick boundary layer, e representing the thickness of the undisturbed flow, as represented in figure 2. In the 1990s a certain number of experimental researches have been carried on, mainly from Stapper et al [110], Stapper and Samuelsen [109] and Mansour and Chigier [58, 59]. They can be considered as the most representative, and they give the basis for the comprehension of the mechanisms. As numerous parameters determine the regime of atomization, these works explore the dependencies of the atomization characterization on parameters like viscosity, surface tension, thickness of the injection veins and injection velocity of both fluids. The order of magnitude which are retained are a very thin liquid sheet, from $2a = 200$ μm to $2a = 1$ mm, and a wide velocity range, from 1 to 5 m/s for the liquid and from 15 to 60 m/s for the gas. The investigation was focused in particular on the time and space scales of the liquid break-up and the configuration of the final droplet cloud.

The atomization mechanism

The assisted atomization process can be described as the ensemble of mechanisms which occur in the injection of a high pressurized liquid through a small fence (planar or annular) sandwiched by airflows, and its effect on the configuration of the liquid sheet. Two different parts can be distinguished in this process. The first involves the region near the orifice (some diameters/thickness from the injector) where the *primary atomization* take place; the second one is the *secondary atomization* region, which usually extends well further. The first one in particular has undergone different research approaches, starting from linear stability analytical analysis.

Linear stability

A certain number of researches have been oriented towards the linear stability analysis of the sheet oscillation. In particular, in the 1950s, Squire [108], York et al [131], Taylor [117], Hagerty and Shea [37] performed the first incompressible Navier-Stokes temporal linear stability analysis in the case of a liquid sheet injected into still air. The conclusion of these studies were that only two oscillation modes can occur: a sinusoidal anti symmetrical mode and a symmetrical dilatation mode, shown in figure 3. Experiences have put in evidence the strong deformations of the liquid sheet subjected to the sinusoidal mode to not be themselves sufficient to cause the

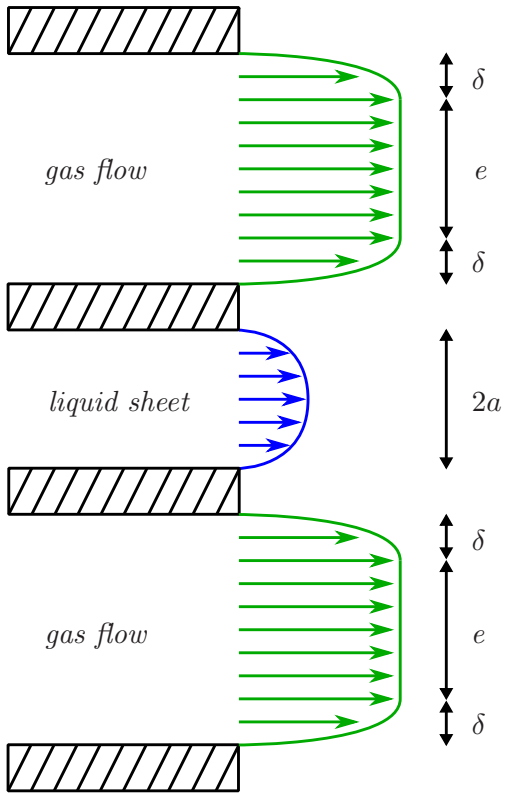


Figure 2: Configuration of the *airblast* injector, resumed from Couderc [18].

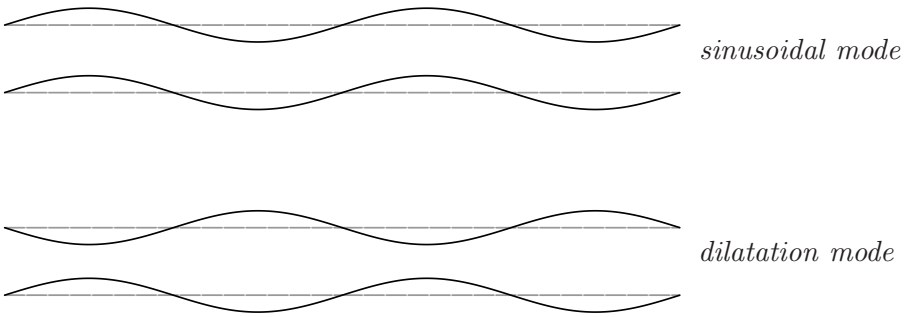


Figure 3: Two possible modes of oscillation for the liquid sheet.

break-up of the sheet. Effectively, the "walls" of the sheet remain parallel in this configuration. The non linear stability analysis of Jazayeri and Li [45] and Rangel and Sirignano [94] give, as answer, that the sheet may be subjected to break-up following a pinch given by a dilatation mode. In particular, the first harmonic of the sinusoidal fundamental is a dilatation mode which provokes the pinch each half wavelength of the fundamental.

The atomization regimes

The *primary atomization* is the results of the shearing instabilities triggered by the interaction between the liquid and gas flows. Stapper and Samuelsen [109] identified two main regimes from their experiences. Both start with same general behaviour: a first oscillation in the longitudinal direction is followed by a transversal one. After exiting the nozzle, the most unstable wavelength

is determined by the thickness of the incoming vorticity layer, formed in the gas flow inside the injector (Gorokhovski and Herrmann [33]), and the gas-liquid density ratio. These longitudinal waves are accelerated by the shearing gas flow, and make the sheet to wave inside and outside the gas flow; in these conditions the sheet is subjected to a Rayleigh-Taylor instability, which manifests itself as a transverse modulation, which leads in turn to the formation of streamwise ligaments. A Rayleigh-Plateau instability finally triggers the formation of the droplets by ligaments break-up (Marmottant and Villermaux [60]). Lozano et al [56] were the first to employ the momentum flux ratio M to express the boundary between the two regimes: this boundary lies at a value of M around 0,5.

In the first mode, called *cellular breakup*, the longitudinal and transversal instability shares similar wavelengths. The longitudinal oscillation causes the sheet to wave in and out the air flow, like a flag in wind; at the same time, the added transversal oscillations generates bag-like structures in the sheet, which collapse quickly into smaller structures and droplets. The second mode is called *stretched streamwise ligament breakup*, and is characterized by a quicker growth of the transversal instabilities face to the longitudinal ones, which effectively do not have time to develop themselves. The oscillation of the sheet is far less evident, because the fragmentation begins almost as soon as the liquid leaves the injector.

Other regimes have been observed at very low values of M. The *ruffled* and *coalescence break-up* were observed in the research of Carvalho et al [15]. The ruffled regime occurs at flow conditions were the liquid velocity is at least one order of magnitude greater than the air velocity. The turbulence ruffles the liquid surface and disintegrate the liquid sheet by perforation far away from the injection point. The coalescence break occurs when the fluid inertia effects are much more smaller than the surface tension ones, and a triangular pattern forms, finishing into a liquid filament. This jet breaks following the mechanics of the Rayleigh-Plateau instability. The *torn sheet break-up* was initially observed by Lefebvre [51]. It was observed that the atomization occurred at the liquid exit, with relatively long break-up lengths and streamwise ligaments lengths. It is observed for the weakest ratio of momentum M.

Mansour and Chigier [58] determined three regions of different behaviours for the same problem. They found the sinusoidal mode to be dominant respect to the dilatation mode for many regimes; however, for low air-liquid velocity ratios this is no more true, as it is the second mode which become dominant. They show in a diagram 4 the frequencies of the sheet oscillation versus the liquid exit velocity, this for different air speeds. This diagram defines three zones, called A, B and C. In the A and B zones the sinusoidal mode is dominant in the vibration of the sheet. In the transition between B and C the two mode coexist, while in C the oscillation is mostly driven by the dilatation mode. In the fist two regions a correlation between the air velocity and the resulting frequency can be found, as the growth of the first engenders the growth of the second. This does not happen in the third region. The B region is considered the best regime for the atomization. In this zone, three stages can be identified in the process. At first, a shearing Kelvin-Helmholtz instability perturbs the plane sheet starting the sinusoidal streamwise oscillation, which manifests itself all along the sheet: this is called the global oscillation. Subsequently transversal deformation waves appear, and start to interact with the aerodynamic forces to generate longitudinal ligaments. The sheet breaks into smaller liquid packs, which are in turn affected by the shearing effect of the air flow. This continuous fragmentation in smaller and smaller structures is the *secondary atomization*, a mechanism which ends in the formation of a spray of droplets. The most influential parameter in this process is the aerodynamic Weber number.

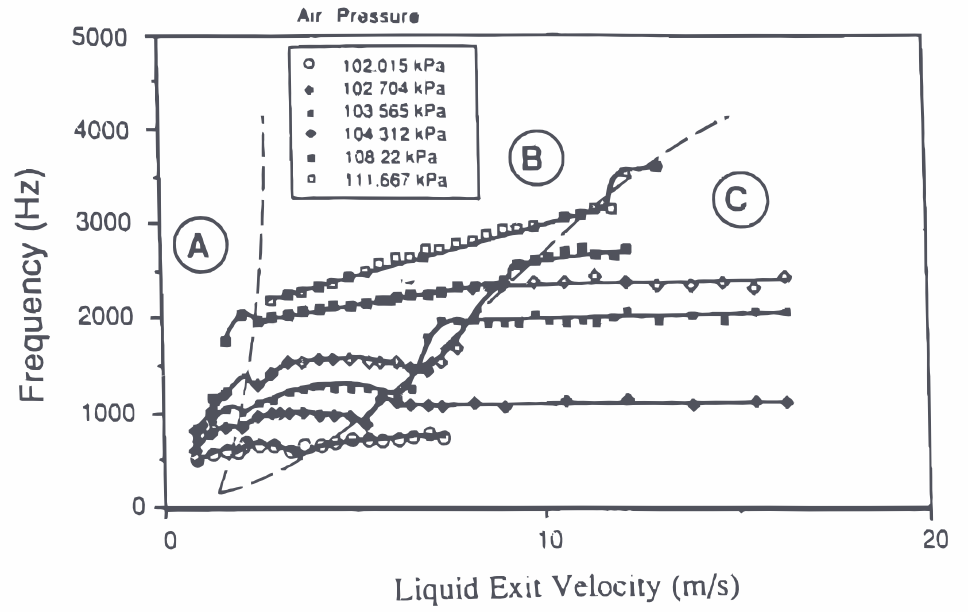


Figure 4: Diagram of Mansour and Chigier [58] (from Couderc [18], Fernandez [31], Trontin [121]) representing the global oscillation frequency versus the liquid injection speed, for different air pressures.

2.1 The global oscillation frequency

This characteristic corresponds to the frequency of the predominant oscillation observed in the liquid sheet. Its magnitude gives a measure of the droplet generation rate and therefore is likely the most studied parameter in the primary atomization. It has been found that the frequency of the global oscillation depends linearly on the velocity of the gas flow. This behaviour having been noticed from different experiences, the dependency upon the liquid velocity is not so well defined. The figure 5 comes from the experiences of Lozano et al [56] with air and water, and it shows a linear dependency of the frequency on the air velocity. The values seem to be reduced by a factor, which allows a non dimensional representation of the curves:

$$\begin{cases} M = \dfrac{\rho_g U_g^2}{\rho_l U_l^2} \\ f^* = \dfrac{fa}{U_g - U_{min}} \end{cases} \tag{1}$$

where a is the sheet thickness, U_l and U_g the velocities of water and air respectively, ρ_l and ρ_g the densities. U_{min} is the minimum airflow velocity, at which the streamwise oscillation is originated. Lozano et al [56] referred to this parameter as minimum air oscillation velocity, a constant value with the liquid velocity. This parameter has been confirmed in Fernandez [31] to vary with the liquid physical properties (with values of about 10 m/s for water, below 4 m/s for ethanol and kerosene) but to remain independent of the liquid velocity. A dimensional analysis in this work suggested that the minimum air velocity was proportional to the fluids surface tension, which damped the oscillation, and inversely proportional to the dynamic viscosity, which enhanced the energy transfer.

$$v_{min} = 2 \times 10^{-4} \frac{\sigma}{\mu_l} \frac{\rho_l}{\rho_g} \tag{2}$$

The so found value of f^* frequency can be plotted against the momentum ratio M, as shown in figure 6, obtaining a compaction of the curves on a single f^* value function of M. Other

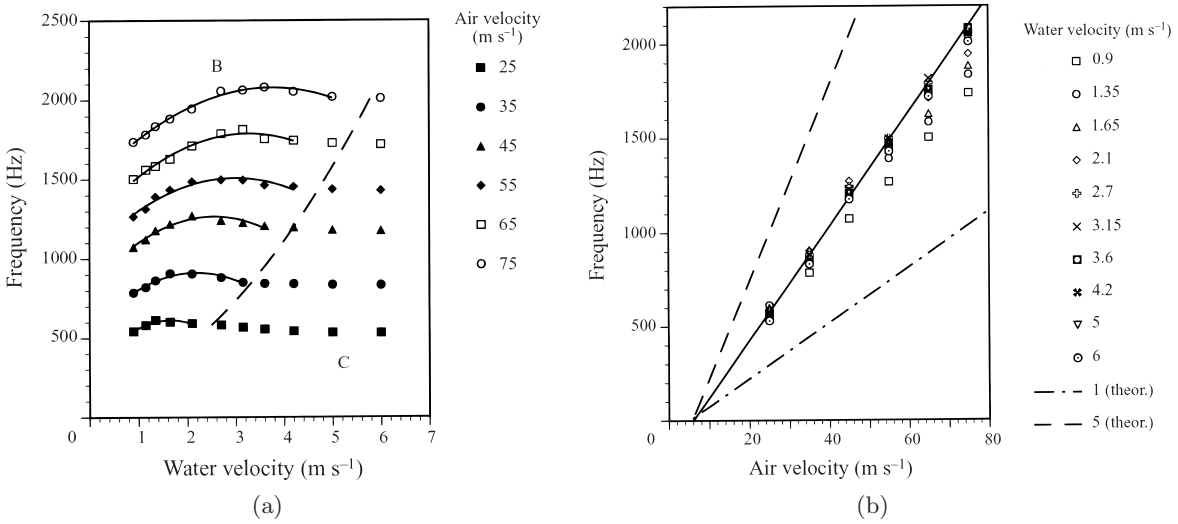


Figure 5: Results from Lozano et al [56] (in Couderc [18], Fernandez [31], Trontin [121]) representing the global oscillation frequency versus the liquid injection speed, for different air pressures. (a) Frequency against air velocity for different liquid velocities (b) Frequency against liquid velocity for different air velocities.

parameters seem to be important in the frequency determination, above all the boundary layer thickness. Its augmentation causes a diminution on the grow rate of the primary wavelength, and a diminution of the wave number which gives the maximum grow rate. Given the typical dimension of the boundary layer, near to the sheet thickness, a study which neglects its influence should be excluded.

3 Numerical simulations

In this section it will be given an overview of the numerical treatment of the jet disintegration. For this problem, the configuration consists of a two phase flow where the two fluid keep separated. Two possible approaches can be distinguished:

- ▷ The interfacial flow approach. In this case each phase occupies its own part of the domain; the Naver-Stokes equations are solved independently in each phase, while in the separation zone (the interface) the stress tensor jump imposes the boundary conditions for each phase.

- ▷ The dispersed phase approach. In this case, one phase is considered as continuous, the second is dispersed inside the previous one. The first phase follows the Navier-Stokes equations. In a pure Eulerian approach, the dispersed phase follows the transport equations of a continuous flow; in a Lagrangian-Eulerian approach, each dispersed phase particle is treated in a Lagrangian way.

For the primary atomization problem, the first approach is more appropriate because of the initial clear separation of the two fluids. For this reason, in this work, it has been chosen, continuing with the development found in Couderc [18] and Trontin [121]. The second method class is more oriented towards the simulation of a set of particles being transported by the flow: it is, for this reason, more suitable to follow the droplet cloud formed during the atomization process, and to simulate the vaporization and combustion processes.

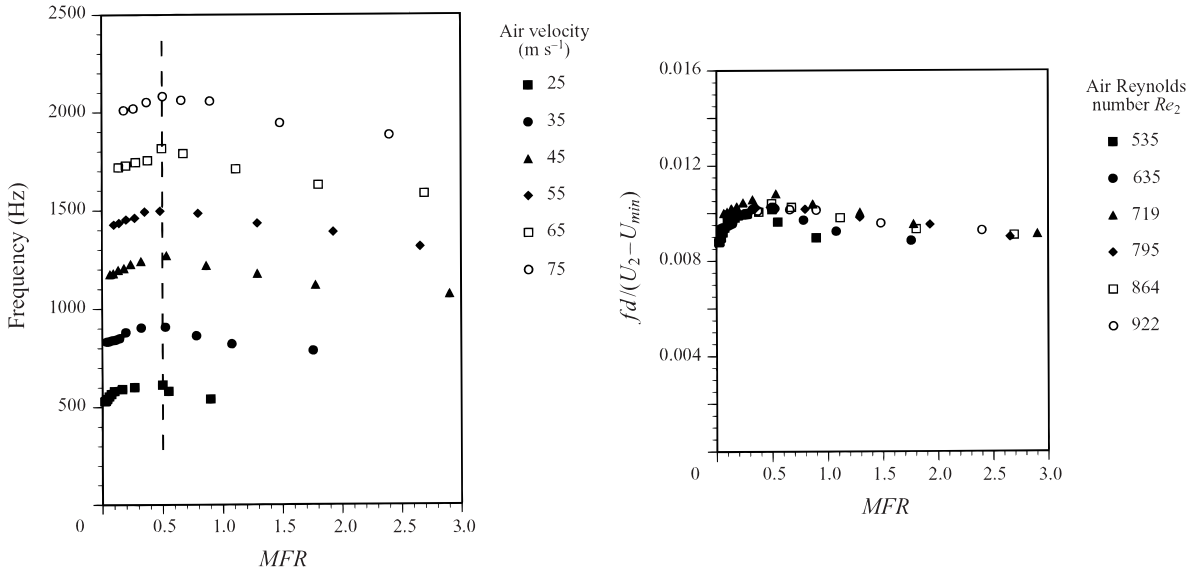


Figure 6: Results from Lozano et al [56] (in Couderc [18], Fernandez [31], Trontin [121]) representing the global oscillation frequency versus the momentum ratio, for different air velocities. (a) Frequency f (b) Non dimensional frequency f^*.

3.1 DNS of primary atomization

It is clear that the Direct Numerical Simulation is more a research tool than a true "bruteforce attack" method for industrial problems. It is mainly employed to understand the physical phenomena which take place in particular problems, as the jet atomization. Most of the primary atomization processes occur at low Mach number, and involves two immiscible fluids. The flow is consequently governed by the variable density incompressible Navier-Stokes equations. The challenges of such approach are multiple:

- ▷ length and time scales can span on several orders of magnitude;
- ▷ the presence of the interface insert discontinuities in the fluid properties;
- ▷ the surface tension is a punctual force which acts at points in general not coincident with nodes;
- ▷ topology changes occur very frequently.

The DNS purpose is to solve all the scales of the flow to avoid any incertitude related to small scale models. Multiphase flows add a smaller scale than the Kolmogorov's, corresponding to the size of the smallest liquid structure. When the topology is subjected to extensive deformations, this scale may become really small, as the smallest ligaments approaches its break-up. In general, the size of the smallest droplets determines the cell size (about 2-5 cells per droplet from Gorokhovski and Herrmann [33]). Fortunately, this is only necessary on the surroundings of the interface, naturally asking for an interfacial mesh adaptation.

The constraints in terms of maximum Reynolds and the large difference of scales in the atomization prevent the simulation of realistic conditions without reaching a compromise in terms of resolution. The under resolution of the meshes may lead to erroneous solutions which can logically appear as "true". This is the typical case of the coalescence and break-up, which

are strongly affected by the accuracy of the mesh. Again, an obvious remedy is an adaptation of the mesh to the needs of the computation. In literature, interfacial methods have seen a great development in the last ten years, treating a wide range of problems such as jet break-up, bubble dynamics, free surface flows and organic flows. In particular, two main families have evolved in parallel, the Level Set methods of Osher and Fedkiw [79] and the Volume of Fluid originally developed in DeBar [22]. They present diametrically opposed merits and faults, and a certain number of works have tried to use them together, as M. Sussman [57] and Menard et al [67]. They will be detailed more deeply in the dedicated chapter 3.

Remarkable progresses in the atomization simulation have been done in particular by the CORIA laboratory in Rouen, starting from the bubble coalescence and break-up, continuing with the natural disintegration of jets and finally with the investigation of a full break-up of a turbulent liquid jet with a Level Set/VOF/Ghost Fluid method (developed in Menard et al [67]) in Lebas et al [50], in 2009 (figure 7). The method is compared to an Eulerian/Lagrangian Spray Atomization (ELSA) model in a Diesel atomization problem: a good agreement is found when they are compared in the dense zone of spray.

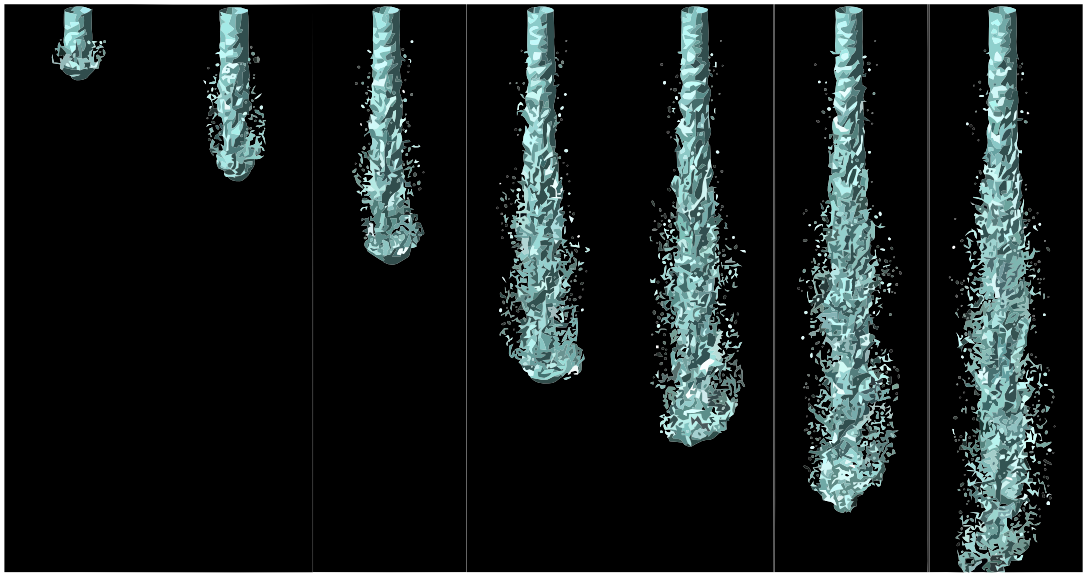


Figure 7: Interface location for the sheared liquid jet in Menard et al [67]

Desjardin et al [24] developed an accurate Level Set method which conserves mass discretely. This accurate conservative level set (ACLS) approach combines the concept of conservative Level Set with high-order implicit transport schemes in order to conserve mass while maximizing accuracy and robustness. This method has been successfully applied in the computation of the temporal simulation of the atomization of a diesel jet with realistic material properties (figure 8). Even if the Reynolds and Weber numbers were decreased to make the computation affordable, the main atomization processes were observed: capillary waves, ligament and drop formation, air entrainment. For this diesel jet, the liquid fuel was injected at high speed in quiescent air, and the momentum was mainly transferred from the liquid to the gas.

Moreau and Desjardins [72] pursued this way by implementing an higher order Ghost Fluid method. They show results of a round jet of low-speed water atomized by an annular high-speed air stream (figure 9). This experiment has been used extensively by Marmottant and Villermaux [60] to study the break-up regimes and the mechanisms of the formation of drops and ligaments. Their momentum ratio is $M = 1.76$.

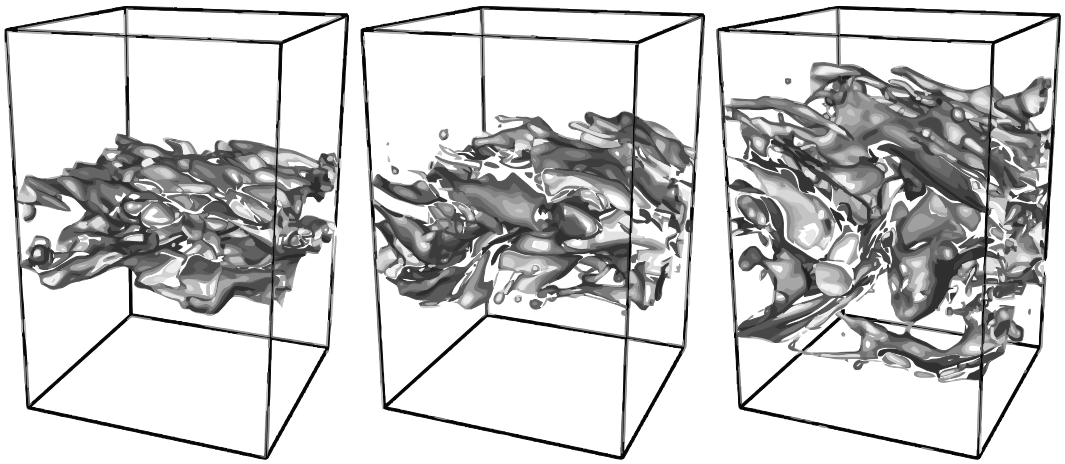


Figure 8: Interface location for the liquid jet in Desjardin et al [24]

Figure 9: Interface location for the sheared annular liquid jet in Moreau and Desjardins [72]

3.2 The mesh adaptation

As already mentioned before, the adaptive mesh seems a natural choice for such a problem where very different scales coexist, and are very difficult to treat in a DNS sense. The main idea is that the resolutions needed to accurately capture the interfacial interactions can be used only locally, allowing a cheaper resolution far from the interface.

A lot of research has been done around the problem of adapting the incompressible Navier-Stokes solvers to a non uniform mesh. In 1984 Berger [9] put the basis for the structured mesh adaptive mesh refinement (AMR). The principle was that different mesh of different cell size can be solved independently, given appropriate boundary conditions. A system of hierarchical grids can build a mesh of which the resolution can vary in space and time. After each time step new grids can be inserted or deleted; on the new ones the initial condition is given by the old underlying coarser grid.

A lot of efforts were put in the adaptation of the projection methods to the hierarchical AMR. The incompressible flow case is essentially more difficult for local refinement methods, particularly in the presence of refinement in time. In that case, it is not entirely obvious how to represent the divergence-free constraint. The refinement in time (also called time sub-cycling) is the possibility to advance each level at its own time step (given by the local cell-size-driven

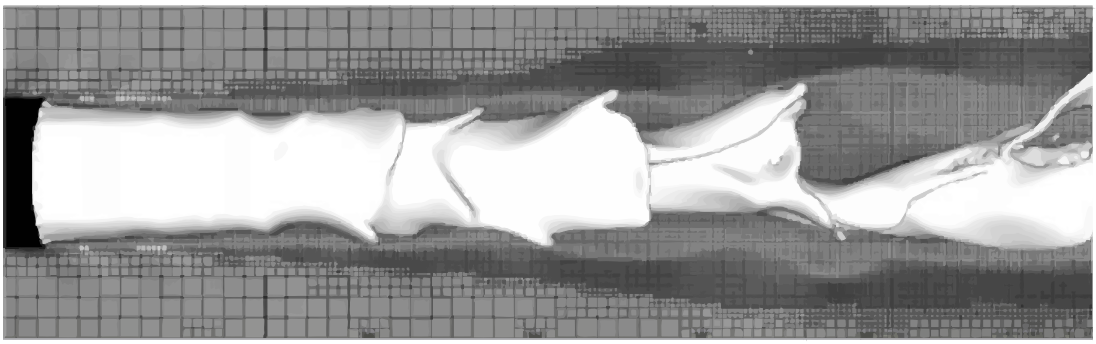


Figure 10: Liquid jet disintegration with the *Gerris* code by Popinet and Zaleski [88].

CFL), enforcing consistency with synchronization steps. A series of works investigated this topic: Almgren et al [2], Bell et al [8], Howell and Bell [43] treated different important aspects of the Navier-Stokes solution, in particular the conservative and free-stream preservation aspects.

In the work of Almgren et al [3] introduced the variable density computation, stressing the issue of conservation. The interfacial flow have naturally followed from these Navier-Stokes adaptive solvers. Sussman [111], Sussman et al [114] developed a series of adaptive projection methods based on a Level Set/VOF/ Ghost fluid method to treat the two phase aspects. The CORIA code of Menard et al [67] is at the moment being adapted to a Berger's AMR as for the Sussman's code.

As alternative to the Berger's based algorithm, other kind of adaptive meshes have proven their effectiveness on the atomization process: the *Gerris* code from Popinet [87] has been recently applied to the jet disintegration problem, using a VOF method and a local cell refinement (which will be detailed later in chapter 5). Other works as the One Cell Local Multigrid of Vincent and Caltagirone [127] have not yet faced the atomization problem, but show interesting capabilities in the efficient solution of two phase flows.

Another topic related to the reduction of the computational needs of those simulations is the parallel computation. If for a classical code its utilization is straightforward, for a dynamically evolving mesh an efficient repartition is no longer evident. Some kind of AMR are designed to be parallel (possibly giving up on some others features). The PARAMESH (Peter MacNeice and Packer [85]) and SAMRAI ([41]) packages are examples of parallel design for any kind of AMR application.

4 Objectives of the thesis

The aim of this work is to study the application of parallel adaptive mesh refinement to a two phase DNS code. The idea is to keep a strong resolution near to the interface where the smaller scales have to be resolved, but allow a coarser mesh size inside each phase to soften the computational workload. At the same time the global accuracy of the simulations should not be worsened nor its convergence rates hampered. The general requirement would finally be to always have a gain in the solution accuracy when a refinement level is added, and a reduction in the computational time and/or resources, depending on the problem to be studied.

The numerical resolution is taken from Couderc [18] with only small modifications, and it consists in a projection method for the Navier-Stokes equations and a coupled Level-Set/Ghost Fluid method for the interface treatment. This algorithm allows the resolution of the flow independently in each fluid, with the contributions of the interface added as forcing terms. The implementation of the adaptive mesh refinement has been performed with the help of the open source libraries PARAMESH of Peter MacNeice and Packer [85], Olson and MacNeice [78], Olson

[77].

4.1 Outline of the thesis

The first chapters of the thesis are dedicated to the description of the physical and numerical model of the DNS solver. Chapter 1 contains the description of the equations and the associated hypothesis, in particular the physical effects which are considered in the computation, as the surface tension and the viscosity. Chapter 2 describes the single grid solver, the projection method for the single phase Navier-Stokes incompressible equations, which can be solved in each fluid when the jump conditions give appropriate boundary conditions to the two zones. In chapter 3 the interface tracking methods are reviewed, and the Level Set approach is described, as well as the numerical resolution of the associated transport equation; it also illustrates the treatment of the jump conditions, performed via Ghost Fluid method.

The following two chapters are dedicated to the implementation of the adaptative mesh. In chapter 4 an overview of the AMR techniques is given first; then, more details are reserved for the methods which approach the desired characteristics (structured mesh, hierarchical...). A precise description of the code AMR implementation and the PARAMESH package follows (Peter MacNeice and Packer [85], Olson and MacNeice [78], Olson [77]), with the set of mesh operations which are associated. Chapter 5 is dedicated to the elliptic adaptive solver alone, with continuous references to the literature strategies for this interesting and (very) challenging problem.

The last two chapters are dedicated to the code results and performances. Results of validation in terms of accuracy, AMR speed-up and parallel speed-up are contained in chapter 6, with reference to analytical solution where available or literature results elsewhere. Chapter 7 describes the preliminary simulations of two dimensional primary atomization, with results in terms of global oscillation frequency. A very high resolution simulation has been performed too, in order to show the capability of the code and to have some reference solution for the two dimensional computations, where a more realist behaviour can be observed.

Chapter 1

Physical Model

Contents

Introduction

In this chapter it will be presented the physical model which has been employed to describe the two phase interfacial flows, and in particular the disintegration of a liquid jet. The two phase aspect adds different contributions to the motion and interaction of the two fluids, such as the capillarity force and the viscous stresses. However, a certain number of hypothesis should be made to allow an easier numerical resolution of the equations.

1 Governing equations

The first consideration to be done concerns the scale and the nature of the physical phenomenon to be studied. In the case of fluid observation, the ratio between the mean free path and the characteristic length, known as the *Knudsen* number, determines whether a continuum ($Kn \leq 0.1$) or a discrete model ($Kn > 0.1$) of the fluid itself is possible. In the fist case the physical quantities are defined for a small control volume, and obtained by taking the mean of all the molecules contained in the volume. On the other case, it is not possible to neglect the strong variation of physical properties of the molecules and atoms; if they are rarefied enough, then a particle based approach must be employed. The fluid model used in this work is always a continuum model; the term fluid may without distinction refer to a liquid or a gas.

1.1 Mass conservation

Under the continuum hypothesis, the Navier-Stokes equations describe the properties of the fluids as functions of space and time $(\mathbf{x}, t)$, which is from an Eulerian point of view. The first

equation describes the temporal evolution of the fluid mass, or the density ρ integrated at each instant of time in the control volume Ω included in the surface $\partial\Omega$. The conservation of the matter implies that, in absence of chemical reactions, each variation in time of the mass is balanced by the net flux crossing the surface

$$\frac{d}{dt}\int_{\Omega} \rho \; dV = -\int_{\partial\Omega} \rho.\boldsymbol{u} \; dS \tag{1.1}$$

where V and S are the unit of volume and surface, $\boldsymbol{n}$ is the normal vector oriented outside the surface. The Gauss-Ostrogradsky theorem affirms that the outward flux of a vector field through a closed surface is equal to the volume integral of the divergence on the region inside the surface. It allows the transformation of the right hand term into a volume integral:

$$\int_{\Omega} \left(\frac{\partial \rho}{\partial t} + \nabla.(\rho \boldsymbol{u}) \right) \; dV = 0 \tag{1.2}$$

Under the hypothesis of a continuous function, this can be reduced to the local equation of mass conservation:

$$\frac{\partial \rho}{\partial t} + \nabla.(\rho \boldsymbol{u}) = 0 \tag{1.3}$$

1.2 Momentum conservation

In a similar way, the momentum $(\rho \boldsymbol{u})$ of the fluid particle of volume Ω and surface $\partial\Omega$ follows the Newton second law, which states that the net force on a particle is equal to the time rate of change of its linear momentum:

$$\frac{d}{dt}\int_{\Omega} \rho \boldsymbol{u} \; dV = \int_{\Omega} \rho f \; dV + \int_{\partial\Omega} \boldsymbol{T}.\boldsymbol{n} \; dS \tag{1.4}$$

The tensor $\boldsymbol{T}$ contains the surface forces, while the vectorial function f contains the volume forces, like gravity. The temporal derivative is a Lagrangian derivative, as it is written in a reference system which is placed on the volume particle and follows it. In order to return to an Eulerian description, the total derivative definition is applied:

$$\frac{d\boldsymbol{u}}{dt} = \frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u}.\nabla)\boldsymbol{u} \tag{1.5}$$

By using the Gauss-Ostrogradsky theorem again, the equation 1.4 becomes

$$\rho \left(\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u}.\nabla)\boldsymbol{u} \right) = \nabla.\boldsymbol{T} + f \tag{1.6}$$

1.3 Fluid assumptions

Some assumptions can be made on the general form of the Navier-Stokes equations in order to simplify them. At first, given the typical velocities that appear in the liquid sheet disintegration are largely inferior to the speed of sound, both fluids can be considered as incompressible. The volumetric mass can be considered as constant in space and time, with a step discontinuity localized between the fluids. The time derivative of equation 1.3 becomes equal to zero, and the equation itself degenerate to the zero divergence condition

$$\nabla.(\rho \boldsymbol{u}) = 0 \tag{1.7}$$

The description of the surface forces is contained in the $\boldsymbol{T}$ tensor. It can be divided in two parts, the first including the isotropic normal stresses corresponding to the thermodynamic pressure, and the anisotropic stresses caused by the fluid viscosity:

$$\boldsymbol{T} = -p\mathbf{I} + \mathbf{D} \tag{1.8}$$

If the Newtonian fluid model is used together with the incompressibility assumption, the $\mathbf{D}$ tensor elements e_{ij} depend on the instantaneous deformation values:

$$\boldsymbol{D}_{ij} = 2\mu e_{ij} = \mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) \tag{1.9}$$

2 Interface and jump conditions

Once the equations have been written for the for the two fluids, the two phase aspects must be dealt with. The computational domain is therefore divided in two regions (figure 1.1), which are separated by an interface. The interface as it is considered in this work is a separation zone of negligible thickness [1], where the function representing the physical quantities are discontinuous (or their derivatives are), so that the fluid properties jump instantaneously in space. This is reflected in the model by the imposition of jump conditions across an infinitely thin interface. For a space of dimension n, the interface is an object of dimension $n-1$.

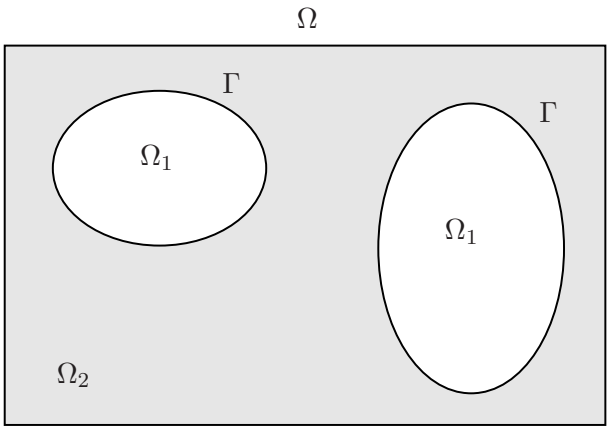


Figure 1.1: Representation of two fluids Ω_1 and Ω_2 and the interface which separates them, $\Omega = \Omega_1 \cup \Omega_2 \cup \Gamma$.

2.1 Surface tension

The cohesive Van Der Waals forces between liquid molecules are responsible for the force known as surface tension. The molecules inside the fluid share their cohesive force with all the surrounding neighbouring molecules, resulting subject to a zero net total force. But molecules at the surface do not have other like molecules on all sides of them, and consequently they receive a net force which pulls them inwards. The inter molecular attractive forces on the surface are enhanced to achieve force balance: this enhancement is called surface tension. In energetic terms, the surface tension balance the energy loss due to the missing molecular bonds on the surface of the fluid; the minimization of this surface energy is the reason for the spherical shape of a droplet in absence of gravity, as it is the shape which minimizes the free surface. One of the

[1] Other approaches may consider it of finite, even if very small, thickness.

consequences of this phenomenon is the generation of an overpressure inside the concave side of the surface. This difference of pressure is predicted by the Laplace's law

$$[p] = \sigma \left(\frac{1}{R_1} + \frac{1}{R_2} \right) = \sigma\kappa \tag{1.10}$$

where $R_{1,2}$ are the main curvature radius, κ the associated curvature and σ the surface tension coefficient, which depends on the difference of the molecular bond energy and $[p] = p_1 - p_2$ the pressure jump. A bubble or a droplet are always in an overpressure situation in comparison to the external environment.

2.2 Viscosity

The Laplace's law does not fully describe the jump conditions when the interface is moving: under the hypothesis of Newtonian fluids the viscous forces act on the interface, depending on the velocity derivatives. The tensor $\boldsymbol{T}$ contains these surface forces, their jumps being defined by

$$\begin{cases} \boldsymbol{n}.[\boldsymbol{T}].\boldsymbol{n} = \sigma\kappa \\ \boldsymbol{t}.[\boldsymbol{T}].\boldsymbol{n} = 0 \end{cases} \tag{1.11}$$

where $[\boldsymbol{T}] = \boldsymbol{T}_1 - \boldsymbol{T}_2$ represents the jump between the two fluids. The viscous stresses on the interface impose the continuity of the tangential velocity, while the conservation of mass imposes the continuity of the normal component, so that

$$[\boldsymbol{u}] = 0 \tag{1.12}$$

The balance of the tangential velocities comes from the balance of the stress force applied on

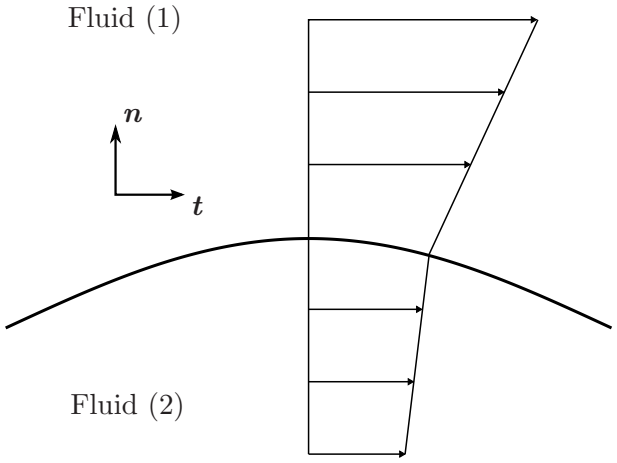


Figure 1.2: Representation of two the tangential velocity profile, continuous on the interface but discontinuous in the first derivative.

the two sides of the interface, which are the product of the dynamic viscosity and the velocity gradient, which describes the jump in the first derivative of tangential velocity on the interface, as depicted in figure 1.2.

3 The final model

The final model consists into the incompressible Navier-Stokes equations solved in each of the two fluids. The computational domain is divided in the corresponding two regions separated by a sharp interface as in figure 1.1. In each of them the model is:

$$\begin{cases} \nabla \cdot \boldsymbol{u} = 0 \\ \dfrac{\partial \boldsymbol{u}}{\partial t} + (\nabla . \boldsymbol{u})\, \boldsymbol{u} = \dfrac{1}{\rho} (\nabla \cdot \mathbf{T} + f) \end{cases} \tag{1.13}$$

With the following hypothesis:

$$\begin{cases} \mathbf{T} = -p\mathbf{I} + \mathbf{D} \\ \mathbf{D} = \mu \left(\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^T \right) \end{cases} \tag{1.14}$$

where $\boldsymbol{u} = [u, v]^T$ represents the velocity vector field, p the hydrodynamic pressure, ρ the density, μ the dynamic viscosity of the fluid and f the external forces like gravity. Across the interface, the jump conditions are imposed by capillarity and viscosity; the velocity respects a no-slip condition:

$$\begin{cases} [p] - \boldsymbol{n}. \left[\mu \left(\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^t \right) \right] . \boldsymbol{n} = \sigma \kappa \\ \boldsymbol{t}. \left[\mu \left(\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^T \right) \right] . \boldsymbol{n} = 0 \\ [\boldsymbol{u}] = 0 \end{cases} \tag{1.15}$$

Here the notation $[\cdot] = (\cdot)_1 - (\cdot)_2$ represents the jump across the interface, $\boldsymbol{n}$ the normal vector pointing from 1 to 2, $\boldsymbol{T}$ the tangential one. σ is the surface tension, κ the interface curvature.

Conclusion

In this chapter the physical model for the two phase interfacial flows simulation has been presented. The Navier-Stokes equations are written in the simplified form given by the incompressible and Newtonian flow hypothesis. The equations are solved for each fluid in two distinct domains, divided by a sharp interface. On the interface act surface tension and viscous forces; the jump conditions allow the definition of reciprocal boundary conditions for the two fluids.

Chapter 2

Single Grid Solver

Contents

Introduction

In this chapter the algorithm to solve the incompressible two phase Navier-Stokes equations is described. This algorithm and its discretization will be maintained unaltered within the adaptive mesh refinement structure, thanks to the AMR block implementation. The main difference lies in the resolution of the equations on a composite mesh, consisting in an union of sub-meshes of different cell size, generated by the adaptive mesh algorithm. The additional operations needed to deal with the refinement jumps can be applied in general as *a posteriori* corrections to the unmodified single grid algorithm.

1 The resolution of Navier-Stokes equations

In this section the numerical method used to solve the Navier-Stokes momentum equations (2.2) is presented. At first the attention is focused on the resolution of a single phase flow; the treatment of the two phase jump conditions will be added later on. Numerical test of the solver and its behaviour with the adaptive mesh will be presented in chapter 6. The unknown of the equation system are the pressure and velocity field $(p, \boldsymbol{u})$, in each phase. Some assumptions on the equations have already been done in the previous chapter: in particular, the assumption of

constant density (2.1) imposes the choice of the numerical method among the incompressible solvers. If some advantages come from this formulation, such as a less restrictive time step, some difficulties arise as well. For example, the boundary conditions are not easy to define. Numerically, an elliptic equation must be solved, adding an expensive iterative solver to the algorithm; this problem becomes more and more difficult to solve as the composite mesh is used and the two phase aspect is considered.

$$\begin{cases} \nabla \cdot \boldsymbol{u} = 0 & (2.1) \\ \dfrac{\partial \boldsymbol{u}}{\partial t} + (\nabla \boldsymbol{u})\, \boldsymbol{u} = \dfrac{1}{\rho} (\nabla \cdot \mathbf{T} + f) & (2.2) \end{cases}$$

Boundary conditions are needed to close the system; their description is given later in this chapter.

There are, in literature, different methods to solve this system of equation. Among these are the fundamental pressure correction methods (SIMPLE, SIMPLER), the artificial compressibility methods, the penalization methods. However, one of the most widely spread algorithms are the projection or predictor-corrector methods, developed by Chorin [16] and Temam [119]. This procedure needs two resolution of equations plus a correction, when the variables are arranged in a staggered way. For this reason the variables in the code are so located, the pressure lies on the cell center, the u component of the velocity vector on the right cell face and the v component on the upper one (figure 2.1). The projection method comes from the Helmholtz-Hodge theorem,

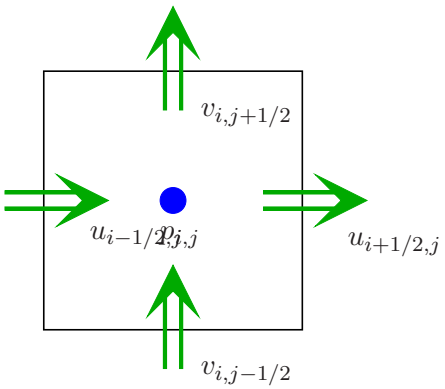


Figure 2.1: Position of the unknowns on the staggered mesh.

which states that any velocity field $\boldsymbol{u}$ defined on a simply connected domain can be uniquely decomposed into a divergence free (solenoidal) part $\boldsymbol{u}_{sol}$ and an irrotational part $\boldsymbol{u}_{irrot}$. Given the mathematical property $\nabla \times \nabla \Psi = 0$, the velocity vector can be rewritten as the solenoidal part plus the gradient of a scalar function:

$$\boldsymbol{u} = \boldsymbol{u}_{sol} + \boldsymbol{u}_{irrot} = \boldsymbol{u}_{sol} + \nabla . \Psi \tag{2.3}$$

If the divergence of equation (2.3) is taken, the results is that, since $\nabla . \boldsymbol{u}_{sol} = 0$, the divergence of the velocity is equal to the Laplacian of the potential:

$$\nabla . \boldsymbol{u} = \Delta \Psi \tag{2.4}$$

This is the origin of the Poisson equation associated with this method, where the pressure is related to the potential Ψ through a factor Δt.

1.1 Projection method

The Chorin projection method, as it was at first developed, de-couples velocity and pression by using first a totally explicit discretization for the effects of advection and diffusion in a predictor

step, and then enforcing the compliance with a velocity divergence constraint obtained in a subsequent elliptic equation solve. The algorithm to advance the velocities from time t to time $t + \Delta t$ can be summarized as follows:

▷ Prediction step: resolution of the momentum equation without pressure term

$$\boldsymbol{u}^* = \boldsymbol{u}^n + \Delta t^n \left(-\left(\boldsymbol{u}^n.\nabla\right)\boldsymbol{u}^n + \frac{\mu}{\rho}\nabla.\left(\nabla\boldsymbol{u}^n + \left(\nabla\boldsymbol{u}^n\right)^T\right)^n \right) \tag{2.5}$$

▷ Pressure computation: resolution of a Poisson equation with the divergence of the predicted velocities as right hand side

$$\nabla.\left(\frac{1}{\rho}\nabla p^{n+1}\right) = \frac{\nabla.\boldsymbol{u}^*}{\Delta t^n} \tag{2.6}$$

▷ Correction step: correction of the intermediate velocities by the pressure gradient

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^* - \frac{\Delta t^n}{\rho}\nabla p^{n+1} \tag{2.7}$$

This results in a non linear parabolic advection equation for the momentum without pressure terms (non incremental form) and an elliptic equation for the pressure to solve. The third step is a correction of the velocity vector $\boldsymbol{u}^*$ with the pressure gradient to make it a zero divergence field. The temporal convergence order of this original algorithm is, following the work of Guermond et al [35], Minion et al [71], an $O(\Delta t)$ for both velocity and pressure, when using Dirichlet boundary conditions for $\boldsymbol{u}^n$ and $\boldsymbol{u}^*$ and homogeneous Neumann for the second. If the order of convergence can be raised for the velocity with a more accurate $\partial\boldsymbol{u}/\partial t$ discretization, the imposition of the numerical boundary layer from the $\boldsymbol{n}\nabla.p^{n+1}$ condition does not allow the same for the pressure. It has been found in the same works that the order of convergence can be raised to two by two means, the first being to incorporate in the prediction step the gradient of pressure ∇p^n (incremental form), the second an implicit treatment of the viscous terms $\boldsymbol{T}^n$ (which should in this case be called $\boldsymbol{T}^*$). However, in order to avoid the construction of the linear system associated to this implicit solution, an explicit form is maintained.

1.2 Numerical resolution

The first approximation

The discretization on the staggered mesh of the equations (2.5), (2.6), (2.7) becomes respectively:

▷ Prediction step

$$\begin{aligned}
u^*_{i+\frac{1}{2},j} &= u^n_{i+\frac{1}{2},j} + \Delta t^n \left(-\left(u^n\frac{\partial u^n}{\partial x} + v^n\frac{\partial u^n}{\partial y}\right)_{i+\frac{1}{2},j} + \frac{\mu}{\rho}\left(\frac{\partial^2 u^n}{\partial x^2} + \frac{\partial^2 u^n}{\partial y^2}\right)_{i+\frac{1}{2},j}\right) \\
v^*_{i,j+\frac{1}{2}} &= v^n_{i,j+\frac{1}{2}} + \Delta t^n \left(-\left(u^n\frac{\partial v^n}{\partial x} + v^n\frac{\partial v^n}{\partial y}\right)_{i,j+\frac{1}{2}} + \frac{\mu}{\rho}\left(\frac{\partial^2 v^n}{\partial x^2} + \frac{\partial^2 v^n}{\partial y^2}\right)_{i,j+\frac{1}{2}}\right)
\end{aligned} \tag{2.8}$$

▷ Pressure step

$$\frac{\partial}{\partial x}\left(\frac{1}{\rho}\frac{\partial p^{n+1}}{\partial x}\right)_{i,j} + \frac{\partial}{\partial y}\left(\frac{1}{\rho}\frac{\partial p^{n+1}}{\partial y}\right)_{i,j} = \frac{1}{\Delta t^n}\left(\frac{u^*_{i+\frac{1}{2},j} - u^*_{i-\frac{1}{2},j}}{\Delta x} + \frac{v^*_{i,j+\frac{1}{2}} - v^*_{i,j-\frac{1}{2}}}{\Delta y}\right) \tag{2.9}$$

▷ Correction step

$$\begin{aligned} u_{i+\frac{1}{2},j}^{n+1} &= u_{i+\frac{1}{2},j}^{*} - \frac{\Delta t^n}{\rho_{i+\frac{1}{2},j}} \left(\frac{p_{i+1,j}^{n+1} - p_{i,j}^{n+1}}{\Delta x} \right) \\ v_{i,j+\frac{1}{2}}^{n+1} &= v_{i,j+\frac{1}{2}}^{*} - \frac{\Delta t^n}{\rho_{i,j+\frac{1}{2}}} \left(\frac{p_{i,j+1}^{n+1} - p_{i,j}^{n+1}}{\Delta y} \right) \end{aligned} \tag{2.10}$$

The pressure equation

A standard five points, second order centered discretization is used for the Laplacian operator in equation (2.9):

$$\begin{aligned} \frac{\partial}{\partial x}\left(\frac{1}{\rho}\frac{\partial p^{n+1}}{\partial x}\right)_{i,j} + \frac{\partial}{\partial y}\left(\frac{1}{\rho}\frac{\partial p^{n+1}}{\partial y}\right)_{i,j} =& \frac{1}{\rho_{i+\frac{1}{2},j}}\left(\frac{p_{i+1,j}^{n+1} - p_{i,j}^{n+1}}{\Delta x^2}\right) - \frac{1}{\rho_{i-\frac{1}{2},j}}\left(\frac{p_{i,j}^{n+1} - p_{i-1,j}^{n+1}}{\Delta x^2}\right) \\ &+ \frac{1}{\rho_{i,j+\frac{1}{2}}}\left(\frac{p_{i,j+1}^{n+1} - p_{i,j}^{n+1}}{\Delta y^2}\right) - \frac{1}{\rho_{i,j-\frac{1}{2}}}\left(\frac{p_{i,j}^{n+1} - p_{i,j-1}^{n+1}}{\Delta y^2}\right) \end{aligned} \tag{2.11}$$

Under the hypothesis $\Delta x = \Delta y$, this discretization leads to a linear system with a symmetrical matrix of five diagonals in 2D. The resolution of this equation is detailed in the dedicated chapter 5, where all the problems that arises with the use of the adaptive mesh are discussed.

The temporal derivative

There are multiple choices for the temporal resolution of a partial derivative equation. The advection/diffusion equation is clearly simpler than the full Navier-Stokes system. Among the different methods, the Runge-Kutta and the linear multi-step methods are the most spread. The more performing high order Runge-Kutta methods have a severe drawback, however: they require as many applications of the methods as the desired order of accuracy. This would mean, for the incompressible solver, multiple solving of the pressure equation, which, in turn, costs on average more than three quarter of the total computational time (up to more than 95% in the most difficult cases). This is not a problem for the multi-step methods, where the results at the previous time steps, instead of computed intermediate ones, are used to improve the accuracy of the solution.

The Adams-Bashford scheme belong to this category. It numerically solves the problem

$$\frac{\partial \boldsymbol{u}}{\partial t} = P(\boldsymbol{u}) \tag{2.12}$$

where $P(\boldsymbol{u})$ is the projection operator applied to the velocity $\boldsymbol{u}$. The time integration gives:

$$\boldsymbol{u}^{n+1} - \boldsymbol{u}^n = \Delta t^n \int_{t^n}^{t^{n+1}} P(\boldsymbol{u}(t))\, dt \tag{2.13}$$

One approximation of $P(\boldsymbol{u})$ between the two instants of time can be obtained by a Lagrange polynomial fit:

$$L(t) = P(\boldsymbol{u}^{n-1})\left(\frac{t - t^n}{t^{n-1} - t^n}\right) + P(\boldsymbol{u}^n)\left(\frac{t - t^{n-1}}{t^n - t^{n-1}}\right) \tag{2.14}$$

so that the integral (2.13) can be approximated as

$$\begin{aligned}\int_{t^n}^{t^{n+1}} L(t)\ dt &\approx \frac{\Delta t^n}{2}\left(L(t^{n+1}) + L(t^n)\right)\\ &\approx \frac{\Delta t^n}{2}\left(P(\boldsymbol{u}^{n-1})\left(\frac{t^{n+1}-t^n}{t^{n-1}-t^n}\right) + P(\boldsymbol{u}^n)\left(\frac{t^{n+1}-t^{n-1}}{t^n-t^{n-1}}\right) + P(\boldsymbol{u}^n)\right)\\ &\approx \frac{\Delta t^n}{2}(P(\boldsymbol{u}^n)\left(\frac{\Delta t^n + 2\Delta t^{n-1}}{\Delta t^{n-1}}\right) - P(\boldsymbol{u}^{n-1})\left(\frac{\Delta t^n}{\Delta t^{n-1}}\right)\end{aligned} \tag{2.15}$$

which allows the writing of the final form of the Adams-Bashford scheme described in Couderc [18]:

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^n + \frac{\Delta t^n}{2}\left(P(\boldsymbol{u}^n)\left(\frac{\Delta t^n + 2\Delta t^{n-1}}{\Delta t^{n-1}}\right) - P(\boldsymbol{u}^{n-1})\left(\frac{\Delta t^n}{\Delta t^{n-1}}\right)\right) \tag{2.16}$$

One last approximation, $\Delta t^n \approx \Delta t^{n-1}$, leads the simplified form

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^n \ + \ \frac{\Delta t^n}{2}\left(3\ P(\boldsymbol{u}^n) - P(\boldsymbol{u}^{n-1})\right) \tag{2.17}$$

The advection terms

The spatial derivatives of the velocity of the non linear advective term $\boldsymbol{u}.\nabla\boldsymbol{u}$ is discretized using the fifth order WENO scheme from Shu [104]. The same are used for the linear advection equation of the Level Set, the difference being that the advected term is here the velocity vector itself, not a passive scalar. The details about this discretization are given in chapter 3, where the linear advection equation solver is presented. For the velocities it is however necessary to interpolate the u and v components as shown in figure 2.2 because of the staggered mesh and the location of the advected variables.

The viscous terms

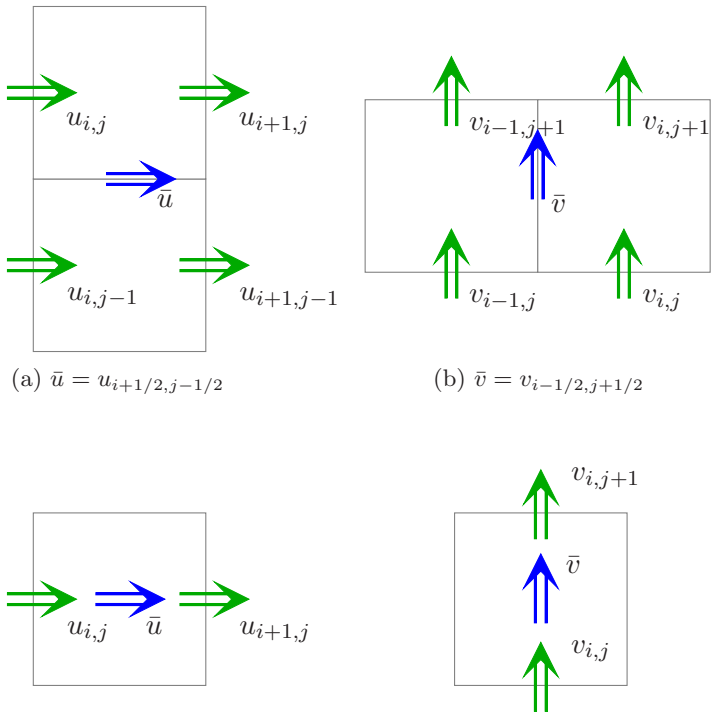
The discretization of the viscous terms is totally explicit second order centered scheme

$$\begin{aligned}\left(\frac{\partial^2 u^n}{\partial x^2} + \frac{\partial^2 u^n}{\partial y^2}\right)_{i+\frac{1}{2},j} &= \frac{u^n_{i+\frac{3}{2},j} - 2u^n_{i+\frac{1}{2},j} + u^n_{i-\frac{1}{2},j}}{\Delta x^2} + \frac{u^n_{i+\frac{1}{2},j+1} - 2u^n_{i+\frac{1}{2},j} + u^n_{i+\frac{1}{2},j-1}}{\Delta y^2}\\ \left(\frac{\partial^2 v^n}{\partial x^2} + \frac{\partial^2 v^n}{\partial y^2}\right)_{i,j+\frac{1}{2}} &= \frac{v^n_{i+1,j+\frac{1}{2}} - 2v^n_{i,j+\frac{1}{2}} + v^n_{i-1,j+\frac{1}{2}}}{\Delta x^2} + \frac{v^n_{i,j+\frac{3}{2}} - 2v^n_{i,j+\frac{1}{2}} + v^n_{i,j-\frac{1}{2}}}{\Delta y^2}\end{aligned} \tag{2.18}$$

1.3 Time step

In order to ensure the temporal stability of the explicit projection step, an adaptative time step Δt^n in respect of the convective, viscous, gravitational and capillary effects can be defined as in Kang et al [47]:

$$\begin{aligned}C_{conv} &= \max\left(\frac{\|u\|_\infty}{\Delta x}, \frac{\|v\|_\infty}{\Delta y}\right)\\ C_{visc} &= \max\left(\frac{\mu_l}{\rho_l}, \frac{\mu_g}{\rho_g}\right)\left(\frac{2}{\Delta x^2} + \frac{2}{\Delta y^2}\right)\\ C_{grav} &= \sqrt{\max\left(\text{abs}\left(\frac{g_x}{\Delta x}\right), \text{abs}\left(\frac{g_y}{\Delta y}\right)\right)}\\ C_{capl} &= \frac{\sigma\|\kappa\ \|_\infty}{\min\left(\rho_g, \rho_l\right)\min\left(\Delta x, \Delta y\right)^2}\end{aligned} \tag{2.19}$$



(a) $\bar{u} = u_{i+1/2,j-1/2}$ (b) $\bar{v} = v_{i-1/2,j+1/2}$

(c) $\bar{u} = u_{i+1/2,j}$ (d) $\bar{v} = v_{i,j+1/2}$

Figure 2.2: Interpolation of velocity for the staggered mesh and the advection equation.

that, combined, give

$$\Delta t^n \left(C_{conv} + C_{visc} + \sqrt{(C_{conv} + C_{visc})^2 + 4(C_{grav} + C_{capl})} \right) \leq 2 \text{ CFL} \tag{2.20}$$

where the parameter CFL is typically less or equal to 0.5, depending on the simulation parameters.

2 Jump conditions

Once the single phase Navier-Stokes has been set up, the next step is to realize the coupling between the two fluids. The jump conditions given by the physical model are here taken from chapter 1:

$$\begin{cases} [p] - \boldsymbol{n}.\left[\mu\left(\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^T\right)\right].\boldsymbol{n} = \sigma\kappa & (2.21) \\ \boldsymbol{t}.\left[\mu\left(\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^T\right)\right].\boldsymbol{n} = 0 & (2.22) \\ [\boldsymbol{u}] = 0 & (2.23) \end{cases}$$

The jump conditions are treated by the Ghost Fluid method, as detailed in chapter 3. The position of the interface is known after the advection of the Level Set function, and it is used to compute the jumps for the new velocity field $\boldsymbol{u}^{n+1}$. The objective is to realize a good coupling between the two fluids for the research of a final zero divergence velocity field.

2.1 Interface geometrical properties

The information given by the Level Set is the distance of the grid point to the interface in each cell. The imposition of the jump conditions require two more informations, which are relatively easy to derive from a Level Set formulation, as, differently from methods based on VOF, no reconstruction of the interface is needed. These are the normal (and consequently the tangential) vector and the curvature, that take part in the computation of pressure jump in the Laplace law. The normal vector and the curvature can be derived from the derivatives of ϕ:

$$\boldsymbol{n} = \frac{\nabla\phi}{\|\nabla\phi\|} \tag{2.24}$$

$$\kappa = \nabla\cdot\left(\frac{\nabla\phi}{\|\nabla\phi\|}\right) \tag{2.25}$$

Te divergence of gradient of the Level Set function is evaluated as $\nabla.(\nabla\phi) = 2\phi_x\phi_y\phi_{xy} - \phi_x^2\phi_{yy} - \phi_{xx}\phi_y^2$. The first and second derivatives can be disretized with a centered second order scheme as follows:

$$\begin{aligned} \phi_x &= \frac{\phi_{i+1,j} - \phi_{i-1,j}}{2\Delta x} \qquad \phi_{xx} = \frac{\phi_{i+1,j} + \phi_{i-1,j} - 2\phi_{i,j}}{\Delta x^2} \\ \phi_y &= \frac{\phi_{i,j+1} - \phi_{i,j-1}}{2\Delta y} \qquad \phi_{yy} = \frac{\phi_{i,j+1} + \phi_{i,j-1} - 2\phi_{i,j}}{\Delta y^2} \\ \phi_{xy} &= \frac{\phi_{i+1,j+1} + \phi_{i-1,j-1} - \phi_{i-1,j+1} - \phi_{i+1,j-1}}{4\Delta x\Delta y} \end{aligned} \tag{2.26}$$

while the denominator of the equations (2.24) and (2.25) can be expressed as

$$\|\nabla\phi\| = \left(\phi_x^2 + \phi_y^2\right)^{3/2} \tag{2.27}$$

If this quantity is zero, the derivatives are computed with a first order de-centered scheme.

2.2 Evaluation of jumps

The pressure jump

The Ghost Fluid allows an explicit treatment of jump conditions across an interface. In order to use this method the discontinuities of the fluid properties and unknowns as well as their first derivatives must be explicitly evaluated in each direction. Given a two dimensional case where the vector $\boldsymbol{u}$ contains the components (u, v) and the normal vector the components (n_x, n_y), the diffusion term of equation (2.21) can be written as

$$\boldsymbol{n}.\left(\nabla\boldsymbol{u} + (\nabla\boldsymbol{u})^T\right) = \begin{pmatrix}\nabla u.\boldsymbol{n}\\ \nabla v.\boldsymbol{n}\end{pmatrix} + \begin{pmatrix} n_y\frac{\partial v}{\partial x} - n_y\frac{\partial u}{\partial y} \\ n_x\frac{\partial u}{\partial y} - n_x\frac{\partial v}{\partial x}\end{pmatrix} \tag{2.28}$$

and then

$$\boldsymbol{n}.\left(\nabla\boldsymbol{u} + (\nabla\boldsymbol{u})^T\right).\boldsymbol{n} = \begin{pmatrix}\nabla u.\boldsymbol{n}\\ \nabla v.\boldsymbol{n}\end{pmatrix}.\boldsymbol{n} \tag{2.29}$$

Then the normal jump in equation (2.21) can be expressed as

$$[p] - \left[2\mu\begin{pmatrix}\nabla u.\boldsymbol{n}\\ \nabla v.\boldsymbol{n}\end{pmatrix}\right].\boldsymbol{n} = \sigma\kappa \tag{2.30}$$

Kang et al [47] develops the equation which, with the incompressibility of the fluids, becomes

$$\left[\begin{pmatrix}\nabla u.\boldsymbol{n}\\ \nabla v.\boldsymbol{n}\end{pmatrix}.\boldsymbol{n}\right] = 0 \tag{2.31}$$

The pressure jump can be written as

$$[p] - 2\,[\mu] \begin{pmatrix} \nabla u.\boldsymbol{n} \\ \nabla v.\boldsymbol{n} \end{pmatrix} .\boldsymbol{n} = \sigma\kappa \tag{2.32}$$

which is easy to implement on the code once the normal direction computed: the discontinuity is now relative to p and μ alone.

The viscous term jump

In the (2.22) the tangential equation allows the computation of the velocity tangential derivatives across the interface. The passage from a Cartesian base formulation to a local base one has been avoided by Kang et al [47], who give the following Cartesian based system of equations

$$\begin{aligned} \begin{pmatrix} [\mu u_x] & [\mu u_y] \\ [\mu v_x] & [\mu v_y] \end{pmatrix} &= [\mu] \begin{pmatrix} \nabla u \\ \nabla v \end{pmatrix} \begin{pmatrix} 0 \\ \boldsymbol{t} \end{pmatrix}^T \begin{pmatrix} 0 \\ \boldsymbol{t} \end{pmatrix} \\ &+ [\mu]\ \boldsymbol{n}^T\boldsymbol{n} \begin{pmatrix} \nabla u \\ \nabla v \end{pmatrix} \boldsymbol{n}^T\boldsymbol{n} \\ &- [\mu]\ \begin{pmatrix} 0 \\ \boldsymbol{t} \end{pmatrix}^T \begin{pmatrix} 0 \\ \boldsymbol{t} \end{pmatrix} \begin{pmatrix} \nabla u \\ \nabla v \end{pmatrix}^T \boldsymbol{n}^T\boldsymbol{n} \end{aligned} \tag{2.33}$$

This equation allows the computation of the jump on the velocity derivatives from the computations of the Ghost Fluid forcing terms. The Navier Stokes equations can now be solved in each fluid independently, as it is coupled to the other by the imposition of the jump conditions which act as boundary conditions for the interface border.

3 Boundary conditions

In this section the ensemble of the boundary conditions for the pressure and velocity used for the simulations are presented. They are necessary to close the equation system, and they must be carefully chosen in order to maintain the convergence rate of the solver. Once Dirichlet boundary conditions are imposed in the projected and corrected velocity $\boldsymbol{u}^*$ and $\boldsymbol{u}^{n+1}$, the pressure conditions are imposed by the correction equation (2.7) as homogeneous Neumann in the normal direction:

$$\boldsymbol{n}.\nabla p^{n+1} = \frac{\rho}{\Delta t^n}(\boldsymbol{n}.\boldsymbol{u}^* - \boldsymbol{n}.\boldsymbol{u}^{n+1}) = 0 \tag{2.34}$$

and the convergence rate of the solver is formally set. For the others conditions it is more difficult to establish, in particular for the inflow and outflow conditions.

In the solver guardcells are allocated to store the boundary values. The number of allocated cells depends upon the spatial extension of the numerical stencil. The number is imposed to 3 by the WENO spatial discretization scheme of the velocity. Only one cell is needed by the Poisson equation discretization. The image 2.3 shows the localization of boundary values on the right boundary. This is taken as an example to show the different kind of conditions; the other three boundaries are treated in the same way with the opportune index change.

3.1 Periodic conditions

These are the simplest boundary conditions, as they should represent an infinite domain which repeats indefinitely in space. They are obtained by replacing the boundary cells values with the

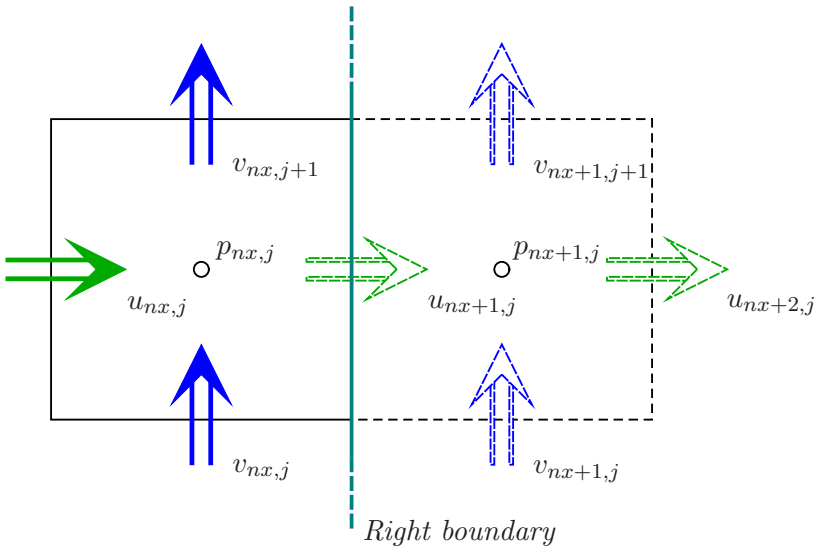


Figure 2.3: Variables and ghost values on the right boundary.

first values on the opposite face of the computed vector:

$$\begin{aligned} u_{nx+1,j} &= u_{1,j} & v_{nx+1,j} &= v_{1,j} \\ u_{nx+2,j} &= u_{2,j} & v_{nx+2,j} &= v_{2,j} & p_{nx+1,j} &= p_{1,j} \\ u_{nx+3,j} &= u_{3,j} & v_{nx+3,j} &= v_{3,j} \end{aligned} \tag{2.35}$$

These conditions are useful for convergence tests, because they are intrinsically "exact". They are also useful for temporal simulations where a fixed wavelength of a phenomenon is followed in time.

3.2 Slip conditions

Slip conditions are used when the physical boundary layer on a wall is negligible if compared to the boundary cell size. In this way the tangential velocity is not zero, but the Navier-Stokes solution value. The normal velocity is equal to zero; as the tangential velocity on the staggered mesh is not positioned on the real boundary but shifted of a half cell, symmetry conditions are used for this component:

$$\begin{aligned} u_{nx+1,j} &= 0 & v_{i,ny+1} &= v_{i,ny} \\ u_{nx+2,j} &= 0 & v_{i,ny+2} &= v_{i,ny-1} \\ u_{nx+3,j} &= 0 & v_{i,ny+3} &= v_{i,ny-2} \end{aligned} \tag{2.36}$$

The condition is applied to both $\boldsymbol{u}^*$ and $\boldsymbol{u}^{n+1}$, so the pressure has to respect the equation (2.34) and its boundary condition is an homogeneous Neumann.

3.3 No slip conditions

No slip conditions impose the respect of zero tangential velocity on the walls, as for the normal component. It is used when the physical boundary layer is important and has to be numerically taken in account. As for the previous case, the staggered mesh does not allow a direct imposition of tangential velocity on the physical boundary. The value in this location can be imposed by an interpolation of the velocity nodes on both sides. Moreover, the wall may move with its own speed. The usual method to adapt the no slip conditions to a wall moving at velocity V is to

mirror them across the boundary:

$$\begin{aligned} u_{nx+1,j} &= 0 \qquad v_{i,ny+1} = 2V - v_{i,ny} \\ u_{nx+2,j} &= 0 \qquad v_{i,ny+2} = 2V - v_{i,ny-1} \\ u_{nx+3,j} &= 0 \qquad v_{i,ny+3} = 2V - v_{i,ny-2} \end{aligned} \tag{2.37}$$

As in the slip conditions, it is adapted to both $\boldsymbol{u}^*$ and $\boldsymbol{u}^{n+1}$, and the pressure respects a zero gradient condition again.

3.4 Inflow conditions

An inflow condition imposes a mass flow entering the domain with a defined velocity profile $u_{inj}(y)$, perpendicular to the boundary. If the flow is turbulent, some models have to be used, but none of them is employed in this work. Theoretically the pressure should respect a non zero gradient depending on the diffusive term

$$\boldsymbol{n}.\nabla p^{n+1} = \mu \frac{\partial u_{inj}(y)}{\partial y} \tag{2.38}$$

If the same condition is applied to the projected velocity too, the pressure will follow a zero gradient condition. However, different conditions can be applied to the projected and corrected velocities in order to better define the pressure gradient

$$\frac{\rho}{\Delta t^n}(\boldsymbol{n}.\boldsymbol{u}^* - \boldsymbol{n}.\boldsymbol{u}^{n+1}) = \frac{\rho}{\Delta t^n}(\boldsymbol{n}.\boldsymbol{u}^* - u_{inj}) = \boldsymbol{n}.\nabla p^{n+1} \tag{2.39}$$

which gives the new boundary conditions

$$\begin{aligned} \boldsymbol{n}.\boldsymbol{u}^{n+1} &= u_{inj}(y) \\ \boldsymbol{n}.\boldsymbol{u}^* &= u_{inj}(y) + \frac{\Delta t^n}{\rho}\mu \frac{\partial u_{inj}(y)}{\partial y} \\ \boldsymbol{n}.\nabla p^{n+1} &= \mu \frac{\partial u_{inj}(y)}{\partial y} \end{aligned} \tag{2.40}$$

However, into the projection method the right hand side of the pressure equation make the term $\nabla.\boldsymbol{u}^*$ and $\boldsymbol{n}.\nabla p^{n+1}$ to elide themselves, so that the actual conditions revert to

$$\begin{aligned} \boldsymbol{n}.\boldsymbol{u}^{n+1} &= u_{inj}(y) \quad \boldsymbol{n}.\nabla.(\boldsymbol{t}.\boldsymbol{u}^{n+1}) = 0 \\ \boldsymbol{n}.\boldsymbol{u}^* &= u_{inj}(y) \quad \boldsymbol{n}.\nabla.(\boldsymbol{t}.\boldsymbol{u}^*) = 0 \\ \boldsymbol{n}.\nabla p^{n+1} &= 0 \end{aligned} \tag{2.41}$$

3.5 Outflow conditions

The outflow conditions should represent an "interruption" on the computational domain as it were a window for the observation of the flow. This is a numerical problem, because there are no informations available from the outside of the domain, so that no discretization stencil can be built, even at first order. In Couderc [18] two main axes of research for this problem are described. The first involves the resolution of a linear advection equation with the estimated advection speed on the boundary. Knowing the injected mass flow, this velocity can be evaluated to match the total mass flow. This method is reported, however, to generate a numerical aspiration towards the boundaries. Moreover, it is very difficult to adapt it to the two phase flow simulations. In the second approach, the mass conservation is imposed for $\boldsymbol{u}^{n+1}$ on all the last cells belonging

to the computational domain, so that all the mass entering the boundary cells is forced to flow outside. The ensemble of the conditions becomes then

$$\begin{aligned} \nabla.\boldsymbol{u}^{n+1} &= 0 \qquad & \boldsymbol{n}.\nabla.(\boldsymbol{t}.\boldsymbol{u}^{n+1}) &= 0 \\ \boldsymbol{n}.\boldsymbol{u}^{*} &= u_{inj}(y) \qquad & \boldsymbol{n}.\nabla.(\boldsymbol{t}.\boldsymbol{u}^{*}) &= 0 \\ p^{n+1} &= 0 \end{aligned} \tag{2.42}$$

The convergence of the so defined boundary conditions has been demonstrated by Guermond Guermond et al [35] for a rotational projection method; these condition are retained for the sheet simulations for the last chapter.

Conclusion

In this chapter the numerical discretization of the Navier-Stokes equations has been detailed. The pressure-velocity decoupling is performed by an explicit projection method. The discretization of the spatial derivatives of the momentum equation uses a fifth order WENO; the pressure equation is discretized by a second order centered scheme. The temporal derivative is discretized by the Adams-Bashford scheme. A global CFL condition is computed each time step by considering the different restrictions. The jumps of the properties on the interface are projected in the normal and tangential direction. Different boundary conditions are defined, including inflow and outflow conditions.

Chapter 3

The Interface Tracking

Contents

Introduction

The physical model described in the first chapter defines two distinct flows divided by an interface. This chapter describes the nature of the interface and its temporal evolution, which take place side by side with the evolution of the flows. At molecular length scale, an interface is a transition region where some molecules of the first fluid are mixed with molecules of the second fluid. Consequently, the thickness of this region is equal to few mean free molecular path. Depending on this dimension, the interface can be considered diffuse when the properties change gradually, or punctual, meaning a step change in the fluid properties. The first case can be described by the Cahn-Hilliard equation (Cahn and Hilliard [14]) as a transition zone. The other approach, used in this work, considers the interface as a discontinuity zone. The exact localization of the interface ensures the mass conservation in time when the fluids evolve and cause its topological deformation. Several numerical methods have been developed to localize and track this separation and to follow the topological changes, such as break-up or coalescence. Two main families can be distinguished.

The first contains the methods where the interface is defined on a second computational grid of which the nodes are evolved in a Lagrangian way. They are referred to as the *front tracking* methods, because they perform an explicit track of the interface by advection of the nodes on which it is defined. They allow a very accurate resolution of the properties jumps; they are, however, expensive and somehow complex to realize, especially when extended to three dimensions. To this class belongs the work of Daly [20], where a free surface flow is simulated by a Lagrangian advection of particles moving with an interpolated velocity coming from the fixed Navier-Stokes grid. Later works, such as Unverdi and Tryggvason [123], allow a full two phase flow simulation by limiting the markers to the interface itself instead of a whole phase (*Markers and Cells methods*). Its weaknesses are mainly the *reseeding* problem, arising when the markers are accumulated in certain regions due to the shape of the velocity field, and the necessity of a coalescence/break-up model. The method has been also coupled with the discontinuous Galerkin solver in Nguyen et al [74]. Some evolution of this philosophy involve some Euler-Lagrange hybrid treatment of the mesh (ALE, *arbtrary Lagrangian-Euler*), such as in Hirt et al [40], Duarte et al [27] and Ramaswamy and Kawahara [93], where the mesh evolves with a different velocity from the mean flow. The last option is to treat in a fully Lagrangian way the flow and the interface with the so called SPH (*smoothed particle hydrodynamics*), where the fluid model consists of distinct particles with a set of physical properties.

The second class includes the methods called *front capturing*, which use a fixed Eulerian Navier-Stokes grid with a scalar field defined on to implicitly represent the interface. The values relative to the interface such as the velocity are interpolated from the Eulerian field. The two main currents of this methodology are the Level Set and the Volume of Fluid (VOF). The first, developed mainly by Osher and Sethian [80], Sethian [102] and Sussman et al [113], consists in considering the zero contour of a signed distance function as interface. Its main advantages are the simplicity in its concept and numerical implementation, and the easiness in retrieving the topological properties as the normal vector and the curvature; its drawback is the lack of explicit conservation of mass as the simulation advances. The second one contains the methods based on the transport of a volume fraction (the *colour function*), varying from 0 to 1, which describes the percentage of presence of one fluid inside the cell. Conversely to the Level Set methods, the mass is exactly conserved in time, but the reconstruction of the interface is difficult. The VOF method can be found in the works of Hirt and Nichols [39], B. Lafaurie and Zanetti [5], and other variants, mainly in the interface reconstruction, in Noh and Woodward [75], Rider and Kothe [97], Gueyffier et al [36] and Scardovelli and Zaleski [100].

Some details on these methods are given in this chapter; the most part will be dedicated to the description and the numerical resolution of the solution which has been retained in this work, the Level Set.

1 Review of interface tracking methods

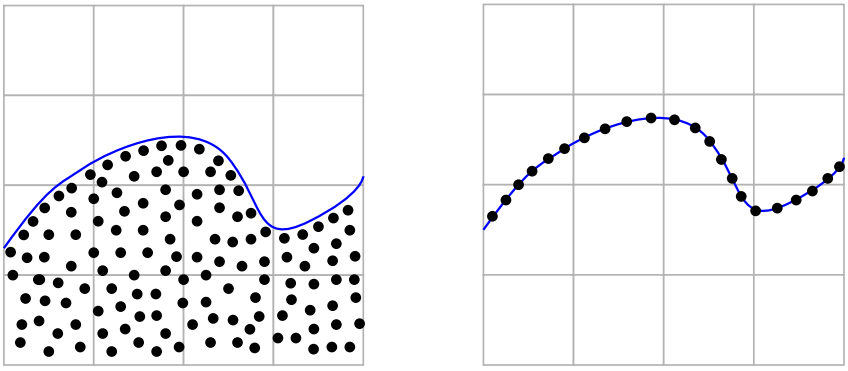
1.1 Mass trackers methods

In the precursor work of Harlow and Welch [38] one of the two phase is described by a set of particles without mass (*markers and cells*), as illustrated in figure 3.1a, superimposed to a fixed grid on which the velocity field is defined. The markers presence or absence define the interface; their position in time is solution of the linear equation

$$\frac{d\boldsymbol{x}_i}{dt} = \boldsymbol{u}_i \tag{3.1}$$

The method shows its full capabilities in Daly [20] where it is able to perform a simulation of a Rayleigh-Taylor instability. One of its remarkable advantages is the exact conservation of mass.

However several drawback hampers it. First of all, it is limited to the free surface flows, where only one fluid moves, the other being considered only to provide boundary conditions. The interface is not precisely defined, so that the computation of the topological characteristics as normal or curvature is complex. Moreover, the markers have to be always concentrated on the proximity of the interface when important deformations occur, in order to avoid an overload of computational resources, and the flow can induce excessive concentration or rarefaction of the markers.



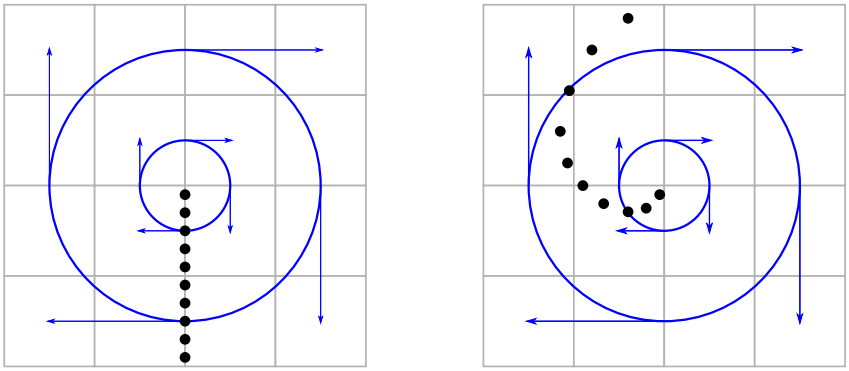
(a) (b)

Figure 3.1: Representation of different markers interface tracking methods. (a) Mass trackers (b) Interface trackers

1.2 Front-Tracking (explicit interface)

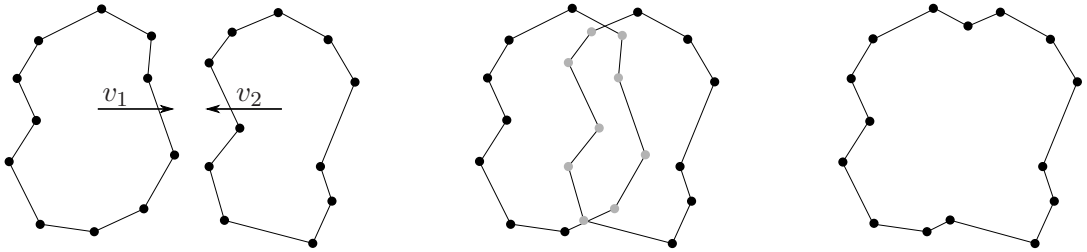
One simple alternative to the mass trackers is the utilization of markers to define the interface alone instead of one whole phase. The list of massless markers is again evolved by a Lagrangian advection, and the interface is each time step redefined by its new position, as shown in figure 3.1b. The Markers are connected by polynomial or *spline* interpolation of the desired order to get more or less accuracy (and oscillations). This method, introduced by Daly and Pracht [21], has the obvious advantage over the previous one of allowing a real two fluid simulations, and to have the interface always exactly defined by the interpolation of the markers. The Lagrangian advection of the zero mass particles is still subject to the effects of the underlying flow, and they tend to concentrate on "calm" flow regions and conversely to rarefy on strongly inertial flow ones, as can be seen represented in figure 3.2

An important issue concerning this method applied to the interfacial flow mechanics is the *coalescence* process, by which two or more droplets or particles merge during contact to form a single final structure. If two markers-traced interfaces representing the starting droplets (figure 3.3a) are brought from their mean velocity to overlap, if no action is taken some of the markers end up inside the interface of the merging droplets figure (3.3b). They have to be numerically removed because they do not represent an interface any more. The new resulting interface representing the final "big" drop keeps all the old markers outside the coalescence zone, and possibly adds markers on this latter to rebuild the merging point. Another criterion could be a minimum distance of the markers where the two old interfaces are broken and a new bond is created between the droplets. The opposite problem arises as well: the markers defined interface do not have any natural break-up criterion. As for the coalescence, a minimum distance of two markers inside a thin ligament can be used as the limit for the imposition of the division in two of the ligament, as can be represented in figure 3.4. The numerical algorithms which should deal with those problems are difficult and numerically "heavy", mainly when extended in three dimensions. Some advanced methods in which there is no explicit connection between markers



(a) (b)

Figure 3.2: Representation of the advection of massless markers by a whirling velocity field. The markers can be seen concentrating towards the center of the whirl, while becoming more rarefied towards the periphery. (a) Initial condition (b) After some time.



(a) (b) (c)

Figure 3.3: Coalescence of two structures with Lagrangian method. (a) Initial situation, two distinct droplets approach (b) Overlapping of the droplets, the grey points are inside the new interface. (c) Final interface: the internal markers have been removed and the interface reconstructed.

have been developed by Shin and Juric [103] by employing an "index function" of which the 0.5 contour should represent the interface. But apart from this enhancement, the markers based interface suffers from the same problems as the above mentioned methods.

1.3 Front-Capturing (implicit interface)

Volume of Fluid

This is the first method based on a discontinuous field Eulerian approach. It was originally applied to multiphase flow by DeBar [22]. In this method a scalar value representing the volumetric percentage of one fluid is given on each cell. This function, called *the color function* χ can vary from zero to one, being the two extremes the presence of only one of the two fluids in the cell:

$$\begin{cases} \chi(\mathbf{x},t) = 1 & \text{, if fluid 1} \\ \chi(\mathbf{x},t) = 0 & \text{, if fluid 2} \\ 0 < \chi(\mathbf{x},t) < 1 & \text{, } \chi \text{ fraction of fluid 1} \end{cases} \tag{3.2}$$

The method consists essentially in two steps. The first is the advection of the color function; the second is the reconstruction of the interface from the colour function. The colour function

(a) (b)

Figure 3.4: Artificial break-up criterion for Lagrangian advected ligament. (a) When two markers not consecutive are too near, a new interface is created between the two points (b) The left (circles) and right (triangles) define now two different items newly created.

is the solution of the linear advection equation

$$\frac{\partial \chi}{\partial t} + \boldsymbol{u}.\nabla\chi = 0 \tag{3.3}$$

$\boldsymbol{u}$ being the Eulerian velocity field. The interface is represented by the discontinuity of χ. The discrete correspondent function C_{ij} defined on the (i,j) cell of volume ν_{ij} is the integral value of the volumetric fraction of the one of the two fluids (exemple in figure 3.5a):

$$\chi_{ij}\nu_{ij} = \int_{\nu_{ij}} \chi(\mathbf{x})\ d\nu \tag{3.4}$$

The value C_{ij} is the solution of the conservation law in the sense of a finite volume discretization; the transport equation can be written in conservative and non conservative form given the $\nabla.\boldsymbol{u} = 0$ condition:

$$\frac{\partial C_{ij}}{\partial t} + \boldsymbol{u}.\nabla C_{ij} = 0 \text{ , non conservative form} \tag{3.5}$$

$$\frac{\partial C_{ij}}{\partial t} + \nabla.(\boldsymbol{u}C_{ij}) = 0 \text{ , conservative form} \tag{3.6}$$

The equations (3.5) and (3.6) are hyperbolic conservation laws, and their numerical solution is well documented in literature. The conservative resolution of such equation should guarantee the exact conservation of the mass. However, this equation is not usually directly solved because of the discontinuous nature of the discrete function C: the numerical diffusion would smooth the jumps as the simulation advances in time. Instead, the interface is reconstructed with the aim of building numerical fluxes from geometrical considerations, in order to ensure the local mass conservation. In this sense the differences amongst the VOF variants are mainly found into the interface reconstruction. The topological information are, on the other hand, needed to project the jump conditions in the local normal and tangential reference system. The quality of the overall method strongly depends on the accuracy of the reconstruction.

The simplest algorithm is the SLIC (*Simple Line Interface Calculation*), initially developed by Noh and Woodward [75]. It fills each cell with its value of C_{ij} by a constant value reconstruction, its position on the cell decided by the local velocity vector (figure 3.5b). It is at best a first order method. A straightforward enhancement of this algorithm is the PLIC (*Piecewise Linear Interface Calculation*, figure 3.5c), which tries to build a linear piecewise reconstruction always conserving the integral quantity of mass, thus obtaining a normal to the interface from the discontinuous C^1 reconstruction. The values of C_{ij} from the surrounding 3×3 stencil is used to compute the parameters of the reconstruction. To this category belong the works of Saltzman and Puckett [98] and Parker and Youngs [83], which use in different ways the stencil to obtain their geometry (the first computes the center of mass, the second the concentration

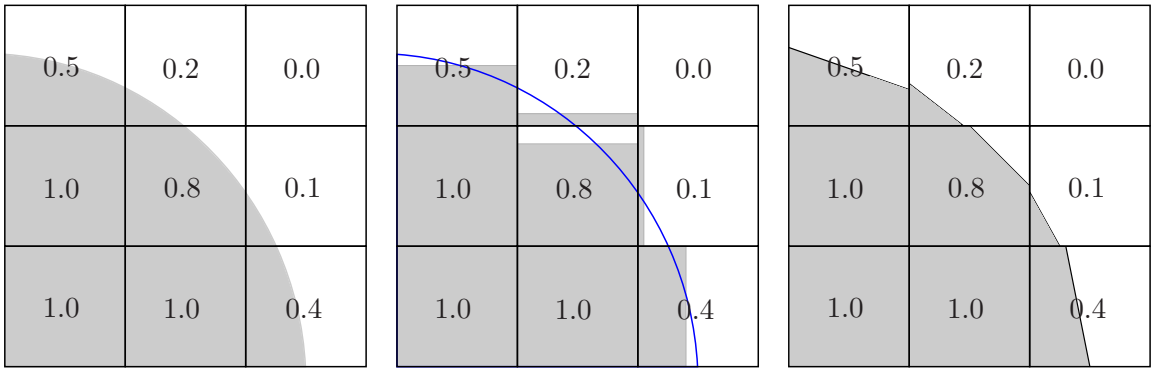


Figure 3.5: VOF reconstruction of the interface. (a) Interface position (b) SLIC constant piecewise reconstruction (c) PLIC linear piecewise reconstruction

gradient directly). A second order reconstruction has been developed by Puckett et al [92] and Pilliod [86], with respectively the LVIRA and ELVIRA methods, thanks to a minimization of a L_2 norm. Another approach has been presented by Aulisa et al [4]: in their work, several candidates linear (planar in three dimensions) interpolations are selected by a least-square fit procedure. This method is reported to be second order when applies iteratively. As summary, the VOF allows a natural treatment of the interface and its topological changes, breakup and coalescence. The numerical transport of the volume of fluid allows an exact conservation of the mass. On the other side, the position of the interface is not known explicitly, and its reconstruction may prove difficult to realize, mainly in three dimensions; the imposition of jump conditions when solving the Navier-Stokes equations may be difficult as a consequence.

Level-Set

The Level Set method was originally developed by Osher and Sethian [80] to model an interface moving with a velocity dependent on its curvature. Knowing the difficulty to compute the curvature in the VOF methods, they represented the interface as a contour of a smooth function (the ensemble of point (x, y) for which the function $\phi(x, y) = \text{const}$) where this and other topology informations would be easier. Usually this function is chosen to be hyper regular ($\|\nabla\phi\| = 1$), so that it is equivalent in its definition to a distance function. The Level Set function is passively advected by the local velocity field, it is therefore the solution of the transport equation

$$\frac{\partial\phi}{\partial t} + \boldsymbol{u}.\nabla\phi = 0 \tag{3.7}$$

This formulation for two phase flow problem is mathematically simple and elegant. It is however bound to the redistance problem. This mathematical problem consists in the loss of the distance properties as the initial function evolves and changes its shape. The redistance algorithm is necessary to reimpose this property at each time step. Additionally to the discretization error, this algorithm may generate mass losses by artificial shift of the interface and additional numerical diffusion, as well as the round-off of sharp corners. For this reason The Level Set equation is usually discretized with high order schemes. For the code described in this work the Level Set formulation has been chosen because of its simple definition and its advantages in the interface topology quantification. For this reason a more detailed overview of this method is given in the next section.

1.4 Hybrid methods

Levels Set / VOF

Given some opposite advantages and disadvantages of the described methods, some hybrid algorithms have been developed to obtain advantage of both the original ones. The first interesting case is the coupling between Level Set and VOF, a technique pioneered by Sussman and Puckett [112] to simulate the dynamics of an axisymmetric rising bubble, with a remarkable application in the liquid sheet primary atomization in Menard et al [67]. The method is called CLSVOF (*Coupled Level Set Volume Of Fluid*). Its main purpose is to use the Level Set function to precisely compute the interface characteristics and to correct the mass conservations errors with a VOF colour function. The flow-field is computed with a Level-Set method, which is advantageous for computing interface normals and curvature and regularization of discontinuities. In order to conserve mass, the interface is advected by a VOF (PLIC) method. After each interface update the coupling between the Level-Set and VOF colour function is done: some small local (meaning on each cell) Level Set shifts are performed to recover any artificial mass alteration. A drawback might be that the elaborateness of the VOF methods is imported. The advantage of the method is that the mass-conservation properties are shown to be comparable with "pure" VOF methods. VanDerPijl et al [125] proposed a variant where no VOF reconstruction is necessary, called MCLS (Mass Conserving Level Set). Differently from the CLSVOF, the only present VOF element is the volume fraction. Under the hypothesis of piecewise linear interface, a simple relation between this quantity and ϕ is set: a good mass conservation is achieved with a simpler method than the PLIC reconstruction.

Levels Set / Lagrangian

The results of the interface tracking given by the LS/VOF coupling are remarkable, but, being Eulerian, these methods still suffer from the smoothing of the sharp corners. In Enright et al [28] the Level Set interface is enhanced by a set of massless particles disposed in both sides, which are treated as seeding particles as in the Lagrangian methods. A particle crossing the interface is a detector of a local error, and consequentially the Level Set is corrected to match the information from the particles. Another possible application of the particles to the Level Set is the disposition of markers on the interface itself, as proposed in Mihalef et al [68]. Still in his work the accuracy of the Lagrangian method is not fully exploited, as there is no coupling with the momentum equation. Trontin [121] developed a particle-Level Set method fully coupled to the momentum equation, obtaining impressive results in the interface tracking.

1.5 SPH methods

Another alternative for the interfacial flow simulation is a mesh-less Lagrangian technique, where fluid interfaces are advected with very little numerical diffusion. It is called Smoothed Particle Hydrodynamics (SPH), and was originally developed to model astrophysical problems; however, it has since been extended to model a wide variety of problems in computational physics. The fully Lagrangian nature of SPH maintains sharp fluid interfaces without employing high-order advection schemes or explicit interface reconstruction, so it fits very well the two phase flow simulations. Several possible implementations of surface tension have been tested in Morris [73], giving some first successful simulations. Some work has still to be done concerning high density or high viscosity ratios. Finally, its high computational cost can hamper the diffusion of this algorithm for this kind of problems.

2 The Level Set method

2.1 Level Set definition

The Level Set is used to numerically reproduce an interface Γ of dimensions $n-1$ existing into a space Ω of dimension n, transported by the flow and undergoing consequent topological changes. Being $\boldsymbol{x}$ a point defined into the space Ω, the value of the Level Set function $\phi(\boldsymbol{x},t)$ is, at each instant of time t, the minimum distance of the point from the interface Γ, with a sign depending on which side it is contained. This function $\phi(\boldsymbol{x},t)$ can be seen as an infinite set of contour lines each of them at a fixed distance d from the interface, as shown in figure 3.6. When the distance is zero, the contour line represents the interface itself, $\Gamma = \{\boldsymbol{x} \mid \phi(\boldsymbol{x}) = 0\}$. This is one possible option for describing a Level Set function. It is however the simplest and has the advantage in the adaptive mesh refinement perspective to allow the mesh to "know" how far is the interface, thus making the refinement and de-refinement easier to manage. In order to define the properties on the two regions an arbitrary plus sign is assigned to one of the two sides, the minus being the other one. Any quantity α becomes

$$\alpha\left(\boldsymbol{x},t\right) = \begin{cases} \alpha^{(1)}\left(\boldsymbol{x},t\right) & \text{, if } \phi\left(\boldsymbol{x},t\right) \geq 0 \\ \alpha^{(2)}\left(\boldsymbol{x},t\right) & \text{, if } \phi\left(\boldsymbol{x},t\right) < 0 \end{cases} \tag{3.8}$$

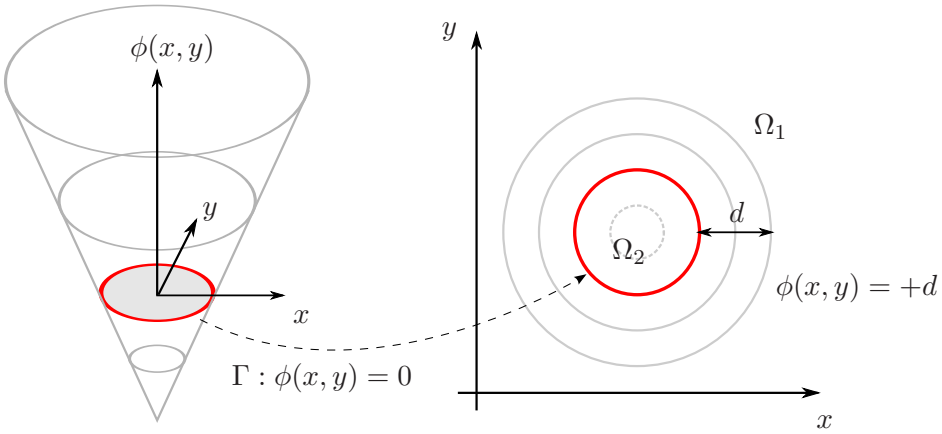


Figure 3.6: The Level Set "point of view": the contour lines of the function ϕ are the the locus of points in the plane distant ϕ from the interface, which is represented by the $\phi(x,y) = 0$ points.

The so formulated Level Set has some distinct advantages. The first is that the interface is exactly known at each time step, no reconstruction is needed. It has still to be numerically extracted the zero contour, but this is not necessary to get its geometrical properties. The normal vector and the curvature can be directly obtained from the values of ϕ:

$$\mathbf{n} = \left.\frac{\nabla\phi}{\|\nabla\phi\|}\right|_{\phi=0} \tag{3.9}$$

$$\kappa = \left.\nabla\cdot\left(\frac{\nabla\phi}{\|\nabla\phi\|}\right)\right|_{\phi=0} \tag{3.10}$$

The second is its hyper regularity property $\|\nabla\phi(\boldsymbol{x})\| = 1$. Being a signed function, no discontinuity appear on the interface, with the obvious advantages of an enhanced accuracy, stability and diffusion in the numerical schemes. This property can be locally lost when the function must describe sharp corners or filaments, especially in situations of under resolution. This aspects

shows how the Level Set function may naturally ask for a local refinement of the mesh. Another advantage is its capability to naturally handle the topological changes such as the coalescence and ligaments break-up, as shown as example in figure 3.7 without the need, differently for the front tracking methods, of heavy dedicated algorithms.

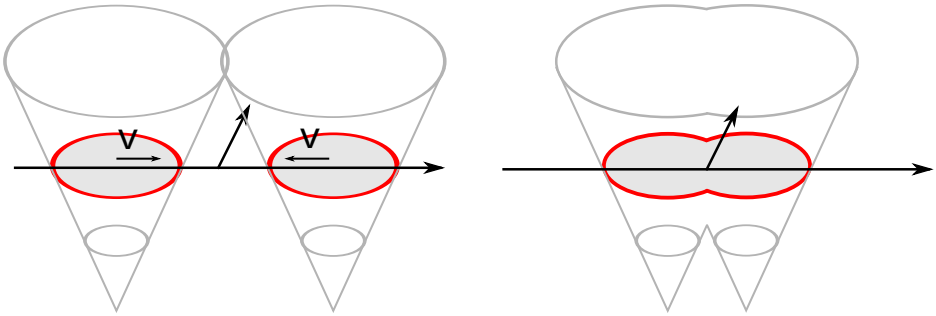


Figure 3.7: The Level Set simulated coalescence: two approaching droplets merge and create naturally a new structure (flow changes not taken in account).

2.2 Numerical resolution

The evolution of the Level Set follows the linear hyperbolic advection equation

$$\frac{\partial \phi}{\partial t} + \boldsymbol{u}.\nabla\phi = 0 \tag{3.11}$$

which means that it is passively advected by the velocity field $\boldsymbol{u}$. The numerical schemes should satisfy some requirements in order to correctly treat this problem. First of all the precision: the advection must be accurate enough to give a precise localization of the interface to allow a correct application of the jump conditions. In Tanguy [116] it has been demonstrated how the second order schemes are insufficient for the interface tracking problem. A higher order method should be used. The chosen scheme should also limit the numerical issues of diffusion and dispersion, in order to limit the loss of mass and the errors in the advection velocity for all the interfacial scales. Another fundamental property is the robustness. Although the Level Set is a regular function, some local under resolution situations may appear, especially in presence of strong deformations and gradients.

In Trontin [121] several high order advection schemes are combined with different interface tracking methods. In particular, the fifth order WENO of Shu [104] in conservative and non conservative form (Couderc [18]) and the spectral methods for both finite volumes (Wang [130]) and finite difference variants (Liu et al [54]) are compared. The retained scheme will be the WENO, which is reported to give good accuracy and to have optimized variants which help to improve the mass conservation. The results from Trontin [121] and Couderc [18] suggest a better behaviour in the conservative form of the WENO applied to the equation (3.11); however, when performing the high resolution sheet atomization simulation the scheme was reverted to the non conservative version for stability issues. For this reason both of them will be detailed in the following section. The temporal discretization must be realized by a high order scheme as well to couple a good temporal accuracy with the spatial one. The third order Runge-Kutta is used as for the previous cited algorithms.

2.3 Spatial discretization

In this section are presented the WENO (Weighted Essentially Non-Oscillatory) schemes based on the work in Osher and Shu [81], Shu [104], Shu and Jiang [106], which are an evolution of the

precedent ENO schemes of Shu and Osher [107]. They were initially developed for the simulation of discontinuities propagation (shock waves) in the compressible domain. The requirements were mainly the robustness and the ability to conserve the step with as limited as possible numerical smearing. These characteristics make them valuable to spatially discretize the equation (3.11). The scheme is suitable to solve both a conservative and a non conservative formulation of the advection equation, as described in Couderc [18] and Trontin [121].

Non conservative form

The equation in the "non conservative" form of the Level Set advection is written as

$$\frac{\partial \phi}{\partial t} + \boldsymbol{u}.\nabla\phi = 0 \tag{3.12}$$

Its discrete form in two dimensions looks like

$$\frac{\partial \phi_{ij}}{\partial t} + u_{ij}\left.\frac{\partial \phi}{\partial x}\right|_{ij} + v_{ij}\left.\frac{\partial \phi}{\partial y}\right|_{ij} = 0 \tag{3.13}$$

Leaving the temporal derivative for the next section, the interest is now in the spatial derivatives discretization. A centered discretization for this hyperbolic equation is unstable; the stability can be obtained (under opportune CFL condition $\Delta t < \ min(\frac{\Delta x}{|u|}, \frac{\Delta y}{|v|})$) with de-centered *upwind* schemes, which respect the direction of propagation of the characteristics by choosing the stencil according to the local direction of information propagation. With a compact notation of $\partial\phi/\partial x = \phi_x$, in the x direction this can be written as

$$\begin{cases} (\phi_x)_{ij} = (\phi_x)^+_{ij} = \dfrac{\phi_{i+1j} - \phi_{i,j}}{\Delta x} \text{ , if } \bar{u}_{i,j} < 0 \\ (\phi_x)_{ij} = (\phi_x)^-_{ij} = \dfrac{\phi_{ij} - \phi_{i-1,j}}{\Delta x} \text{ , if } \bar{u}_{i,j} > 0 \end{cases} \tag{3.14}$$

which correspond to a first order Euler upwind discretization, the same consideration being applied to the y direction too. The velocity $u^-_{i,j}$ is in this case the cell centered linear interpolation of the face centered values $u_{i,j}$ and $u_{i+1,j}$. The fifth order WENO scheme (WENO5) offers a better approximation of the derivatives $(\phi_x)^+_{ij}$, $(\phi_x)^-_{ij}$, $(\phi_y)^+_{ij}$, $(\phi_y)^-_{ij}$ by exploiting a total of five points organized into an upwind de-centered linear combination of three derivatives based on as many different stencils, as shown in figure 3.8.

$$(\phi_x)^\pm_{ij} = \sum_{k=1}^{3} \omega_k (\phi_x)^{\pm,k}_{ij} \tag{3.15}$$

where, for the values of $k = 1, 2, 3$, the derivatives are defined as

$$\begin{cases} (\phi_x)^{\pm,1}_{ij} = +\dfrac{1}{3}\, q^\pm_1 & -\dfrac{7}{6}\, q^\pm_2 & +\dfrac{11}{6}\, q^\pm_3 \\ (\phi_x)^{\pm,2}_{ij} = -\dfrac{1}{6}\, q^\pm_2 & +\dfrac{5}{6}\, q^\pm_3 & +\dfrac{1}{3}\, q^\pm_4 \\ (\phi_x)^{\pm,3}_{ij} = +\dfrac{1}{3}\, q^\pm_3 & +\dfrac{5}{6}\, q^\pm_4 & -\dfrac{1}{6}\, q^\pm_5 \end{cases} \tag{3.16}$$

with

$$\begin{cases} q^k_+ = \dfrac{\phi_{i-4+k,j} - \phi_{i-5+k,j}}{\Delta x} \\ q^k_- = \dfrac{\phi_{i+5+k,j} - \phi_{i+4+k,j}}{\Delta x} \end{cases} \tag{3.17}$$

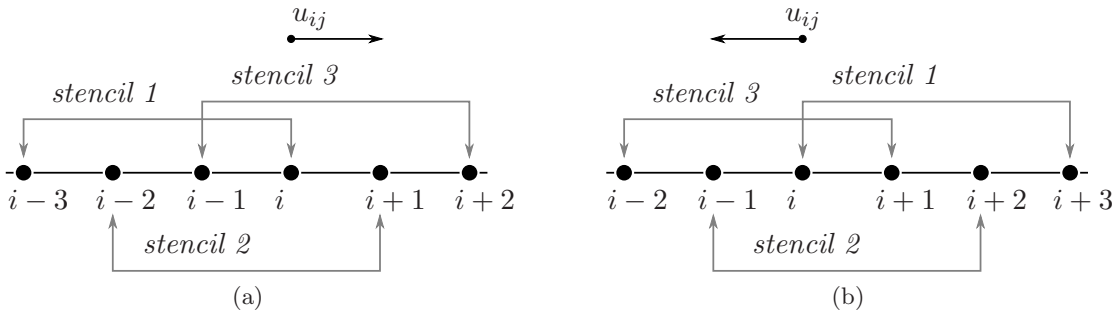


Figure 3.8: WENO 5 stencil in the non conservative form: (a) $(\phi_x)^-_{ij}$ (b) $(\phi_x)^+_{ij}$.

The coefficients ω_k are a convex combination ($\omega_1+\omega_2+\omega_3=1$) and are computed for $\omega^\pm$ by a the following expression of $\alpha^\pm$, :

$$\omega_k^\pm = \alpha_k^\pm / \sum_{i=1}^{3} \alpha_i^\pm \tag{3.18}$$

with the α defined as

$$\begin{cases} \alpha_1^\pm = \dfrac{1}{10}\left(\dfrac{1}{\epsilon + IS_1^\pm}\right)^2 \\ \alpha_2^\pm = \dfrac{6}{10}\left(\dfrac{1}{\epsilon + IS_2^\pm}\right)^2 \\ \alpha_3^\pm = \dfrac{3}{10}\left(\dfrac{1}{\epsilon + IS_3^\pm}\right)^2 \end{cases} \tag{3.19}$$

where ϵ ensures the denominator to be different from zero and the IS some Level Set "regularity" indicators:

$$\begin{cases} IS_1^\pm = \dfrac{13}{12}\left(q_1^\pm - 2q_2^\pm + q_3^\pm\right)^2 + \dfrac{1}{4}\left(q_1^\pm - 4q_2^\pm + 3q_3^\pm\right)^2 \\ IS_2^\pm = \dfrac{13}{12}\left(q_2^\pm - 2q_6^\pm + q_4^\pm\right)^2 + \dfrac{1}{4}\left(q_2^\pm - q_4^\pm\right)^2 \\ IS_3^\pm = \dfrac{13}{12}\left(q_3^\pm - 2q_4^\pm + q_5^\pm\right)^2 + \dfrac{1}{4}\left(3q_3^\pm - 4q_4^\pm + q_5^\pm\right)^2 \end{cases} \tag{3.20}$$

Another possible choice would be the so called "optimum" WENO, where the ω_k are fixed to the values $(\omega_1, \omega_2, \omega_3) = (0.1, 0.6, 0.3)$.

Conservative form

The equation (3.11) can be written in the "conservative" form without physically changing its meaning thanks to the incompressibility hypothesis, $\nabla.\boldsymbol{u} = 0$.

$$\frac{\partial \phi}{\partial t} + \nabla.(\boldsymbol{u}\phi) = 0 \tag{3.21}$$

The solution $\bar{\phi}$ is, in a finite volume sense, the integral value of the function on the cell of area Ω_{ij}. The the Gauss-Ostrogradsky theorem allows the transformation of the integral of the second term into a surface integral representing the net flux of the conserved quantity across the cell's faces:

$$\int_{\Omega_{ij}} \frac{\partial \phi}{\partial t}\, d\Omega + \int_{\Omega_{ij}} \nabla.(\boldsymbol{u}\phi)\, d(\partial\Omega) = \int_{\Omega_{ij}} \frac{\partial \phi}{\partial t}\, d\Omega + \int_{\partial\Omega_{ij}} (\boldsymbol{u}\phi).\boldsymbol{n}\, dl = 0 \tag{3.22}$$

the numerical fluxes on the faces of the cell $\Omega_{i,j}$, F located on the right cell face and G on the upper one are then defined as

$$\begin{cases} F_{i+\frac{1}{2},j} = \int_{y_{i,j-\frac{1}{2}}}^{y_{i,j+\frac{1}{2}}} u_{i+\frac{1}{2},j}\phi_{i+\frac{1}{2},j}\ dy \\ G_{i,j+\frac{1}{2}} = \int_{x_{i-\frac{1}{2},j}}^{x_{i+\frac{1}{2},j}} v_{i,j+\frac{1}{2}}\phi_{i,j+\frac{1}{2}}\ dx \end{cases} \tag{3.23}$$

The discretization of the equation (3.22) then becomes:

$$\left(\frac{\partial\phi}{\partial t}\right)_{i,j} + \frac{F_{i+\frac{1}{2},j} - F_{i-\frac{1}{2},j}}{\Delta x} + \frac{G_{i,j+\frac{1}{2}} - G_{i,j-\frac{1}{2}}}{\Delta y} \approx 0 \tag{3.24}$$

A numerical approximation for the fluxes (3.23) can be written, as for the derivatives in the non conservative form, using the upwind formulation:

$$F_{i+\frac{1}{2},j} \approx \begin{cases} u_{i+\frac{1}{2},j}\bar{\phi}^{+}_{i-\frac{1}{2},j} & \text{, if } u_{i+\frac{1}{2},j} > 0 \\ u_{i+\frac{1}{2},j}\bar{\phi}^{-}_{i+\frac{1}{2},j} & \text{, if } u_{i+\frac{1}{2},j} < 0 \end{cases} \tag{3.25}$$

and

$$G_{i,j+\frac{1}{2}} \approx \begin{cases} v_{i,j+\frac{1}{2}}\bar{\phi}^{+}_{i,j-\frac{1}{2}}, \text{ if } v_{i,j+\frac{1}{2}} > 0 \\ v_{i,j+\frac{1}{2}}\bar{\phi}^{-}_{i,j+\frac{1}{2}}, \text{ if } v_{i,j+\frac{1}{2}} < 0 \end{cases} \tag{3.26}$$

The value $\bar{\phi}_{i,j}$ can be chosen as the integral mean value ϕ_{ij}, which would give a first order discretization. A better approximated is obtained by using the same WENO5 discretization as used before: a smoothed ponderation of the quantities ${\phi_{i+\frac{1}{2},j}}^{+}$ and ${\phi_{i+\frac{1}{2},j}}^{-}$, as shown in figure 3.9.

$$\phi^{\pm}_{i+1/2,j} = \sum_{k=1}^{3} \omega_k \phi^{\pm,k}_{i+1/2,j} \tag{3.27}$$

with a similar formulation as in equation (3.16)

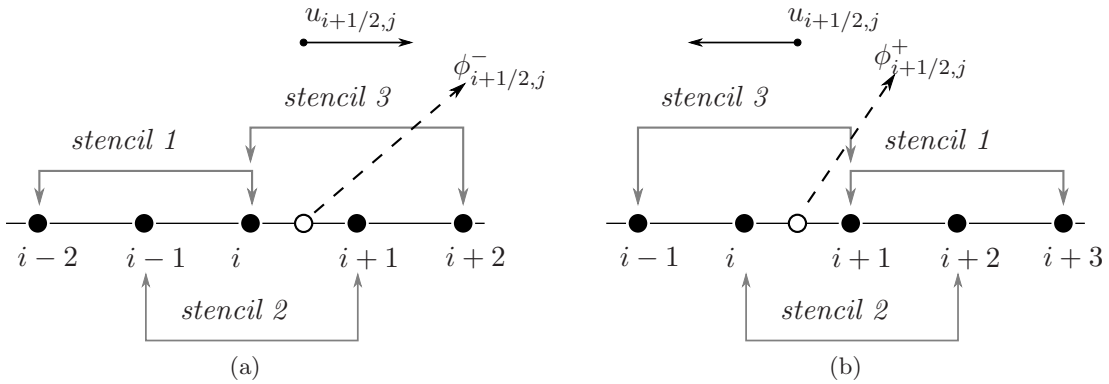


Figure 3.9: WENO 5 stencil in the conservative form. (a) $\phi^{-}_{i+1/2,j}$ (b) $\phi^{+}_{i+1/2,j}$.

$$\begin{cases} \phi^{\pm,1}_{i+1/2,j} = +\frac{1}{3}\ q^{\pm}_1 & -\frac{7}{6}\ q^{\pm}_2 & +\frac{11}{6}\ q^{\pm}_3 \\ \phi^{\pm,2}_{i+1/2,j} = -\frac{1}{6}\ q^{\pm}_2 & +\frac{5}{6}\ q^{\pm}_3 & +\frac{1}{3}\ q^{\pm}_4 \\ \phi^{\pm,3}_{i+1/2,j} = +\frac{1}{3}\ q^{\pm}_3 & +\frac{5}{6}\ q^{\pm}_4 & -\frac{1}{6}\ q^{\pm}_5 \end{cases} \tag{3.28}$$

and

$$\begin{cases} q_+^k = \bar{\phi}_{i-4+k,j} \\ q_-^k = \bar{\phi}_{i+5+k,j} \end{cases} \tag{3.29}$$

with the same definition of α and IS as in the equations (3.19) and (3.20).

2.4 Temporal discretization

An high order temporal resolution is necessary as well for the correct evolution of the numerical interface. The time derivative $\frac{\partial \phi}{\partial t}$ is discretized by a third order Runge-Kutta scheme from Gottlieb and Shu [34]. If the spatial discretization of the advection term is written as $-(\boldsymbol{u}.\nabla)\phi = L(\phi)$, the equation (3.11) can be written as

$$\frac{\partial \phi}{\partial t} = L(\phi) \tag{3.30}$$

and an explicit Euler step would look like

$$\phi^{n+1} = \phi^n + \Delta t L(\phi^n) \tag{3.31}$$

stable under the CFL condition $\Delta t_{Euler} < min\left(\frac{\Delta x}{|u|}, \frac{\Delta y}{|v|}\right)$. The Runge Kutta scheme can be seen as a combination of several Euler like "guesses" of the solution:

$$\begin{cases} \phi^{(i)} = \sum_{k=0}^{i-1} \left(\alpha_{ik}\ \phi^{(k)} + \Delta t\ \beta_{ik}\ L(\phi^{(k)})\right) \\ \phi^{(0)} = \phi^n, \qquad \phi^{(m)} = \phi^{n+1}, \qquad i = 1, ..., m \end{cases} \tag{3.32}$$

This scheme is stable under the following CFL condition:

$$\begin{aligned} &\Delta t < c\Delta t_{Euler} \\ &\text{with } c = min_{i,k}\frac{\alpha_{i,k}}{\beta_{i,k}} \\ &\text{and } \alpha_{i,k} \geq 0\ ,\ \beta_{i,k} \geq 0 \end{aligned} \tag{3.33}$$

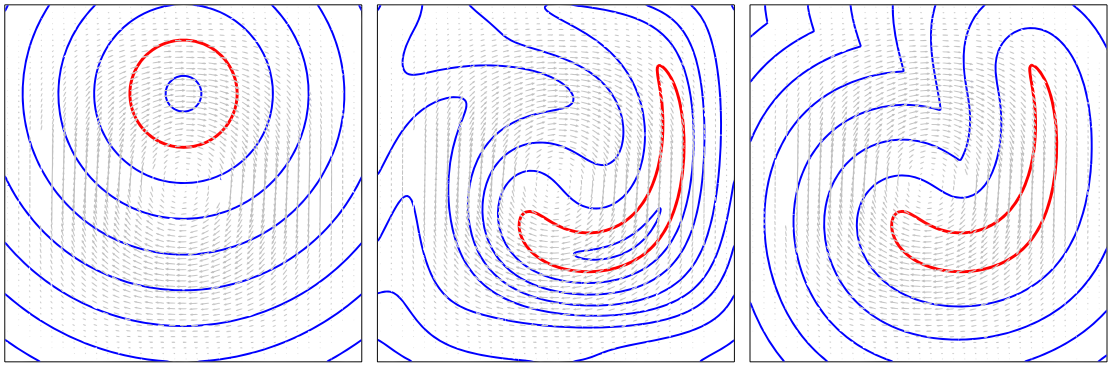
The coefficients of the third order RK can be found in Shu and Gottlieb [105], and give the final temporal discretization

$$\begin{cases} \phi^1 = \phi^n + \Delta t L(\phi^n) \\ \phi^2 = \frac{3}{4}\phi^n + \frac{1}{4}\left(\phi^1 + \Delta t L(\phi^1)\right) \\ \phi^{n+1} = \frac{1}{3}\phi^n + \frac{2}{3}\left(\phi^2 + \Delta t L(\phi^2)\right) \end{cases} \tag{3.34}$$

This allows a stability CFL condition of $c = 1$ in equation (3.33).

2.5 The redistance

During the advection of the Level Set, if the velocity field does not impose a rigid translation or rotation, the different contour lines are advected differently, as the local velocity gradient is not uniform: they are no more parallel and they no more represent a constant distance from the interface, $\|\nabla\phi\| \neq 1$ (figure 3.10a and 3.10b). If the position of the actual interface is not affected by this problem, the characteristic of distance is necessary to the computation of the normal vector and the jumps. The redistance algorithm has to modify the advected Level Set function, without altering the zero contour, in order to reimpose the $\|\nabla\phi\| = 1$, or the distance property.



(a) (b) (c)

Figure 3.10: Example of evolution of a Level Set function. (a) Initial condition (b) After some iterations, the contour lines are no more iso distance lines (c) With redistance, the contour lines are parallel again.

The algorithm

Sussman et al [113] developed an algorithm for the Level Set redistance problem. The main idea is that the zero contour line is correct, and the others should be corrected in function of it. The steady state of the following advection equation solved into a numerical time is the corrected Level Set:

$$\begin{cases} \dfrac{\partial \tilde{\phi}}{\partial t'} = sign\left(\phi(\boldsymbol{x},t)\right)\left(1 - \|\nabla\tilde{\phi}\|\right) \\ \tilde{\phi}(\boldsymbol{x},t'=0) = \phi(\boldsymbol{x},t) \end{cases} \tag{3.35}$$

the steady state solution is a distance function $\|\nabla\tilde{\phi}\| = 1$, and no modifications should be imposed to the interface. However, numerical discretization errors lead to small shifts of the interface, with the tendency to thicken the interface and smooth the sharp corners. The figure 3.10c shows the effect of the application of this algorithm to the situation in figure 3.10b. The equation (3.35) can be rewritten into a hyperbolic advection equation:

$$\frac{\partial \tilde{\phi}}{\partial t'} + \left(sign\left(\phi(\boldsymbol{x},t)\right)\frac{\nabla\phi}{\|\nabla\phi\|}\right).\nabla\phi = \text{sign}\left(\phi(\boldsymbol{x},t)\right) \tag{3.36}$$

This equation can be solved by the application of the methods used for the equation (3.11), thanks to the extension done in Jiang and Peng [46] of the WENO to the Hamilton-Jacobi equations: the equation (3.36) can be effectively written in this way, as described in Couderc [18].

2.6 Mass conservation

The Level Set method is affected by a mass loss effect, an aspect very disadvantageous if compared to the exact conservation of the VOF. If the use of high order schemes and the mesh refinement greatly improves this behaviour, a more definitive solution is still a current topic. Different strategies have been proposed for this problem, and it is a top priority in the continuation of this work to add one of these enhancement in the Level Set method. For example, Enright et al [28] proposes a Lagrangian tracking of the interface, with the obvious advantages in the accuracy of the advection; on the other side, the classical Lagrangian problems arise as well, and the algorithm is somewhat difficult to employ.

The transitory solution employed here is the global mass conservation. It is a correction which can be applied in a closed domain, where the physical phase quantity is not supposed

to change. It consists of an offset calculation for the Level Set function, so that an arbitrary contour can be shifted at the zero contour place. This allows the choice of an arbitrary quantity of mass (being always small changes required) contained inside the Level Set, quantity which can be chosen to be consistent with the previous instant of time. This exact conservation of the mass has the drawback of "teleporting" mass from the higher dissipation region towards the lower dissipation ones, as it adds mass uniformly on all the interface.

One of the most promising solutions, successfully employed by Sussman and Puckett [112] and Menard [66], is a coupled Level Set / VOF: in this method, the advected Level Set is locally corrected by the VOF to discretely conserve the mass. Again, the method allows an exact conservation of mass, but it suffers from some drawbacks: the Level Set function becomes less regular, and the VOF method can generate non physical droplets when facing under-resolution. It is however among the most interesting and complete methods, and it will be taken in serious consideration for future development of this code.

3 Imposition of the jump conditions

In the Level Set formulation of the two phase problem, the space is divided into two sub-domains which represent two distinct fluids. In each of them, the couple velocity-pressure are solution of the Navier-Stokes equations. The resolution of such equations needs boundary conditions to close the system. The jump conditions allow the connection of the two solutions: each sub-domain is solved in an independent way by receiving its boundary conditions from the other one. The system solved in a "single fluid" way is then correctly solved by the respecting of the boundary conditions on the domain limits and the jump conditions across the interface.

3.1 Methods overview

There are multiple possible approach to the numerical implementation of the jump conditions. Two main categories can be distinguished.

The first contains the methods which consider the effects of the interface as source terms in the right hand side of the equations. In this family is the so called "Continuum Surface Force" (CSF) introduced by Brackbill et al [12] and derived from the pioneer work of Peskin [84] on the "Immersed Boundary" method for biological flows. In these algorithms, as the interface does not coincide with the Eulerian mesh nodes, the interfacial surface force is transformed into a volume force in the region near the interface, using a delta function $f_v(x) \approx \int_\Gamma f_s(x)\delta^\epsilon(x - x_s)\, d\Gamma$ where δ^e is the Dirac function centered on the x_s point. The results of the application of the CSF and IB methods model the discontinuities as slightly smoothed variations on some cells around the interface.

The second category includes the immersed interface methods ("IIM") from LeVeque and Li [52, 53], where the discontinuity is taken explicitly in the equation discretization. This method has its basis into a Taylor's series development of the discrete solution, obtained taking in account the position of the interface and inserting the jumps inside the stencils. It is a second order method, but it is somewhat difficult to implement, especially for the three dimensional simulations. Another drawback is the consequent generation of a Poisson's equation matrix no more symmetrical, hurting the performances of the iterative solver. It happens, however, that the matrix resulting from the discretization on the adaptive mesh has the same consequence.

A simpler and symmetrized version of the IIM has been originally developed by Fedkiw for discontinuities like shocks, detonation and deflagration in Euler equations Fedkiw et al [30] and viscous flows Fedkiw and Xu [29], the "Ghost Fluid". This method can be employed to model the Navier-Stokes equations without adding any volume source term, and is easy to implement. Its technique is basically to extend each fluid characteristic to the opposite side of the interface

(generating the "ghost fluid" zone) so that standard stencils can be used, and to bring to the right side of the equation the contribution of the discontinuities. Better results in interfacial flow solutions are reported by Couderc [18] in comparison with the CSF solution. For this reason the GFM has been chosen on the bounce of his compared analysis.

3.2 The Ghost Fluid method

In this section a quick overview to the Ghost Fluid method applied to the pressure equation is given, returning to the original article of Kang et al [48] for the detailed two dimensional explication. Two complementary domains are given, Ω^+ and Ω^-. The problem to be solved is

$$\begin{cases} \dfrac{\partial}{\partial x}\left(\beta(x)\dfrac{\partial u(x)}{\partial x}\right) = f(x) \\ [u]_\Gamma = u^+ - u^- = a(x) \quad x \in \Gamma \\ \left[\beta(x)\dfrac{\partial u(x)}{\partial x}\right]_\Gamma = \left(\beta(x)\dfrac{\partial u(x)}{\partial x}\right)^+ - \left(\beta(x)\dfrac{\partial u(x)}{\partial x}\right)^- = b(x) \quad x \in \Gamma \end{cases} \tag{3.37}$$

This case is focused on C^0 and C^1 discontinuities, as described in equation (3.37), because both appear in the two phase solution, the C^0 in the pressure and the C^1 in the tangential velocity derivatives.

C^0 discontinuity

In this configuration in the system (3.37) there are $a(x) \neq 0$ and $b(x) = 0$. This is the case described in Kang et al [48], where a Poisson equation is solved in a complex domain described by the interface. The imposition of the jump conditions becomes the imposition of the right boundary conditions to the pressure equation. The standard discretization of this equation in the x_i (the left point, the right one x_{i+1} is treated symmetrically) cell is:

$$\frac{\beta_{i+\frac{1}{2}}\left(\dfrac{u_{i+1}-u_i}{\Delta x}\right) - \beta_{i-\frac{1}{2}}\left(\dfrac{u_i - u_{i-1}}{\Delta x}\right)}{\Delta x} = f_i \tag{3.38}$$

In order to evaluate the first and second derivatives, the jump values in C^0 and C^1 are introduced into the definitions of the derivatives themselves. These values allow the computation of a projected value of the quantity to the other side of the discontinuity, finally completing the discretization stencil. In practice the contribution of the interface is in the addition of a right hand side term that gives the discontinuous discrete solution. As it can be seen in figure 3.11, the interface located in x_I between x_i and x_{i+1} divides the two domains Ω^+ and Ω^-. The "ghost fluid" consists into the prolongation of the two domains on both sides of the interface, so the x_i^+ and x_{i+1}^- points. Their values are

$$\begin{cases} u_i^+ = u_i^- + a(x_I) \\ u_{i+1}^- = u_i^+ - a(x_I) \end{cases} \tag{3.39}$$

Then the equation (3.38) can be rewritten for the left side as

$$\frac{\beta_{i+\frac{1}{2}}\left(\dfrac{u_{i+1}^- - u_i^-}{\Delta x}\right) - \beta_{i-\frac{1}{2}}\left(\dfrac{u_i^- - u_{i-1}^-}{\Delta x}\right)}{\Delta x} = f_i \tag{3.40}$$

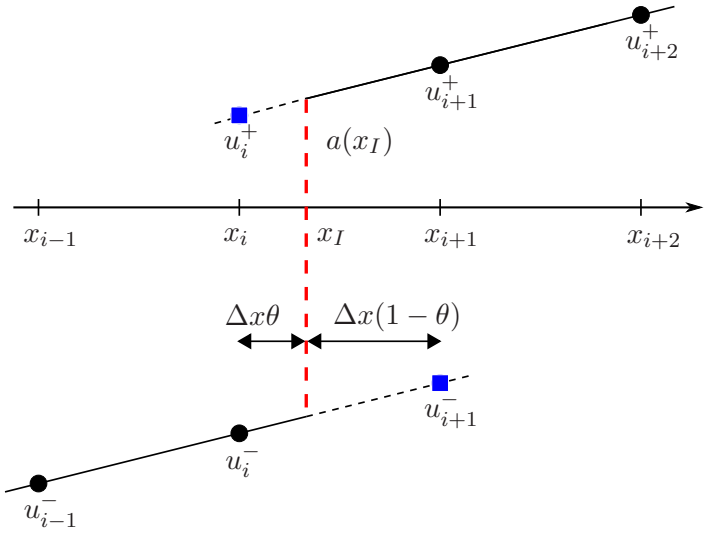


Figure 3.11: C^0 discontinuous solution u and creation of the ghost points.

By substituting (3.39)

$$\frac{\hat{\beta}\left(\frac{u_{i+1}-a(x_I)-u_i}{\Delta x}\right)-\beta_{i-\frac{1}{2}}\left(\frac{u_i-u_{i-1}}{\Delta x}\right)}{\Delta x}=f_i \tag{3.41}$$

and

$$\frac{\hat{\beta}\left(\frac{u_{i+1}-u_i}{\Delta x}\right)-\beta_{i-\frac{1}{2}}\left(\frac{u_i-u_{i-1}}{\Delta x}\right)}{\Delta x}=f_i+\frac{a(x_I)}{\Delta x^2}\hat{\beta} \tag{3.42}$$

where the value $\hat{\beta}$ is a weighted value of the two surrounding β on $x=x_I$

$$\hat{\beta}=\frac{\beta_{i+1}\beta_i}{\beta_{i+1}\theta+\beta_i(1-\theta)} \tag{3.43}$$

once obtained θ from the Level Set values

$$\theta=\frac{|\phi_i|}{|\phi_{i+1}|+|\phi_i|} \tag{3.44}$$

The equation (3.42) recalls exactly the original discretization of the Poisson equation (3.38) with an added forcing term and a modified coefficient β, revealing how the Ghost Fluid method allow the use of the single phase discretization for the two phase flows, without altering the symmetry of the discretization matrix (which will, however, be lost within the block based AMR). The value of the jump $a(x_I)$ can be obtained, as for the value $\hat{\beta}$, from a weighting of the values of the interface

$$a(x_I)=\frac{a_i|\phi_{i+1}|+a_{i+1}|\phi_i|}{|\phi_{i+1}|+|\phi_i|} \tag{3.45}$$

C^1 discontinuity

This time the solution u of the system (3.37) is continuous through the interface, but not its first derivative, as shown in figure 3.12. This concretely happens in the viscous terms $\frac{\mu}{\rho}(\nabla.\boldsymbol{u}+\nabla.\boldsymbol{u}^T)$.

The jump condition for the equation (3.37) can be written as

$$\beta_{i+1}\left(\frac{u_{i+1}-u_I}{(1-\theta)\Delta x}\right)-\beta_i\left(\frac{u_I-u_{i-1}}{\theta\Delta x}\right)=b(x_I) \tag{3.46}$$

and it can be solved for u_I

$$u_I=\frac{\beta_{i+1}u_{i+1}\theta+\beta_i u_i(1-\theta)-b(x_I)\theta(1-\theta)\Delta x}{\beta_{i+1}\theta+\beta_{i-\frac{1}{2}}(1-\theta)} \tag{3.47}$$

so that approximate derivatives can be written on both sides of the interface.

$$\begin{cases}\beta_{i+1}\left(\dfrac{u_{i+1}-u_I}{(1-\theta)\Delta x}\right)=\hat{\beta}\left(\dfrac{u_{i+1}-u_i}{\Delta x}\right)-\dfrac{\hat{\beta}b(x_i)\theta}{\beta_i}\\ \beta_i\left(\dfrac{u_I-u_i}{\theta\Delta x}\right)=\hat{\beta}\left(\dfrac{u_{i+1}-u_i}{\Delta x}\right)-\dfrac{\hat{\beta}b(x_i)(1-\theta)}{\beta_{i+1}}\end{cases} \tag{3.48}$$

As for the previous case, the discretization of the equation (3.37) for the point x_i can be reverted to the original form with an addition to the right hand side:

$$\frac{\hat{\beta}\left(\dfrac{u_{i+1}-u_i}{\Delta x}\right)-\beta_{i-\frac{1}{2}}\left(\dfrac{u_i-u_{i-1}}{\Delta x}\right)}{\Delta x}=f_i+\frac{\hat{\beta}b(x_i)(1-\theta)}{\beta_{i+1}} \tag{3.49}$$

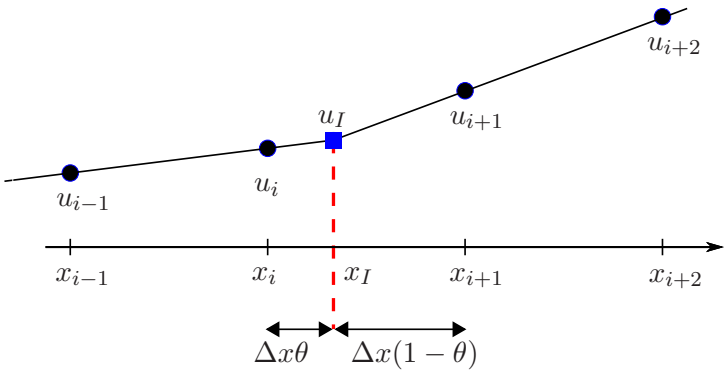


Figure 3.12: C^1 discontinuous solution u, identification of the u_I point.

The final form

The final form of the Poisson equation can finally be represented as

$$\frac{\hat{\beta}_{i+\frac{1}{2}}\left(\dfrac{u_{i+1}-u_i}{\Delta x}\right)-\hat{\beta}_{i-\frac{1}{2}}\left(\dfrac{u_i-u_{i-1}}{\Delta x}\right)}{\Delta x}=f_i+F_i^L+F_i^R \tag{3.50}$$

where the stencil is the same as before, but the coefficients β are recalculated using the weighted distance from the interface:

$$\begin{cases}\hat{\beta}_{i+\frac{1}{2}}=\dfrac{\beta_i\beta_{i+1}\left(|\phi_i|+|\phi_{i+1}|\right)}{\beta_i|\phi_{i+1}|+\beta_{i+1}|\phi_i|}\\ \hat{\beta}_{i-\frac{1}{2}}=\dfrac{\beta_i\beta_{i-1}\left(|\phi_i|+|\phi_{i-1}|\right)}{\beta_i|\phi_{i-1}|+\beta_{i-1}|\phi_i|}\end{cases} \tag{3.51}$$

Two additional terms appear in the right hand side: an F_i^L that contains the modifications imposed by the two types of discontinuities coming from the left side of the interface, and the corresponding F_i^R for the right one. If the interface is not present within the five points, those terms are equal to zero and the coefficients revert to the original ones. The same approach is used in the computation of the velocity derivatives in the viscous terms.

Conclusion

This chapter describes the interface numerical model. A quick review of the different techniques focuses mainly on the Eulerian methods, in particular on the VOF and Level Set. This latter has been used in this work, so it is further detailed, as it is the resolution of its associated transport equation. Spatial and temporal high order schemes are employed to provide accuracy and robustness, respectively the fifth order WENO and the third order Runge-Kutta. In the last section the imposition of interfacial jump conditions by the Ghost-Fluid method has been described as well. No discussion is provided about the final choice of the method, as it is beyond the scope of this work.

Chapter 4

Adaptive Mesh Refinement

Contents

Introduction

Many modern numerical simulations require enormous computational resources, both in computing time and storage. This is the trade-off for the achievement of high precision in the computed solution, as the cell size decreases to better evaluate the numerical derivatives. The cell size also affects the maximum time-step, and with it the number of required steps to reach the desired simulation time, a problem particularly evident in explicit schemes. On the other hand, if the reduced cell size means more points on the mesh, often the different regions of the computational domain are not equally important in term of required precision or difficulty, thus wasting resources where they are not needed. The adaptive mesh refinement (AMR) is a technique meant to make the mesh to adapt, in space and time, to the computation, allowing more and less refined region to coexist and change during the computation.

1 Meshes

Generally speaking, a mesh is a collection of points (or nodes) and some notion of connectivity among them. Meshes come in several types, each with its own means of specification. The most general type of mesh is the unstructured mesh, which is a collection of nodes and a description of the arbitrary connectivity between them. In this case, adding new points requires only the building of the connections between them and the old ones.

A structured mesh, unlike the previous one, has its connectivity implicit in the specification of the mesh; that is, the order in which the nodes are specified determines their connectivity. In other words, each point has the same relationship with his neighbours as any other. In general, structured meshes are arranged in the same manner as arrays in computer memory. They are more difficult to refine, due to those problems:

- ▷ cells can be added to an existing mesh, in which case the implicit connectivity is lost;
- ▷ nodes can be shifted about the physical domain, but their number can't grow or decrease;
- ▷ multiple structured sub-meshes can be added, and their interactions controlled, requiring a lot of additional code.

The general structured meshes are conformal meshes, which are meshes with arbitrary geometry. Less general are regular meshes, with mesh lines parallel to the coordinate axes; uniform meshes have also the cell spacing constant everywhere.

2 About the AMR techniques

There are different reasons for avoiding an uniformly distributed mesh:

- ▷ some regions have small gradients, so the solution is "smooth" and does not vary so much from a cell to its neighbours;
- ▷ the difference between the coarse and the fine mesh is negligible;
- ▷ there is an important physical phenomenon localized in a small region (a shock, an interface, a front);
- ▷ various phenomena of interest may occur at widely varying scales of space and/or time (for example in the astrophysical computation).

The use of the AMR can be seen in different philosophies, depending on the purpose of the simulation and the available resources. For example, one may want to perform his computation with a fixed precision, so that the idea of the adaptive mesh would actually consist in the de-refinement of the less interesting zones; on the other hand, one may need to improve the resolution on a precise spot, but would not want to pay for a global refinement. Another important factor for defining the purpose of AMR is the availability of parallel computation techniques. According to this, the main aspect of computational cost lessening can be different:

- ▷ increased time saving over a fixed grid approach, mainly in serial computations;
- ▷ increased storage saving over a fixed grid approach, mainly for massively parallel computations.

The two are both satisfied when a multi core computation is possible, but the resources are somewhat limited in relation to the desired mesh dimension. Several different application of an evolving mesh exist; each of them can be the best strategy for a particular numerical problem. The most general kind of AMR is the *h-refinement*, where points are added locally where needed. In the *r-refinement* the nodes are only relocated, not added or deleted. The last kind of AMR (*p-method*) does not work by modifying the mesh but instead the order of the numerical scheme. Nothing prevents the three types from being combined into a "total-AMR", hybrids are nevertheless rare because of the implementation complexity.

2.1 AMR for unstructured grids

Unstructured mesh methods for fluid dynamics have been developed in order to handle complex geometries, as demonstrated through finite-element-based approaches mainly in the field of structural analysis. Their use began to spread mainly after the recasting into the finite volume framework, for which approximate Riemann solver technology could be applied. In addition to the flexibility for dealing with complex geometries, the ability to easily incorporate AMR strategies was another quoted advantage of this approach. However, the practical automatic construction of mesh, the difficulty in dynamically load balancing such operations on massively parallel architectures and the lack of reliable error estimation techniques for the refinement criteria have somewhat limited the development of this approach. In this context, the AMR is realized by adding new points or deleting old ones, and then by rebuilding the connectivity between them. As the present work pursuits the development of a Cartesian grid based code, the interest in these meshing strategies is quite limited, so the attention will be more focused on the structured mesh strategy.

2.2 Multigrid

The multigrid methods are fast linear iterative solvers based on the multilevel or multi-scale paradigm. They are not AMR techniques in the common sense: they do not apply the idea of more or less refined regions. Instead, these methods can solve complex simulations by rapidly reducing the error inherent in many single grid iterative methods. They employ a hierarchy of grids of varying resolution, each covering the entire computational domain, as in figure 4.1. The underlying premise is that, although iterations on a fine mesh quickly eliminate the high frequency components of error, the low frequency ones take much longer, resulting in an inefficient or inaccurate solution. However, by iterating on meshes of various scales, the smoother error components can be reduced quickly as well. This method can be used at the same time of AMR, eventually sharing the same data structure. Multigrid is often associated to the resolution of elliptic problem as the Poisson equation, such as in Couderc [18], Martin [61], Sussman [111], Sussman et al [114], Brown and Lowe [13] in the multi step or projection methods, or

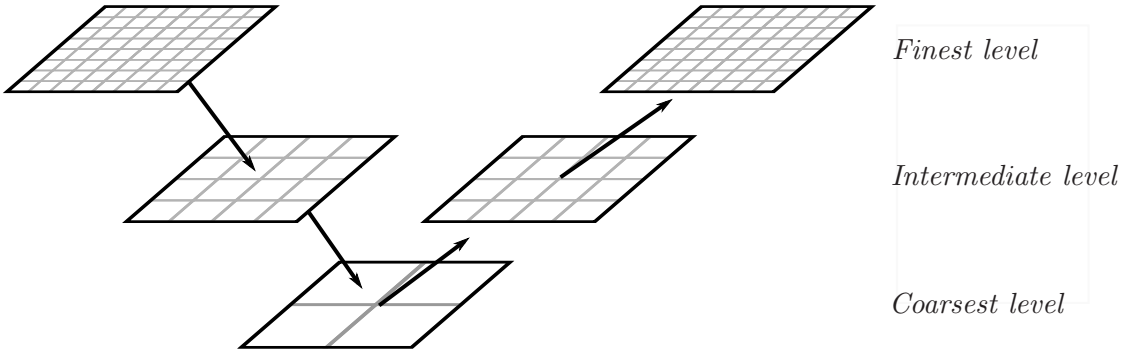


Figure 4.1: Representation of the multigrid principle.

gravitational potential in astrophysical simulations as in Ricker [96]. When the adaptation of mesh is employed additional problems arise, in particular the treatment of refinement jumps.

2.3 Moving mesh methods

These are a class of AMR that refine by redistributing mesh points, rather than by creating new ones, as in figure 4.2 (called sometimes *r-refinement*). They have been developed for solving particular class of problems, such as non linear hyperbolic and parabolic problems that develop shocks or other sharp moving fronts: node are moved in order to be concentrated around them. The problems are that the resulting mesh become non-uniform, so that uniform grid solvers can no longer be used. Moreover, due to the number of points not growing, a given mesh size may prove insufficient for some evolving situations: regions which require very high resolutions may "steal" nodes from other regions, which could be then insufficiently resolved. A more unusual

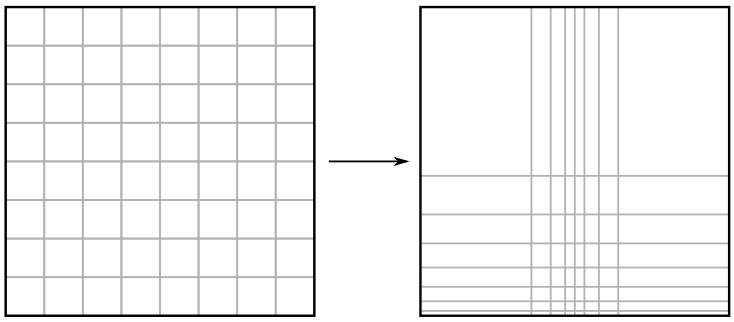


Figure 4.2: Example of evolution of a moving mesh (from left to right, before and after refinement).

kind of mesh with moving nodes is the ALE (*Arbitrary Lagrangian Eulerian*), introduced in Hirt et al [40]. The general idea is to be able to solve a wide class of problems at different flow speed and complex geometries. An Eulerian velocity field is defined on a finite difference mesh with vertices which may move with the fluid (Lagrangian), be held fixed or moved in any arbitrary way.

2.4 Tree-based AMR

The most widely spread kind of mesh adaptation is the *hierarchical* form, also referred to as *tree-based*. The tree-based AMR are among the most widely used kind of AMR. Their base principle is that the computational domain can be seen as a sum of several increasingly finer

levels, each of them covering only a fraction of it, with some kind connectivity kept between levels. This data structure can be presented schematically as a "tree" in which the "trunk" is the coarsest level, the "branches" the intermediate levels and the "leaves" the finest level. The Berger's refinement strategy and the quad-tree methods are two main examples of this philosophy. This AMR strategy will be detailed further in this chapter.

3 Hierarchical AMR

The hierarchical AMR comes from the idea of Marsha Berger [9]. This point of view describes the adaptation of the mesh as a set of separated mesh of different size with a "vertical" relationship between them; a coarser refinement level is never lost in the refinement process, but it continue to exist to support the finest levels, and they keep their relationship as long as they are not deleted. The way the finer grids are created from the coarser ones and the hierarchy is managed is the main difference among the different methods described in the literature.

3.1 Berger's AMR

The original Berger's version of structured mesh hierarchical AMR is based on a sequence of nested grids with successively finer spacing in both time and space. Increasingly finer grids are recursively embedded in coarse grids until the solution is sufficiently resolved. An error estimation procedure based on user specified criteria evaluates where additional refinement is needed and grid generation procedures dynamically create or remove rectangular fine grid patches as resolution requirements change. This AMR formulation is very oriented towards a level-by-level resolution, which naturally leads to a temporal sub-cycling on the refinement levels, each of them being tied to its own cell size depending CFL condition.

Incompressible Navier-Stokes with AMR

This kind of mesh refinement has been investigated a lot for the incompressible Navier-Stokes equations, with some serious issues bound to the divergence-free condition and the temporal synchronization of the grids. Howell and Bell [43] presented a two dimensional non conservative adaptive algorithm based on the Bell's projection formulation (Bell et al [8]), which did not sub-cycle in time. An error estimation procedure automatically determines where the solution resolution is inadequate, and grid generation procedures dynamically create rectangular properly nested finer patches in these regions. The solver is a projection method where the convection-diffusion equations is solved without strictly enforcing the incompressibility constraint, then the intermediate vector field is projected into the space of discretely divergence-free vector fields, all steps performed on the composite grid hierarchy. The use of multigrid iterations has in this work been applied on the individual grids that are distinct from the overall multilevel iteration. Coarser levels of the individual multigrids do not communicate with each other directly, nor do they communicate directly with coarser levels of the adaptive hierarchy. In the convection scheme and the elliptic solver the prolongation of fine grids boundary cells is a third order bi-quadratic interpolation; in the single grid multigrid iteration the restriction operation is a simple average of the four fine points to their associated coarse point, the interpolation is a decoupled piecewise-constant formula where the value from the coarse point is distributed unchanged to each of the associated fine points (direct injection).

Almgren et al [3] introduced the sub-cycling in time to solve the variable density, constant viscosity, incompressible flows based on a second order approximate projection method with a node centered variables location. Once data is initialized on all the AMR hierarchy, each refinement level is solved independently at its own time step, requiring no communication between

levels other than the supplying of Dirichlet data from a coarse level to be used as boundary conditions at the finer level, then to synchronize data at different levels at some specified interval. In this work, special attention is paid to the synchronization step of the algorithm, so that the overall method is conservative for density and free-stream preserving, in the sense that constant scalar fields with no source terms remain constant independently of grid refinement and velocity field. Three types of mismatch are corrected in this algorithm [1]:

- ▷ the data at a coarser level and at the overlapping finer one(s) are not synchronized;
- ▷ the composite advection velocity computed from the MAC projection does not satisfy the composite divergence constraint at the coarse-fine interfaces. This mismatch results in the free-stream loss, that is a spatially constant advected quantities with no source terms not remaining constant;
- ▷ the advective and diffusive fluxes on the refinement jump faces do not agree, resulting in a loss of conservation.

The proposed correction for the mismatches are, in the order:

- ▷ restriction is applied from fine to coarse grid to recover synchronization;
- ▷ coming this mismatch from a lacking elliptic matching condition from the projection, an additional elliptic equation with the mismatch as right hand side gives the correction;
- ▷ advective and diffusive fluxes form part of the right hand sides for the parabolic solves associated with the implicit Crank-Nicholson scheme, so the flux mismatches are the right hand side for the refluxing solves.

The elliptic equation is solved with a reconstruction of intermediate auxiliary levels (the refinement ration is $r = 4$, so there is room for intermediate $r = 2$ levels) to perform the global multigrid iteration. The interpolation and restriction are quite complicate finite-element-like formulation which focus on conservation of interpolated quantities.

The multilevel approximate projection solver for incompressible flows has been subsequently improved in Martin and Colella [63] (inviscid case) and Martin et al [64] (viscous flows). Several improvements are reported. The cell centered approximate projection discretization are employed; since cell-centered solvers are already required for other parts of the algorithm, the use of a single set of elliptic solvers substantially lessen the work required to extend this algorithm to embedded boundary computations and other applications. The flux-correction step, performed to ensure conservation at coarsefine interfaces, is computed in an implicit manner, done as multilevel elliptic solve over all of the appropriate refinement levels. In contrast, the work in Almgren et al [3] performs synchronizations one pair of levels at a time using single-level elliptic solves, interpolating corrections to finer levels. The free-stream preservation is obtained through the introduction of an auxiliary advected scalar field Λ whose purpose is to provide a measure of the extent to which free-stream preservation has been violated:

$$\begin{cases} \dfrac{\partial \Lambda}{\partial t} + \nabla.(\boldsymbol{u}\Lambda) = 0 \\ \quad \Lambda(\boldsymbol{x}, t = 0) = 1 \end{cases} \tag{4.1}$$

Where Λ is different from its initial value, a correction velocity is computed by an elliptic equation from the Λ mismatch. In general, the Martin's enhancements avoid the interpolation of

[1] Actually in the paper there are four of them, but two are the same type of incoherence applied to two different set of velocity vectors.

corrections computed by elliptic and parabolic solves onto finer levels. While such interpolation can easily be done to second-order accuracy, the gradients of such interpolated corrections can be non-smooth. Instead, the multilevel solves compute corrections over entire multilevel hierarchies, which results in smooth corrections with smooth gradients.

Multiphase flows with AMR

In multiphase flows simulation, the Level Set method lends itself well to being computed on an adaptive grid. Since most of the vorticity in these flows is concentrated on or near the interface, a local refinement should yield not only finer resolution of the interface itself but also reduced numerical diffusion of the velocity field. The refinement criterion can easily be defined in terms of the level set function, which is also the distance from the interface. In many works of Sussman [111], Sussman et al [114] coupled Level Set and VOF techniques are used with adaptive projection methods, developing the single grid algorithms described in M. Sussman [57], Sussman et al [113], Sussman and Puckett [112], Sussman et al [115]. They managed to extend the base adaptive algorithm in Almgren et al [3] to incompressible two-phase flows in which the jumps in density and viscosity between fluids are sharp and can be large. Some important improvement oriented toward the Level Set formulations are:

- ▷ the curvature of the interface along with vorticity are used as criteria for determining grid placement at the finest level, while the location of the interface determines grid locations at coarser levels;
- ▷ a redistance synchronization step is applied to correct errors resulting from doing the redistance operation separately on coarse and fine levels: this allows for robust computation even if part of the interface crosses a refinement jump;
- ▷ other synchronization steps related to the level set function are also necessary for maintaining the regularity of not only the level set function but also the regularity of the curvature across refinement jumps;
- ▷ because of the large density ratios, the multigrid solver is replaced by a multigrid preconditioned conjugate gradient solver (the viscous solve is still done with multigrid alone).

In Howell [42] and Martin and Colella [63], the semi-implicit algorithm is solved by a multilevel solver which behaves like a multigrid applied to the hierarchy of meshes. Sussman [111] develops a method that is a combination of the multigrid method and the preconditioned conjugate gradient (PCG) method, this latter being more robust than multigrid methods alone. A preconditioned PCG acts as level smoother (that is, on each grid level a "local" multigrid is performed to smooth the solution, as represented in figure 4.3), finding great success despite the PCG not being known for its smoothing properties. With ten smoothing sweeps per level, its solver reports convergence in about five V-cycles for a wide variety of problems and meshes. With a parallelization given by the *BOXLIB* (Rendleman and Lijewski [95]), a parallel speed-up of around 80% and an adaptive speed up of up to 75% are reported.

3.2 Quad-tree AMR

The quad-tree refers in general to a hierarchical data structure in which each element has exactly four children (so called in 2D because four grids are spawned from one; oct-tree in 3D because eight grids are generated in this case), as shown in figure 4.4. In the adaptive mesh context, it is used to partition a mesh by subdividing each cell or grid into smaller ones, obtained by bisection in each direction, where required. The base object can be a single cell that, when

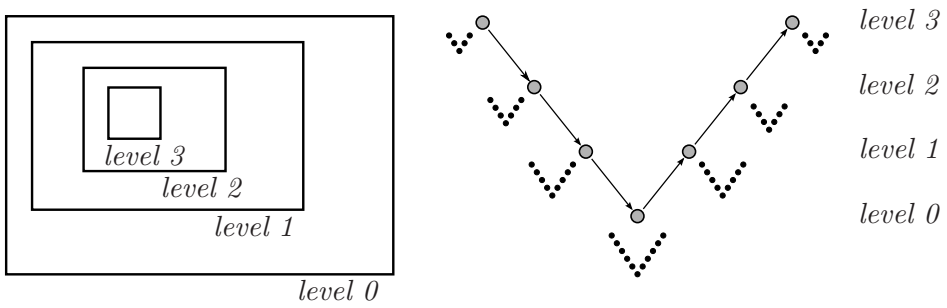


Figure 4.3: Representation of Sussman's multigrid preconditioner for the conjugate gradient: at each level (the bigger dots) smoothing is performed by a multigrid preconditioned PCG (the smaller dots).

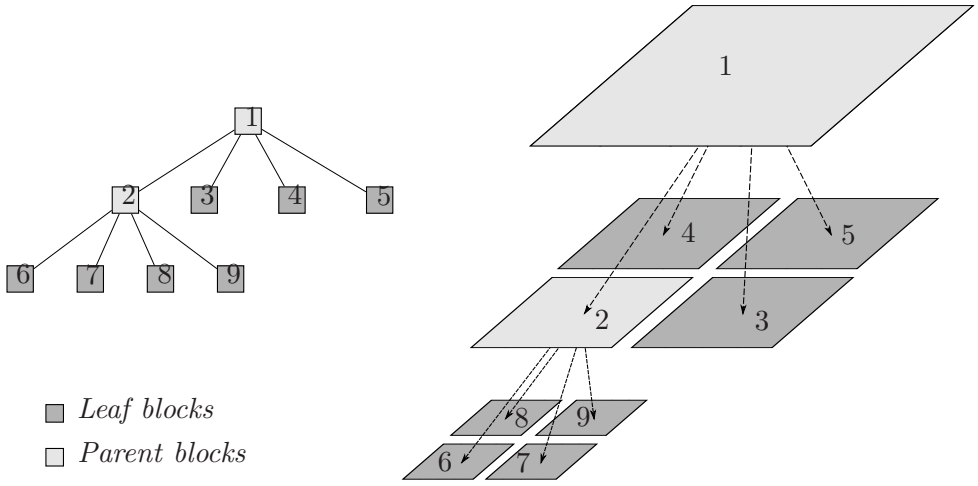


Figure 4.4: Representation of a quad-tree decomposition of a two dimensional domain.

refined, spawns four children cells of half the parent size or a whole grid, which generates four similar grids with smaller cells when refined.

The quad-tree principle has been at first introduced by DeZeeuw and Powell [25] to build grids adapted to complex boundaries such as wing profiles. The first application was for the solution of the steady Euler equations. This algorithm creates an initial uniform mesh, and then refine each cell by a bisection in each direction, following the body curvature; the cells lying on the profile itself are individually cut to adapt to the shape. Next, the solution is converged to a steady state using a linear reconstruction and Roe's approximate Riemann solver. The idea of the recursive splitting of the cells has rapidly spread in different implementation of adaptive mesh which can also evolve in time as for the Berger's algorithm. Another "ingredient" coming from this latter is the use of whole grids instead of single cells to be subjected the bisection refinement process. The main difference between the two approaches is that when working on cells the algorithm must be adapted to work on the "extravagant" stencils coming from the cell neighbours. The main advantage of this latter is the strictest possible control over the modified mesh, as each cell is checked and operated individually. Conversely, the work on grids allow the application of standard algorithms with more or less simple corrections on the grid connections where a refinement jump occur; the mesh is in this case less efficient, as a whole set of cells are processed in the same way.

Cell based quad-tree

This first class of methods consist into a quad-tree dynamical arrangement of different size of cells. One of the first application of the cell based quad-tree AMR is the *Gerris* code in Popinet and Zaleski [89]. It was oriented towards the treatment of complex geometries or interfacial computations, where local refinement are highly advisable. In a similar context this kind of AMR has been employed in the *Shallow waters* simulations in irregular boundaries domains such as lakes in Borthwick et al [11].

In the Popinet work a projection method with a multilevel Poisson solver are employed; a cut cell approach is used to deal with solid boundaries. Some small restrictions are imposed to the mesh generation in order to keep as simple as possible the computation of the stencils, such as the obligation for al the cut cell to have a neighbour at the same level. For the Poisson equation an integrated second order discretization is adapted to use the fluxes coming from the coarse-fine cell interfaces by using parabolic interpolations. If the pressure equations is

$$\nabla^2 \phi = rhs \tag{4.2}$$

this system can be solved through iterative methods (Jacobi, Gauss-Seidel) using a relaxation operator. The integration of (4.2) on a cell C of boundary ∂C becomes

$$\int_{\partial C} \nabla \phi . \boldsymbol{n} = \int_C rhs \tag{4.3}$$

of which the discrete version becomes:

$$\sum_d s_d \nabla_d \phi = ha\ rhs \tag{4.4}$$

where d is the direction, s_d the surface fraction in direction d and a the fluid volume fraction of the cell. The values $\nabla \phi_d$ for the ϕ cell, as represented in figure 4.5, can then be obtained as

$$h \nabla_d \phi = -\frac{\phi}{3} - \frac{\phi_1}{5} + \frac{8\bar{\phi}}{15} \tag{4.5}$$

Obviously, other interpolation stencils can be used as well. A similar treatment will be developed

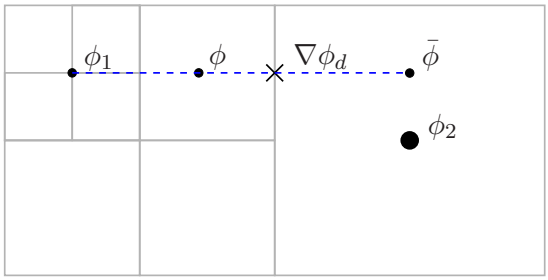


Figure 4.5: One option for the interpolation for the $\nabla \phi_d$ value.

in the present work when dealing with relaxation on the refinement jumps. The *Gerris* code has been recently employed to simulate interfacial flows with a VOF/CSF method Popinet [87]. One of its most remarkable results is the simulation of the primary atomization in Fuster et al [32].

Losasso et al [55] uses an analogue quad-tree AMR to simulate free surface flows such as smoke and water flows, with some improvements in the freedom of the mesh generation (*unrestricted* quad-tree) and in the construction of a symmetrical pressure equation matrix. The

accuracy of the pressure solution drops to first order when the matrix is symmetrical; the unsymmetrical matrix allows conversely a second order convergence. This is offset by the fact that a symmetrical matrix is mathematically easier (faster) to inverse with the use of fast iterative solvers like the Preconditioned Conjugate Gradient. Differently from the previous work, a Level Set method is used for the interface tracking. The aim is, as for the present work, through the use of higher order, hybrid and adaptive techniques, to overcome the numerical dissipation problem. The use of cell based quad-tree has undergone further investigation in Min and Gibou [69] and Min et al [70], in particular on the accuracy of a projection method with different Poisson equation discrete formulations.

Block based quad-tree

One alternative approach comes from both the "patch based" and the quad-tree philosophies, being a block based quad-tree (or oct-tree in three dimensions). In this case one (or more) initial blocks are used to partition the domain. Each block is composed by a fixed number of cells in each direction (it corresponds loosely to a Berger's "patch" [1]). When a block is refined, it is replaced by 2^{dim} children, each again composed by the same number of child cells, of which the extent is half that of the parent's cell. Coarsening is achieved by reversing this process and replacing the 2^{dim} children by their parent. The same relationship between grids defined in the Berger's decomposition are kept. The main difference is that in this algorithm the blocks are all equal in terms of cells number, they have a fixed number of children; in the Berger's, the patches can have arbitrary shape and dimensions, and a coarse patch can have any number of children (they can also be shared between more parents).

This kind of AMR has found great success especially in astrophysical and Magneto Hydro Dynamics (MHD) applications, because of the intrinsic multi-scale attribute of the problem. Those flows are highly compressible, so that Godunov-type techniques and Riemann solvers appear to be the most attractive approach. As for the incompressible Navier-Stokes equations, they have an added constraint of zero divergence: the magnetic field. In Powell et al [90] a MHD solver is built on a block based parallel AMR. At the interfaces of blocks that are at different refinement levels, states are constructed in two layers of ghost cells; a simple trilinear interpolation is used to construct the values in the ghost cells. No others solver modification related to the adaptive mesh appear. In a successive work Pancheshnyi et al [82] used the free parallel AMR libraries PARAMESH Peter MacNeice and Packer [85], Olson and MacNeice [78], Olson [77] to implement its solver [2]. The same libraries are used in the astrophysical simulation code FLASH, extensively tested in A. C. Calder et al [1]. Another approach for a MHD solver with block AMR is described in VanDerHolst and Keppens [124]. The quad-tree structure is modified to avoid the necessity to build all the children of a refined block, thus enhancing the locality of the mesh adaptation. The block quad-tree structure continues to be developed, as in the *Rancoon* code Dreher and Grauer [26], coded in the C++ language, mainly because of its superior parallel performances.

While adaptive patches, unstructured grids, and cell-based trees can all be used to support adaptive mesh refinement (the approaches have other uses as well), the quad-tree approach has several pronounced advantages:

- ▷ loop and cache optimizations can be performed over the arrays of cells in an adaptive block data structure. The use of ghost cells further enhances this advantage for calculations which depend on neighbouring cell values;

[1] In literature there is actually some confusion about this notation. In this context they are the same type of sub-grid with some ghost cells at their boundaries; however, a "patch" is built on an arbitrary number of nodes, a "block" comes from the bisection of its parent, and has its same fixed number of cells.

[2] The same libraries are used in the present work

- ▷ the added tree related data are fewer and simpler compared to other approaches, as there is regularity in dependencies;
- ▷ on parallel computers, adaptive blocks amortize the overhead of communication over entire blocks of cells, instead of over single cells as in unstructured grids;
- ▷ as the blocks are all computationally equal, the dynamic parallel workload balance is easier.

However, adaptive blocks can also have some disadvantages when compared to the alternatives:

- ▷ when the size of the array of cells per block is very large, then the average number of blocks per processor may become too small to allow an efficient workload repartition, as any processor having a number of blocks above the average will be doing significantly more work, causing the other processors to be delayed;
- ▷ large regions covered by refined mesh are fragmented into many small blocks, whereas a larger patch can cover the same space using less ghost cells and more contiguous memory space;
- ▷ it is more difficult to isolate small regions that really need refinement, as the refinement is block-based, not cell-based. More uninteresting intermediate scales than desired may appear.

As this strategy has been chosen for this work, more details are given in the next section, where the implementation of the AMR is described.

3.3 One Cell Local Multigrid

The OCLM method was developed in Vincent and Caltagirone [127] for the simulation of unsteady incompressible multiphase flows. Its main idea is to be able to refine and to find a finer solution around any "interesting" node (for example near to the interface). It does not change the distribution of the nodes, but any time the refinement criterion is fulfilled on a particular node, a small calculation domain (nine points) is built over the node, so that its volume of control is refined. The newly created points can be successively tested for refinement, creating a nested hierarchy of sub-grids. The most external points are usually interpolated from the coarser grid in order to obtain boundary data for the internal ones. This technique minimizes the number of refined points and the memory cost. Moreover, the newly created grids are rectangular, which is very convenient because the calculations can be carried out with an identical solver at each of the different grid levels. At each grid level l ($l = 0, 1..l_{max}$) s_{max} sub-grids are generated, starting from the coarsest fixed mesh G_0 which initially contains all the necessary information and represents the whole calculating domain. A fundamental property of the local mesh refinement is $\forall s,\ G_{l,s} \subset G_{l-1}$ as for many refinement techniques, starting from the fundamental Berger's Berger [9].

In Vincent and Caltagirone [127] and Delage et al [23] this method (which does not suffer from any loss of generality) is coupled with a VOF for the two phase treatment. A criterion related to the distance of a point from the free surface is used to build the multigrid architecture:

$$C_r = \|\nabla C\| \tag{4.6}$$

If $C_r = 0$ on each side of coarse cell centred at a pressure node, no refinement is carried out in this cell. If it is positive on one or more sides, then the free surface crosses the cell and a 9-point refinement is performed On this cell as shown in figure 4.6. The coarse grid G_0 is scanned and a set of s 3×3 sub-grids $G_{1,s}$ is built with a one third cell size. The odd cutting of the coarser

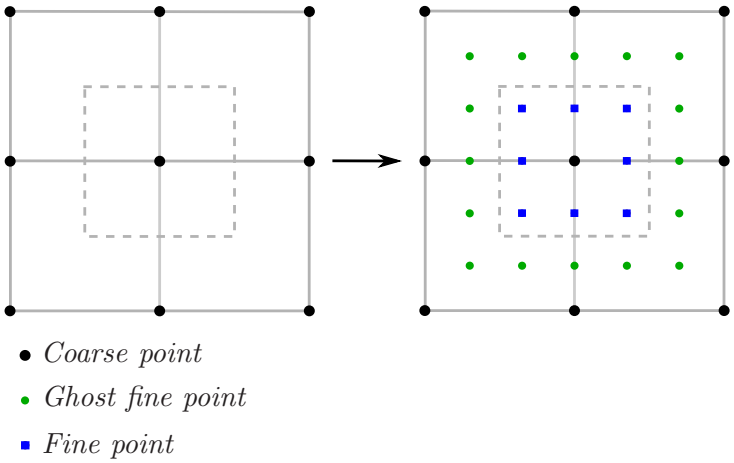


Figure 4.6: Refinement principle in the OCLM method.

cells in each space direction allows a natural connection between the sub-grids. The evolutions of the free surface can then be described at each of the different multigrid levels maintaining the stability and the precision of the interface capturing method. A local refinement procedure is then performed on each sub-grid in the same manner as on G_0. The sub-grids are thus refined recursively until the grid step becomes approximately equal to the smallest length scale of the problem considered, corresponding to a user defined precision criterion.

Because of the incompressibility condition, the fine grids need to be coupled with the coarse grids to achieve the implicit solving of the motion equation. The NavierStokes equation system is first solved on G_0. Next, pressure and velocity are initialised on each $G_{1,s}$ by a linear prolongation operator. The NavierStokes equations are then solved on these grids, and the procedure repeated on all of the multigrid levels. The prolongation operator brings a cell centered scalar ϕ from the coarse grid values to the immediately finer level:

$$\begin{aligned}\phi^l_{i,j} &= P_{l-1,l}(\phi^{l-1}, I, J) = \\ &= \left[1 - \frac{\mathrm{Inc}_x}{R_{ef}} - \frac{\mathrm{Inc}_y}{R_{ef}} + \frac{\mathrm{Inc}_x\mathrm{Inc}_y}{R^2_{ef}}\right]\phi^{l-1}_{I,J} + \left[\frac{\mathrm{Inc}_x}{R_{ef}} - \frac{\mathrm{Inc}_x\mathrm{Inc}_y}{R^2_{ef}}\right]\phi^{l-1}_{I+1,J} \\ &+ \left[\frac{\mathrm{Inc}_y}{R_{ef}} - \frac{\mathrm{Inc}_x\mathrm{Inc}_y}{R^2_{ef}}\right]\phi^{l-1}_{I,J+1} + \left[\frac{\mathrm{Inc}_x\mathrm{Inc}_y}{R^2_{ef}}\right]\phi^{l-1}_{I+1,J+1}\end{aligned} \tag{4.7}$$

where i, j are the indices on the fine grid, I, J on the coarse one, $\mathrm{Inc}_{x/y}$ the position on the refined mesh, R_{ef} the refinement rate (odd, equal to 3 in Vincent and Caltagirone [127]). Some change in the "Inc" indices allow the prolongation of face centered vector variables. Some enhancements have been carried on mainly in Santacreu [99] and Delage et al [23], in particular a technique of connection between refined meshes (which are disconnected in the original method), an extension of the refinement criterion in a more two phase oriented "physical" sense, and the implicitation of the interpolation into the implicit augmented Lagrangian solver.

4 Berger's refinement strategy

In early 1980's Marsha Berger Berger [9] began the development of an adaptive mesh refinement for structured grids based on the notion of multiple independently solvable grids of the same type, but different in number and dimension of cells. The underlying premise is that all grids of any given resolution are equivalent in the sense that, given proper boundary information, they

can be solved independently by identical means. That is, a multigrid concept adjusted to a set of resolution levels instead of grids, each of which employs a disjoint set of sub grids to cover progressively less of the domain (figure 4.7). In this sense, the grids are not supposed to move. The evolution of the mesh is put into the hands of the refining and de-refining operations: the grids are covered or replaced by new grids, or old ones are deleted. An explicit connectivity is given to each sub-grid in order to build the tree: each sub-grid can have multiple interactions with the surroundings. An underlying grid is called "parent" of the finer one; mutually the finer is a "child" of the coarser one. Grids at the same refinement level that share a boundary are "siblings" or "neighbours". The ensemble of grids form a hierarchy of resolution levels. Every grid is completely covered by some non-empty set of parent grids; both its active computational interior and its ghost boundaries, except for those that lie on the exterior of computational domain. In addition, the finer grids abut the coarser cells, so that the interior cell number of a fine grid is an integer multiple of the refinement factor.

When a grid is created or destroyed, its data have to be initialized in the first case, or saved in the latter. Two operations take in charge this work: the "prolongation" and the "restriction" operations respectively. The prolongation brings values from a coarser grid to a finer one; this can be achieved by different means, such as direct injection or interpolation. The accuracy and other characteristics (mean value, divergence, symmetry) of the prolonged data come directly from the chosen interpolation strategy. The opposite operation, the restriction, performs the transfer of data from a more to a less refined mesh. It is somewhat simpler because there is an implicit loss of precision in the removal of grid points. Again, interpolations of any kind bring different solutions. Each grid has some ghost cells surrounding the computational nodes, which give boundary information for the completion of the numerical stencils. The ghost cells are responsible for the communication between the meshes but also form a "buffer region" that allows the grid to be aware of the possible approach of the zone which requires refinement.

4.1 The algorithm

The integration algorithm works recursively through the levels, advancing each of them by the appropriate time interval, and inside the level through the different grids that compose the level. The Berger's algorithm is summarized in 1: Here the adaptivity in time can be seen as well:

Algorithm 1 Pseudocode of the Berger's algorithm.

Procedure Integrate($Level$)
$Evolve(Level)$
if $(Level/ = Level\_max)$ **then**
  for $r = 0, ref\_factor - 1$ **do**
    Integrate$Level + 1)$
  end for
end if
End

smaller grids require smaller time steps in order to respect the CFL conditions. So, for one time step on the coarse grid, multiple (depending on the refinement ratio) steps can be performed on the finer ones.

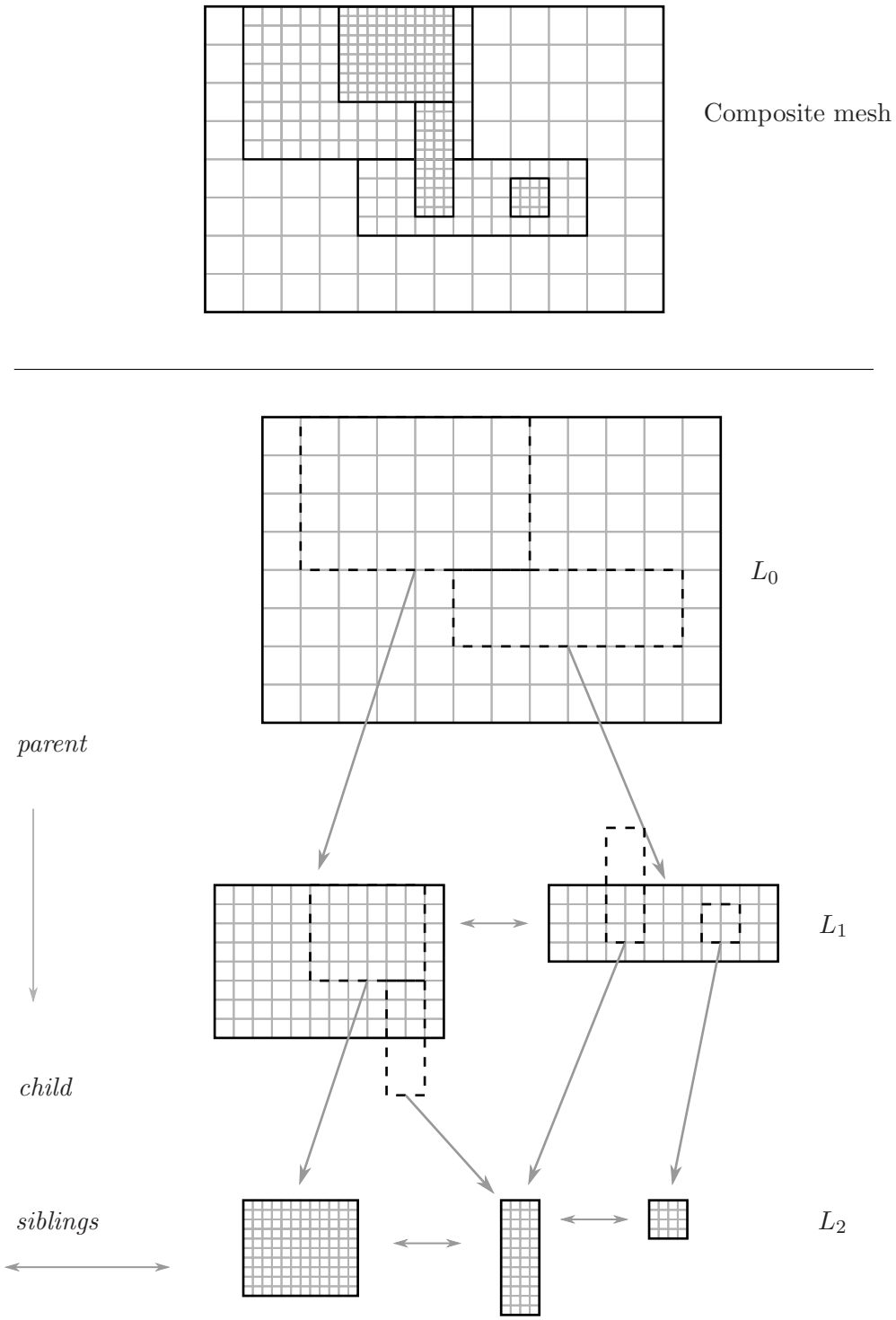


Figure 4.7: Representation of a hierarchy of increasingly fine nested grids.

4.2 Generation of the grids

During the refinement process new grids are created, while those no longer useful are deleted, in a recursive procedure. First of all, some refinement criterion has to be defined. In general, some computations are made on each cell, in order to evaluate some significant quantity, such as gradients. Actually, the most rigorous criterion would be a double computation (Richardson truncation error analysis) done on a mesh spacing h and one spacing $2h$. Where the difference is greater than a threshold, refinement should occur. However, the difficulty of having a computation made on two different meshes and the computational cost of this process often denies the advantages. The cells that satisfy this criterion are "tagged" (marked) for refinement. At this point, a particular process is needed: the clustering algorithm. This should transform a disordered set of tagged point into a sub-domain ready to be refined, often called "patch". There is a lot of freedom in the choice of the algorithm, because several criteria may be relevant: the desired shape (more rectangular or square), the dimension (the number of nodes), the efficiency (the tagged over total nodes ratio). Once the new grid are created, they must be linked to the tree: the connections with parents, siblings and children are established. One fundamental requirement for a newly created grid is that it has to be nested in respect to the others. In other words, a grid at a non-root level, including most of its ghost cells, should always cover a region at the immediately coarser level, except for the grids that abut the physical boundaries.

4.3 Refinement in time

Covering the domain with different levels of mesh allows temporal sub-cycling, or time refinement for explicit time stepping. The time step must be less than a certain time (CFL condition), otherwise the simulation will produce wildly incorrect results. This constrain depends on the cell size, in different ways according to the kind of equation and discrete terms. As a decreased cell size results into a smaller allowed time step, the step on the smaller grids can at most be a fraction of the one on the coarser grids. This can be exploited by performing multiple time steps on the fine grid each time one is done on the coarse one. In this case, however, one should link the coarse to the fine solution by imposing consistence of the numerical fluxes. As shown

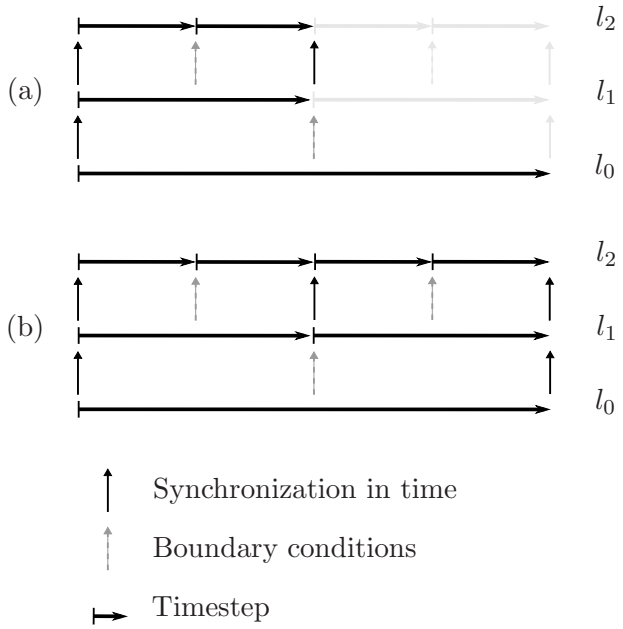


Figure 4.8: Temporal integration in Berger's algorithm: (a) Advancing and synchronization of fine levels (b) Advancing and synchronization of all levels.

in figure 4.8, once the level $l + 1$ data have been advanced to the same point in time as the level l data using grid-by-grid or level operations, synchronization of the data between levels is required. This synchronization attempts to correct for the difference in the solution generated by performing the update operations sequentially on the grids, rather than simultaneously on a composite hierarchy. In the present work adaptive time stepping has not been implemented, mostly because it tends to force ordering of the way the solution must be advanced on the different grid blocks, and this can have a damaging effect on parallel load balance.

4.4 Collection of ghost boundary data

Values of the ghost boundary cells surrounding each grid active region are collected from four sources (figure 4.9), as appropriate: copy from siblings, prolongation or restriction from parents and children respectively, or imposed by physical boundaries. The accuracy of the ghost cell re-

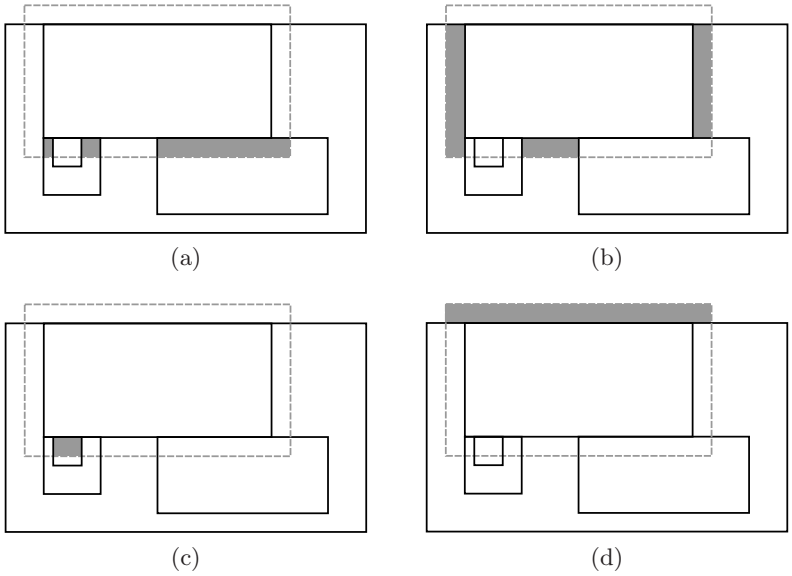


Figure 4.9: Four different ways ghost cells can be filled: (a) Copy from siblings (b) Prolongation from parents (c) Restriction from children (d) Physical boundary conditions .

construction can have a huge impact on the global accuracy of the computation. In addition, the imposition of Dirichlet boundary conditions does not allow special properties (curl, divergence) to be maintained nor derivatives and fluxes to be consistent at refinement jumps.

4.5 Flux correction for conservation

Berger's strategy allows conservative numerical schemes, with conservation achieved by an adjustment to the numerical solver and a flux correction step. Specifically, it is asked to the solver to give, at each time step, not only the computational variables, but also the computed fluxes across the coarse-fine interfaces: those fluxes are stored in special separated arrays for each grid. The fine fluxes are integrated over all the fine time steps that constitute a coarse time step if temporal sub-cycling is used, or at the end of each time step if not, and compared to the integral of the coarse one. Their difference, that corresponds to the inconsistency in the numerical computation, is used to apply a correction to the coarse grid solution (figure 4.10). In a more intuitive explanation, the amount of material, for example mass or momentum, that flows through an interface between a coarse and fine grid is recorded at each time step. If multiple fine time steps are performed, it is summed each time. If the coarse and fine computed quantities are the same at the synchronization point, then no correction is applied. In practice,

this is not likely to happen (conversely, it would indicate a really unnecessary refinement): the detected difference of mass flowing through the interface is removed so that the global amount is maintained.

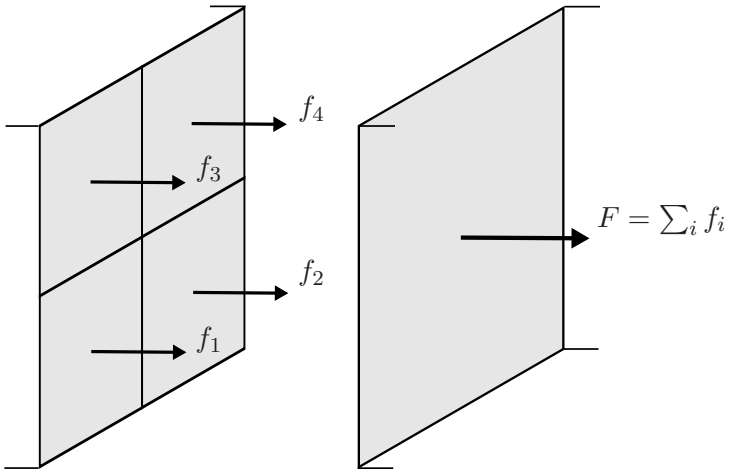


Figure 4.10: Flux matching condition for a two dimensional coarse-fine interface. This condition assures the global conservation of a quantity through a coarse-fine interface.

5 The PARAMESH package

PARAMESH (Peter MacNeice and Packer [85], Olson and MacNeice [78], Olson [77]) is a package of Fortran 90 subroutines, designed to provide an application developer with an easy route to extend an existing serial code which uses a logically Cartesian structured mesh into a parallel code with a block based quad-tree AMR. Alternatively, in its simplest use, and with minimal effort, it can operate as a domain decomposition tool for users who want to parallelize their serial codes, but who do not wish to use adaptivity. The package is distributed as source code which will enable users to extend it to cover any requirements.

5.1 Setting-up the mesh

The PARAMESH package builds a hierarchy of sub-grids to cover the computational domain of the application. These sub-grid blocks form the nodes of a tree data structure (quad-tree in 2D or oct-tree in 3D). All the grid blocks have an identical logical structure. An initial set of n-dimensions blocks is defined by the user to cover the whole computational domain: this set of grids become fixed and it is never deleted during the computation, and all the blocks have a fixed number of cells $nxb \times nyb \times nzb$ and of guardcells $nguard$ (figure 4.11). The first refinement steps will produce 2^n child blocks, each with its own $nxb \times nyb \times nzb$ mesh, but now with cells being one-half of their parents (the refinement level is not allowed to jump by more than one refinement level at any location in the spatial domain). Any or all of these children can in turn be refined, in the same manner. This process continues recursively until the domain is covered with a quilt-like pattern of blocks with the desired spatial resolution everywhere (figure 4.12). Each fine block which does not have any children is called "leaf block". The union of all the leaf blocks form a composite mesh that covers all the computational domain, and corresponds to what one would see if he looked from above the grid tree (just as in figure 4.12).

The grid blocks are assumed to be logically Cartesian (or structured): within a block the grid cells can be indexed as though they were Cartesian. If a cell's first dimension index is i, then it lies between cells $i-1$ and $i+l$. The actual physical grid geometry can be Cartesian,

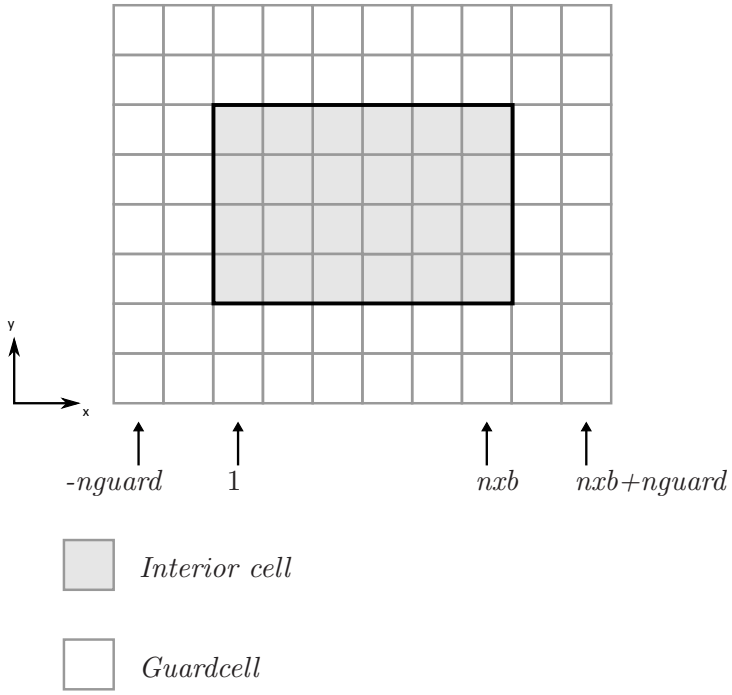


Figure 4.11: A two dimensional block with a 6 x 4 mesh and two guardcells.

cylindrical, spherical, polar, or any other metric which enables the physical grid to be mapped to a Cartesian grid. The metric coefficients which define quantities such as cell volumes are assumed to be built into the user's algorithm. Each grid block has an user prescribed number of guard cell layers at each of its boundaries. These guard cells are filled with data from the appropriate neighbour blocks, or by evaluating user prescribed boundary conditions, if the block boundary is part of a boundary to the computational domain. The package supports 1D, 2D, axisymmetryc and 3D models; the mesh supports different types of variables: cell, face and edge centered.

Along with the creation of the grids, the whole tree containing the connectivity between blocks is generated. Each block knows its geometry, its position in physical space, its parent, its children and neighbours, and the processor where it and the other blocks are stored. A particular flag is given to those blocks which lies on a physical boundary: here a special treatment of ghost cells is applied. Requiring that all grid blocks have identical logical structure, may, at first sight seem inflexible and therefore inefficient. However this has two significant advantages. The first and most important is that the logical structure of the package is considerably simplified, which is a major advantage in developing robust parallel software. The second is that the data-structures are defined at compile time which gives modern optimizing compilers a better opportunity to manage cache use and extract superior performances.

5.2 Load balance

The PARAMESH package uses the MPI standard for the parallel communication. The principle of its efficient parallelization is that a quad-tree generated adaptive mesh is composed by several (computationally speaking) equal blocks, and if the same number of blocks is assigned to each processor, they will likely have to do the same amount of work. The created blocks are distributed amongst the processors in an effort to achieve load balance and to lower communication cost by improving the data locality. A Peano-Hilbert (or Morton) space filling curve is drawn through the list of grid blocks (an example in figure 4.13); a value is given to each block based to its

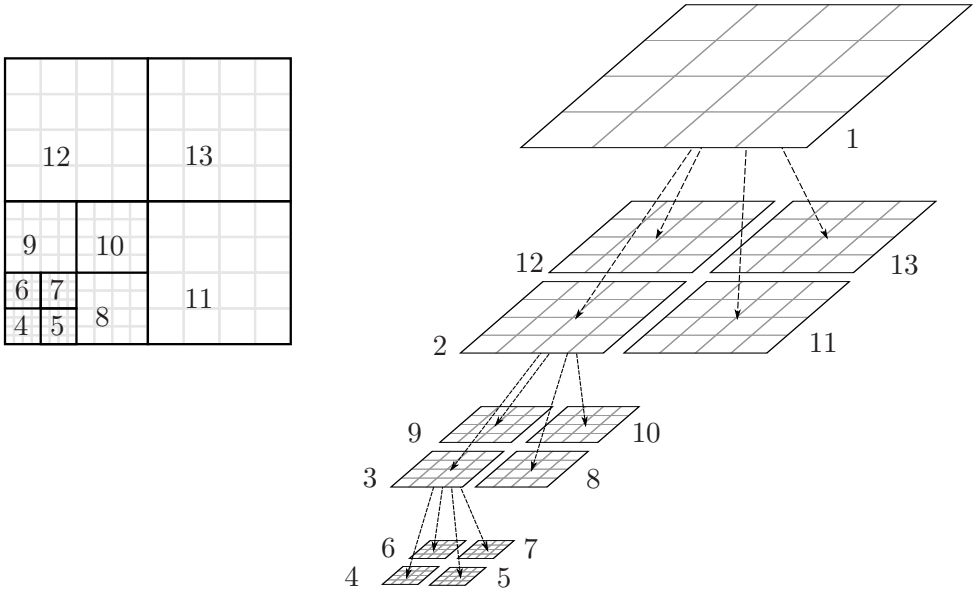


Figure 4.12: PARAMESH grids and numerotation, four levels of refinement.

position along this curve. The interest of this curve is that points close in its numeration are likely to be close in physical space. The list of blocks is sorted following this curve, so that the chance of having neighbouring blocks on the same processor are increased. An system of block weighting is added to allow some customization of this process. A high weight means that the block will have the priority in the Morton numeration; conversely, lower values define the block as less relevant in the balancing process. This is important because frequently the solution is not advanced on the whole grid tree, but on the leaf blocks only, with a limited use of their parents for guardcells filling purposes; so it is important that the leaf and their parent blocks are well balanced among the processors, while the other blocks are not actively used. The standard weighting gives full weight to leaf blocks, half to the parents and zero to everyone else. It is possible for the user, however, to change this value, in particular if all the blocks are kept working. A positive consequence of this parallel handling is that the number of processors has not to be defined at the beginning, as in a uniform mesh code, but the same already compiled program may run with different processor numbers.

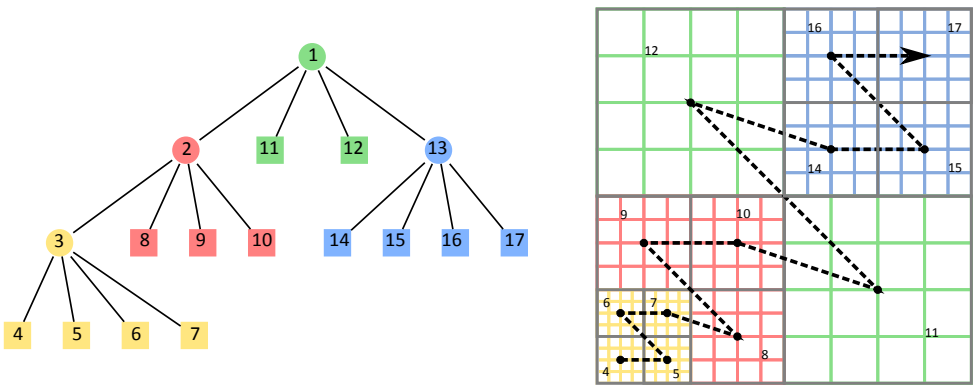


Figure 4.13: PARAMESH parallel distribution to a four processors machine using a Peano-Hilbert space filling curve.

5.3 Development of an application

The PARAMESH package supplies subroutines for all the main AMR operations, with very little user intervention: it manages the creation of the grid blocks, builds and maintains the tree-structure which tracks the spatial relationships between blocks, distributes the blocks amongst the available processors and handles all inter-block and inter-processor communication. The main steps in developing a code can be summarized as follows:

- ▷ edit a few lines in the header files which define the model's spatial dimensionality and the properties of the block-tree;
- ▷ construct a main program: a sequence of calls to the 'upper level' routines in the PARAMESH package will be the main alteration;
- ▷ build the own solver inside the PARAMESH framework;
- ▷ build some routines to handle conservation and coarse-fine interfaces;
- ▷ supply user external boundary conditions.

The sequence of the operations include the generation of an initial grid, the possible computation of a time step, the real advancing of the solution, the refinement test and, if this is positive, the regridding operations. Consequently the pseudo code of a PARAMESH-using main program can be summarized as in the algorithm 2.

Algorithm 2 Pseudocode of the main program.

```
Program main

Create initial mesh
Create initial condition
Set boundary conditions
for (t = 0, t_max) do
   Compute Δt
   Advance the solution on all the blocks
   Perform restriction where needed
   Perform refinement test
   if (Test is positive) then
      Perform creation new/deleting old grids
      Perform prolongation on new grids
   end if
end for

End
```

6 Grid operations

6.1 Prolongation

The prolongation operation $\mathcal{P}$ transfers data from a coarse grid to a finer one. In the AMR code this has two main purposes: to initialize data on newly created blocks and to supply boundary

conditions to a fine grid from the underlying coarser one. In the multigrid algorithm it takes also part in the two grid cycle. When the dedicated routine is called, all the blocks at level $l_{max} - 1$ transfer their data to the level l_{max}. The algorithm that performs the transfer is user defined; however, some standard interpolation routines are given within the package. Among the possible strategies of interpolation are: the direct injection, the Lagrange polynomial interpolation, divergence preserving interpolation. Being the computation performed on a staggered mesh, different algorithms are employed for the cell centered and the face centered variables. The presented algorithms are two dimensional without loss of generality: the interpolations can be extended in three dimensions as well (in certain cases the computation of the coefficients may become complex). All the proposed interpolation have the characteristics of free-stream preservation, which means that a constant field is always prolonged into a constant field, and they are symmetrical on the horizontal and vertical axes.

Cell centered variables

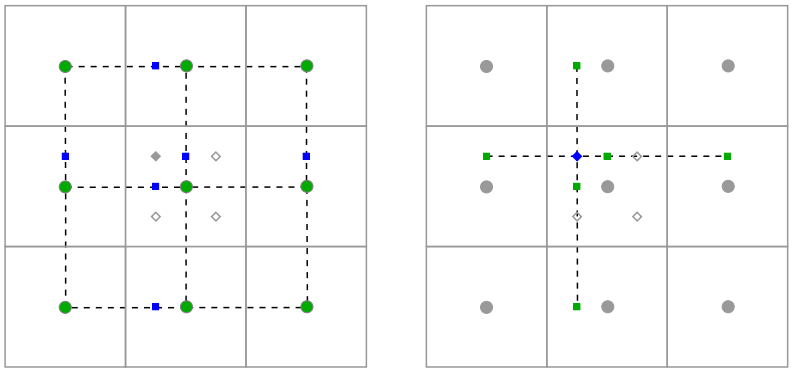
The cell centered variables are the pressure and the Level Set nodes. The prolongation associates on the to-be-refined zone Ω, ref to the scalar field $\phi^l_{\Omega,ref}$ at level l the scalar field $\phi^{l+1}_{\Omega,ref} = \mathcal{P}(\phi^l_{\Omega,ref})$ at level $l+1$. Each coarse node $\phi_{i,j}$ spawns four children $\phi_{i\pm\frac{1}{4},j\pm\frac{1}{4}}$ when refined. Interpolation of zero, first and second order interpolation have been tried in the different solvers.

- ▷ Injection: this zeroth order interpolation is the simplest, as it associates to four children the value of their parent
$$\phi^{l+1}_{i\pm\frac{1}{4},j\pm\frac{1}{4}} = \phi^l_{i,j} \tag{4.8}$$
It is often insufficiently accurate for the AMR purposes, mainly in the guardcell filling operation. The characteristic of the discrete solution are, however, exactly maintained.

- ▷ Linear interpolation: in this first order interpolation two sets of linear interpolation, the first one to evaluate intermediate values on the coarse points axis, the second one to find the final value from the ones computed before. The value can be expressed as
$$\phi^{l+1}_{i\pm\frac{1}{4},j\pm\frac{1}{4}} = (\phi^l_{i\pm1,j\pm1} + 3\phi^l_{i\pm1,j} + 3\phi^l_{i,j\pm1} + 9\phi^l_{i,j})/16 \tag{4.9}$$
which, due to the linearity of the operator, corresponds to the opposite area weighting.

- ▷ Bi-quadratic interpolation: this second order interpolation has been built to be used with the multigrid for the elliptic equation. Again, given the disposition of coarse and fine points, two sets of interpolation are needed, and nine points stencil to obtain the interpolation coefficients (the final coefficients are computed once and remain the same, to speed up the computation [1]). An example is presented in figure 4.14: the four fine points represented by the small rhombus are computed from the nine coarse ones, the circles. A first set of quadratic interpolation in the two directions is performed to find the intermediate points, the small squares; then the set of quadratic interpolations is repeated for the intermediate points, as in figure 4.14b. The average of the two curves in their crossing point is the searched value.
$$\phi^{l+1}_{i\pm\frac{1}{4},j\pm\frac{1}{4}} = \sum_{m,n} w_{m,n}\phi^l_{i+m,j+n} \ , \ m,n = -1,0,1 \tag{4.10}$$
The matrix of coefficients W contains the weights $w_{m,n}$ of the 9 cells $\times$ 4 points = 36 points which can be interpolated from the nine points coarse stencil(the computation of

[1] A lot of computational time is spent into the interpolation process, especially in 3D



(a) (b)

Figure 4.14: Bi-quadratic symmetrical interpolation for cell centred variables. (a) First set of quadratic interpolation (b) Second set of quadratic interpolation.

the weights W is detailed in annexe 1). Actually, only the four internal ones are used in this work, as the other 32 come from a shifted stencil, useful if one does not want to use the coarse guardcells. In this case, points of the coarse stencil may lay outside the coarse block; in other words, it can belong to its guardcells set. This means that, inside a prolongation operation, the guardcell filling must be performed. However, a closer look to the prolongation from parent to child reveals that the blocks surrounding the coarse parent are at the same level of refinement of the parent itself, otherwise there would be a jump in refinement of more than one level: this way, this guardcell filling consists only of a copy from and to blocks at the same refinement level.

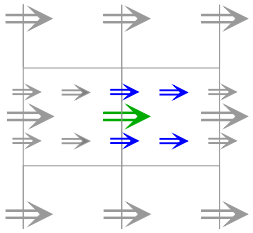
Face centered variables

The face centered variables correspond to the components of the velocity vector $\boldsymbol{u} = (u, v, w)$. The same polynomial interpolation as for the cell centered variables are employed, but the geometry of the variables location is different. Given the symmetry of the velocity treatment, examples are here related to the horizontal u component alone, the same applies to the others. Each u (staggered horizontally by a half cell size from the cell center) lies on the vertical face of the cell. When the cell is halved, a pair of fine u nodes appear alongside the coarse one; two more nodes appear on the x location corresponding to the cell center. The polynomial interpolation are obtained as follows:

- ▷ Injection: the value of the coarse u is copied into the four children, resulting in the exact translation of the vector field into the finest mesh without any modification, and a zeroth order accuracy (figure 4.15).

$$\begin{cases} u^{l+1}_{i,j\pm\frac{1}{4}} = u^{l}_{i,j} \\ u^{l+1}_{i+\frac{1}{2},j\pm\frac{1}{4}} = u^{l}_{i,j} \end{cases} \tag{4.11}$$

- ▷ Bi-linear interpolation: a linear interpolation allows a first order accuracy reconstruction of the velocity field, preserving its discrete divergence. A first linear interpolation between the $u^l_{i,j}$ and its neighbours $u^l_{i,j\pm1}$ gives the first pair of prolonged values; the horizontal linear interpolation of these latter give the second pair (figure 4.16). The final coefficients

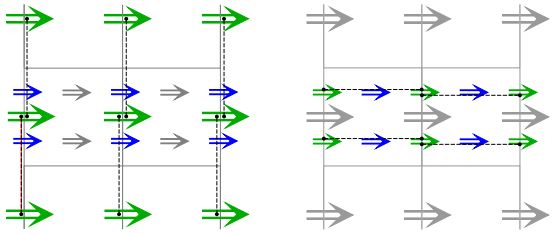


(a)

Figure 4.15: Injection for the face centered values. Green: the interpolation stencil; blue: the interpolated value; black dotted lines: the interpolation direction; grey: points not used.

are, as computed in the work of Zeng Zeng and Wesseling [133]:

$$\begin{cases} u^{l+1}_{i,j\pm\frac{1}{4}} = (3u^l_{i,j} + u^l_{i,j\pm1})/4 \\ u^{l+1}_{i+\frac{1}{2},j\pm\frac{1}{4}} = (3u^l_{i,j} + 3u^l_{i+1,j} + u^l_{i,j\pm1} + u^l_{i+1,j\pm1})/8 \end{cases} \tag{4.12}$$



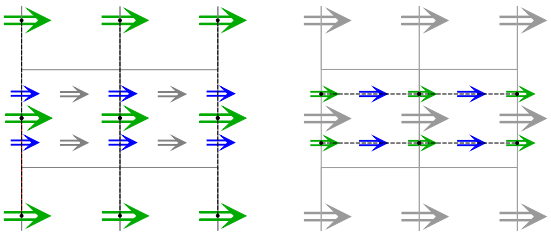
(a) (b)

Figure 4.16: Linear polynomial interpolation for the face centered values. Green: the interpolation stencil; blue: the interpolated value; black dotted lines: the interpolation direction; grey: points not used.

- ▷ Bi-quadratic interpolation: this interpolation works as the linear one, but it uses a wider stencil of three points and it is second order accurate. As for the previous case, a first set of interpolation allows the computation of the aligned pair of children nodes; a second one uses the freshly computed values to interpolate the mid-cell ones, as in figure 4.17. As it is composed by three points, for these latter nodes a single quadratic interpolation would involve a staggered interpolation stencil. This may lead to a loss of symmetry on the vertical axe: for this reason, the left and right valued for both the "candidate" stencils are averaged (the coefficients computation is detailed in Annexe A). This reconstruction is accurate but does not prevent local overshooting of the coarse solution on the fine mesh. One possible solution would be to use slope limiters; actually, in none of the simulations performed in this work any instability has arisen for this reason. A second potential loss is the discrete divergence of the velocity field. However, this can be corrected in case of necessity by the projection of the interpolated velocity field (as proposed by Colella) by a resolution of an elliptic equation: the cleaning is finally achieved by adding the gradient of a scalar potential, computed by solving a Poisson equation with the discrete divergence

as source term.

$$\begin{cases} u^{l+1}_{i,j+\frac{1}{4}} = \left(\frac{3}{8}u^{l}_{i,j+1} + \frac{3}{4}u^{l}_{i,j} - \frac{1}{8}u^{l}_{i,j-1}\right) \\ u^{l+1}_{i,j-\frac{1}{4}} = \left(-\frac{1}{8}u^{l}_{i,j+1} + \frac{3}{4}u^{l}_{i,j} + \frac{3}{8}u^{l}_{i,j+1}\right) \\ u^{l+1}_{i+\frac{1}{2},j\pm\frac{1}{4}} = \left(-u^{l+1}_{i-1,j\pm\frac{1}{4}} + 9u^{l+1}_{i,j\pm\frac{1}{4}} + 9u^{l+1}_{i+1,j\pm\frac{1}{4}} - u^{l+1}_{i+2,j\pm\frac{1}{4}}\right)/16 \end{cases} \tag{4.13}$$



(a) (b)

Figure 4.17: Bi-quadratic polynomial interpolation for the face centered values. Green: the interpolation stencil; blue: the interpolated value; black dotted lines: the interpolation direction; grey: points not used.

▷ Toth-Roe prolongation: Toth and Roe [120] presented prolongation and restriction formulas aimed to the preservation of properties of the solution which are critical for the scheme, and which should also be defined appropriately at resolution changes. Examples are the divergence and, in some cases, the curl of a discrete vector field. The formulas are thought for adaptive and hierarchical mesh algorithms with a refinement rate of two. In case of interpolation of face centered variables as the zero divergence field of the incompressible Navier-Stokes equations, the fine grid reconstruction looks like

$$\begin{cases} u^{l+1}_{i,j+\frac{1}{4}} = u^{l}_{i,j} + \left(u^{l}_{i,j+1} - u^{l}_{i,j-1}\right)/8 \\ u^{l+1}_{i+\frac{1}{2},j\pm\frac{1}{4}} = \left(u^{l}_{i,j} - u^{l}_{i-1,j}\right)/2 + \\ \qquad \left(u^{l}_{i,j+1} + u^{l}_{i-1,j+1} - u^{l}_{i,j-1} - u^{l}_{i-1,j-1}\right)/16 + \\ \qquad \left(v^{l}_{i+1,j} + v^{l}_{i-1,j} - v^{l}_{i+1,j-1} - v^{l}_{i-1,j-1}\right)/16 \end{cases} \tag{4.14}$$

and result a second order accuracy reconstruction (stencil represented in figure 4.18).

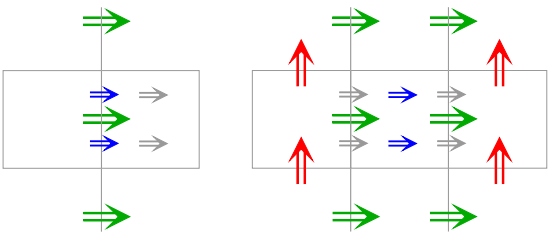


Figure 4.18: Toth divergence preserving polynomial interpolation for the face centered values. Green: the interpolation stencil; blue: the interpolated value; black dotted lines: the interpolation direction; grey: points not used.

- ▷ Balsara prolongation: more or less in parallel with the development of Toth and Roe, Balsara [7] developed similar formulas for the divergence preserving prolongation of continuous fields. The results being obtained independently and in a different manner, both give a second order approximation of the coarse field preserving the $\nabla \boldsymbol{u} = 0$ condition (and both the works come from magnetohydrodynamics considerations). A remarkable characteristics of the Balsara reconstruction is that it follows the TVD (total variation diminishing) principle, where no numerical overshoots of the solution are allowed, dramatically improving stability. The details of the reconstruction are given in annexe A.

6.2 Restriction

The restriction operation $\mathcal{R}$, conversely to the prolongation, brings variables from a fine grid to a coarser one. It is called every time an old block is deleted in order to conserve its data and reinitialize its parent. Again, before any guardcell filling operation a call to the restriction is performed in order to give accurate fine grid boundary values to the surrounding coarse grids. As for the prolongation, different interpolations can be employed. However, in the experiences of this work, it seems that the precision of the restriction has much less impact on the global accuracy of the code.

Cell centered variables

For the restriction of cell centered variables four fine point placed at the edges of a square must transfer their value on the coarse point situated at their center. Using the PARAMESH standard interpolation routines some tests have been performed to evaluate differently accurate restrictions:

- ▷ Average: the simplest restriction algorithm, where the value of the coarse point is the average of its our children; this is equivalent to a linear interpolation, it is the most "natural" restriction for the cell centered values, and it does not break the symmetry of the stencil;
- ▷ Quadratic (or higher) interpolation: the value of the coarse point is the result of quadratic interpolations from the surrounding fine points; however, the geometry of the cell centered variable does not allow a symmetrical stencil without extending the stencil to sixteen points, without a real enhancement of global accuracy.

Face centered variables

Similar considerations as those done for the cell centered variables applies to the face centered ones. The code seems to work well with simple restrictions; moreover a high order restriction would make more complicated the flux fixing procedure for conservation of the flow.

- ▷ Average: differently from the cell centered variables, for the face centered the coarse points are surrounded by a pair of fine points, while the second pair can be ignored given their position. Toth and Roe [120] used in its velocity restriction a simple average of the nearest two points;
- ▷ Higher order interpolation: using more points in the tangential direction of the face allows the use of more accurate restriction, always ignoring the half coarse cell fine values;
- ▷ Wider stencils interpolation: the value of the coarse point is the result of higer order interpolations from more of the surrounding fine points; up to twelve values are used in Zeng and Wesseling [133] in this process.

The restriction can be limited from the leaf blocks to their parents, or well extended down the tree until the coarsest mesh is reached. In general the first operation is sufficient; in multigrid algorithms, for example, this is done recursively until all the tree levels are updated.

6.3 Guardcell filling

The guardcell filling is the operation devoted to the update of boundary conditions of each block. It can be directly called from the user when needed or automatically called in order to support other operations. Actually this operation consists of three phases which can eventually be called separately. The simplest way to fill the ghost region is to perform a copy from the neighbours which are at the same level of refinement. If the block has neighbouring leaf blocks this is trivial. If its neighbour is more refined, then its parent will be at the same refinement level. Given that at the end of each time step the parent blocks at any refinement level are not updated, prior to the guardcell filling, a global restriction operation from leaf to parents is done (figure 4.19 step (1)). The last case is a block that shares its border with a less refined neighbour: in this case a prolongation is called from its underlying coarser blocks (figure 4.19 step (2)), now fully updated. Each block has eight surrounding regions, given by its four faces N,E,S,O (plus, if

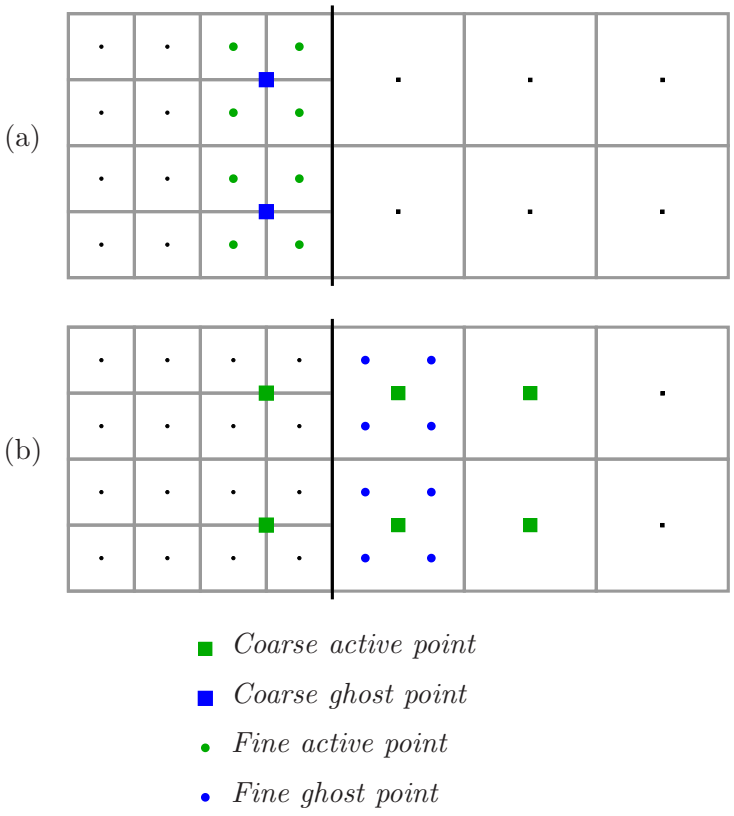


Figure 4.19: Two steps of the guardcell filling procedure: (a) A restriction is performed from the leaf blocks nodes (green circles) to the parent nodes (blue squares); then guardcells are copied between the coarser blocks; (b) Prolongation is called to transfer data from the parent cells (green squares) to the ghost cells of the children (blue circles).

needed, the four diagonals NE,NO,SE,SO), multiplied by the (user defined) number of ghost cells. Those regions are sequentially covered, the values imposed by the block physically lying in the same space. If an external boundary is found, then a dedicated subroutine is called. Here the user can write any boundary condition he needs; different boundary conditions can be imposed in different parts of the domain. If a periodic boundary condition are needed, then the

first coarsest block will have himself as neighbour in all directions. When a parallel computation is done, data have to be exchanged between processors. In order to realize the communication with a minimal cost, all those off-processor informations are packed and communicated before starting to work on the block list.

6.4 Refinement criterion

In the quad-tree AMR strategy the refinement flag is block based, conversely to the original Berger's algorithm where each cell can be tagged for refinement/de-refinement. The clustering procedure is avoided. Instead, inside each block some kind of test has to impose the value of two flags, "refine" and "de-refine". When a contradictory order is given to the blocks, the re-gridding routine makes sure the proper nesting is always maintained. The user must define a minimal and a maximal level of refinement, as well as a maximum numbers of allowed blocks. If the two values coincide, then an uniform mesh will be built.

6.5 Flux conservation

When neighbouring grid blocks have different levels of refinement, the fluxes at the common boundary used to update the solutions on the two blocks may not be completely consistent. This will lead to a loss of conservation if not treated. PARAMESH provides some routines which will update the fluxes on the coarser of the two blocks with data provided by the finer block. This data should be captured while the user's algorithm is computing the appropriate fluxes and advancing the solution. The strategy here is to record the inconsistent fluxes used at the block boundaries as the solution is advanced, modify them to achieve consistency, and finally correct the advanced solution for the differences between the original and modified fluxes. Consistency across an interface refinement jump is needed to maintain conservation, accuracy,

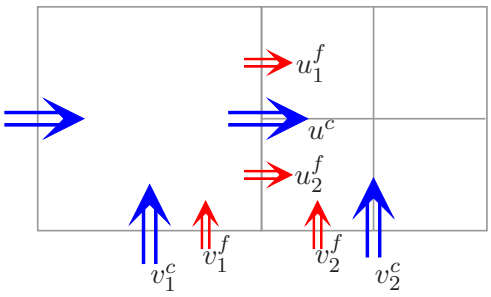


Figure 4.20: Velocity components at coarse-fine interface.

and stability within the method. Ideally, C^1 continuity of all the variables across an interface would preserve consistency. In a discrete differencing sense, this would require that the finite difference approximations to the first derivative match for the coarse and fine blocks at the interface. As illustrated in figure 4.20, a consistent shear stress at the interface would require

$$\frac{v_1^c - v_2^c}{\Delta x^c} = \frac{v_1^f - v_2^f}{\Delta x^f} \tag{4.15}$$

However, a particular interpolation stencil should be used to match derivatives across an interface match by definition. In the present method, shear stress is not conserved at an interface for this reason. The flux across the interface is simply the velocity component normal to the interface. To maintain mass conservation across the interface, these coarse and fine fluxes must be consistent. Using the variables in figure 4.20, the following equation must be respected:

$$u^c \Delta x^c = u_1^f \Delta x^f + u_2^f \Delta x^f \tag{4.16}$$

Assuming the fine values to be more accurate than the coarse, the u^c is corrected to make equation (4.16) hold. This flux correction is applied prior to the guardcell filling operation to assure the blocks to share consistent data. In the code three times this operation is called: at the beginning of the time step on the $\boldsymbol{u}^n$ vector, before the velocity advection (because a mesh refinement or de de-refinement may have altered the block boundary values); to the computed predictor velocity $\boldsymbol{u}^*$; after the correction, on the final projected velocity $\boldsymbol{u}^{n+1}$ to assure consistency for re-meshing operations and to check the global conservation (it would be possible to drop this latter without damage to the mass conservation, but it is likely to provide a more consistent prolongation).

Conclusion

In this chapter the implementation of the adaptive mesh refinement (AMR) into the two phase flow solver has been described. A general overview of different AMR techniques quickly focuses on the original hierarchical Berger's algorithm for structured meshes, and successively one of its derivation, the *quad-tree* AMR. This latter generates refined grids by splitting the coarser ones in two in all the directions, in a recursive dichotomous process. It has been implemented into the code by the use of the open source PARAMESH package, which is able to manage the grids to allow an efficient parallel distribution: it employs a Peano-Hilbert curve to numerate the block and allocate them emphasizing locality. Detailed description of how it works and the associated AMR specific operations, like interpolation and grid communication, has been given.

Chapter 5

The Elliptic Solver

Contents

Introduction

In literature, the resolution of elliptic equations is a constant obstacle for efficient solvers, especially in astrophysics, magnetohydrodynamics and incompressible flow solvers. A great impulse in the efficient solving to these equation has been given by the multigrid method, an iterative solver which is able to keep the convergence rate independent on the mesh size. In parallel, these applications being often multi scale, they have asked for a more efficient dynamic mesh distribution. It is clear that AMR and multigrid share a similar base idea, that an uniform fixed mesh cannot be the best strategy for some determined problems. The logical solution is to make them work together, in a more independent way as it will be realized in this work, or more intrinsically linked together in real *multilevel* algorithms.

This chapter is dedicated to the development of an iterative solver for the resolution of elliptic equations on an AMR hierarchy of grid. In the second step of the Chorin projection method the pressure field is computed by solving the elliptic equation (2.6) in order to project the velocity into a divergence free field. This equation is linear, but the coefficients, containing the densities of the two fluids, are only constants inside each fluid. They are discontinuous on the interface, the jump proportional to the density ratio. A strong variation on some of the discretization matrix coefficients makes the conditioning number to rise, resulting in a linear

system more difficult to solve. The presence of the adaptive mash adds another obstacle to the resolution of the equation, because of the communication between different grids sizes, and an added complexity of grid structure for multigrid type solvers.

All the computations in this chapter are performed on the same SGI ALTIX ICE 8200 architecture, Fortran compiler fce 11.1.059 and SGI MPT 1.26 libraries.

1 The Poisson equation

The Poisson equation is a partial different equation that can describe a wide range of problems, such as electrostatics, mechanical engineering, astrophysics and fluid mechanics. Its general form is

$$\Delta\phi = f \tag{5.1}$$

where Δ is the Laplacian operator, ϕ is the unknown function, sometimes referred as *lhs* for left hand side, and f is the source term, in turn called *rhs* for right hand side term, and a given set of boundary conditions. When the coefficients are not constant, the equation takes the form

$$\nabla\left(\beta\nabla\phi\right) = f \tag{5.2}$$

In the projection step, the coefficients β of ϕ are corresponding to $1/\rho$; the unknown ϕ is the pressure or the potential (they differ from a Δt) and the source term is the divergence of the predicted velocity field $\Delta\mathbf{u}^*$. The boundary conditions are, in general, homogeneous or imposed Dirichlet or homogeneous Neumann. There are different techniques for solving a Poisson problem, divided at first in exact and iterative solvers. The methods of the first family, such as Gauss elimination, are rapidly excluded as they cost is excessive for large dimension problems. Conversely, the iterative solver (which may become an exact solver if enough iterations are allowed, as happens for the Conjugate Gradient Method) can give a reasonably accurate solution in shorter times, especially when the system associated matrix has special properties. In general, the advantage of iterative solvers starts to manifest when they face sparse matrices, of which a great part of the elements are zeroes. The classical discretization of elliptic equations on Cartesian regular structured mesh gives matrices with some diagonals filled, and they are characterized by a diagonal dominance

1.1 Discrete Poisson equation

The position of the pressure unknown is, in this formulation, cell centered. This means also that a general boundary value $\phi_{0,j}$ or $\phi_{i,0}$ does not belong to the physical boundary of the domain, but it has to be evaluated consequently. In two dimensions the Poisson equation is defined as

$$\nabla\left(\beta(x,y)\nabla\phi\right) = \frac{\partial}{\partial x}\left(\beta(x,y)\frac{\partial\phi}{\partial x}\right) + \frac{\partial}{\partial y}\left(\beta(x,y)\frac{\partial\phi}{\partial y}\right) = f(x,y), \text{ on } \Omega \tag{5.3}$$

with a set of boundary conditions

$$a(x,y)\frac{\partial\phi}{\partial n} + b(x,y)\phi = f(x,y), \text{ on } \partial\Omega \tag{5.4}$$

The centered second order discretization, when β is substituted by its actual meaning $1/\rho$, becomes

$$\begin{aligned}\left[\frac{\partial}{\partial x}\left(\frac{1}{\rho}\frac{\partial\phi}{\partial x}\right) + \frac{\partial}{\partial y}\left(\frac{1}{\rho}\frac{\partial\phi}{\partial y}\right)\right]_{i,j} &= \frac{1}{\rho_{i+\frac{1}{2},j}}\left(\frac{\phi_{i+1,j}-\phi_{i,j}}{\Delta x^2}\right) - \frac{1}{\rho_{i-\frac{1}{2},j}}\left(\frac{\phi_{i,j}-\phi_{i-1,j}}{\Delta x^2}\right) \\ &+ \frac{1}{\rho_{i,j+\frac{1}{2}}}\left(\frac{\phi_{i,j+1}-\phi_{i,j}}{\Delta y^2}\right) - \frac{1}{\rho_{i,j-\frac{1}{2}}}\left(\frac{\phi_{i,j}-\phi_{i,j-1}}{\Delta y^2}\right) = f_{i,j}\end{aligned} \tag{5.5}$$

If the coefficient $1/\rho$ is constant, under the hypothesis (in this work always true) of $\Delta x = \Delta y = h$, the resulting 2D discretization stencil form of the equation (5.5) is

$$D = -\frac{1}{\rho h^2}\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \tag{5.6}$$

The whole system $m \times n$ of equations can so be expressed, given the ordered unknowns vector $U = (\phi_{11}, \phi_{2,1}, ..., \phi_{m,1}, \phi_{1,2}, \phi_{2,2}, ..., \phi_{m,n})^T$, the right hand in the same form and I the 3×3 identity matrix, by the matrix form $[A][U] = [F]$ where

$$A = \frac{1}{\rho h^2}\begin{bmatrix} D & -I & 0 & 0 & 0 & \dots & 0 \\ -I & D & -I & 0 & 0 & \dots & 0 \\ 0 & -I & D & -I & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & 0 & -I & D & -I & 0 \\ 0 & \dots & \dots & 0 & -I & D & -I \\ 0 & \dots & \dots & \dots & 0 & -I & D \end{bmatrix} \tag{5.7}$$

The conditioning number of a matrix can be described as the ratio between the maximum and minimum eigenvalues. When the coefficient $1/\rho$ is not constant everywhere, it can not exit the matrix, making the eigenvalues to be more and more different so that the conditioning number may rise very quickly together with the density ratio.

1.2 Numerical Resolution

Among the different techniques for solving a Poisson problem, the natural choice falls on iterative solvers, where an initial guess of the solution is more and more approached to the exact one, and can be stopped when the approximation is good enough (the convergence of the algorithm depends on the characteristics of the matrix, which are satisfied for the given problem). Several algorithm belong to this family: the classical iterative methods: Jacobi, Gauss-Seidel, Krylov subspace methods, multigid.

Given the $n \times n$ linear system $Ax = b$, the Jacobi method is built by splitting the matrix into its diagonal and the remaining part $A = D + R$. So the system can be written as $(Dx - b = Rx)$; the approximations of x are calculated by taking the "left" x as new, and the "right" one as old. This can be written analytically as:

$$x_i^{(k+1)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j \neq i} a_{ij} x_i^{(k)}\right), \qquad i = i, n \tag{5.8}$$

In a similar way the Gauss-Seidel relaxation separates the matrix into the lower triangular, the diagonal and the upper triangular parts, so that the system is rewritten as $(L + D)x = b - U$. In practice, each time one point value is updated, it is immediately used to update its neighbour. The iteration step is given by

$$x_i^{(k+1)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j<i} a_{ij} x_i^{(k)} - \sum_{j>i} a_{ij} x_i^{(k+1)}\right), \qquad i = i, n \tag{5.9}$$

Eventually, the system can be expressed as a fixed point method, so that $Ax - b = 0$ becomes $Ax + Ix - b = Ix$, and can be solved with a Newton-Rapson type iteration

$$x_i^{(k+1)} = x_i^n + \frac{1}{a_{ii}}\left(b_i - \sum_{j=1,n} a_{ij} x_i^{(k)}\right), \qquad i = i, n \tag{5.10}$$

This method has been preferred in the first implementation of a solver adapted to the AMR, because it involves the explicit computation of the residual $r^{(k)} = b - Ax^{(k)}$, which is interesting because of the possibility to be written in a conservative form.

1.3 Boundary conditions

In the case of second order discretization with Dirichlet or Neumann boundary conditions, eliminated boundary conditions can be considered as a natural approach. This is particularly true if the conditions are homogeneous, where their implementation is straightforward; still with small modifications the method can be extended to imposed conditions. With the cell centered notation and the presence around each block of guardcells, the stencil does not change across physical boundaries. The imposition of the homogeneous condition is obtained into the definition of the matrix D itself: where a zero should appear on a certain boundary, the corresponding coefficient is imposed to zero; where a zero gradient condition must appear, together with the apparition of a zero, the central coefficient is reduced by one, as the two contribution are mutually elided. The imposed Dirichlet or Neumann boundary conditions are applied by a modification on the right hand side term.

2 Relaxation with AMR

The relaxation sweep is the first attempt to find an approximation of the Poisson equation exact solution. It is the base algorithm which must be able to work on the union of grid blocks of different mesh size. The relaxation operation consists in the application of a Newton method written in a "conservative" form (which means that the *elliptic matching* condition, described further in this chapter, is satisfied on coarse-fine interfaces).

2.1 Block based Newton iteration

The equation (5.5) can be rewritten as

$$\left[\frac{\partial}{\partial x}\left(\frac{1}{\rho}\frac{\partial \phi}{\partial x}\right) + \frac{\partial}{\partial y}\left(\frac{1}{\rho}\frac{\partial \phi}{\partial y}\right)\right]_{i,j} = \frac{1}{\Delta x}\left(\frac{F_x^+ - F_x^-}{\Delta x}\right) + \frac{1}{\Delta y}\left(\frac{F_y^+ - F_y^-}{\Delta y}\right) \tag{5.11}$$

with the "flux" terms (the first derivatives) defined as

$$\begin{aligned} F_x^+ &= \rho_{i+\frac{1}{2},j}\left(\frac{\phi_{i+1,j} - \phi_{i,j}}{\Delta x}\right) \quad ; \quad F_x^- = \rho_{i-\frac{1}{2},j}\left(\frac{\phi_{i,j} - \phi_{i-1,j}}{\Delta x}\right) \\ F_y^+ &= \rho_{i,j+\frac{1}{2}}\left(\frac{\phi_{i,j+1} - \phi_{i,j}}{\Delta y}\right) \quad ; \quad F_y^- = \rho_{i,j-\frac{1}{2}}\left(\frac{\phi_{i,j} - \phi_{i,j-1}}{\Delta y}\right) \end{aligned} \tag{5.12}$$

In the Newton iteration the left hand side of equation (5.11) is explicitly computed inside the residual. This quantity is evaluated inside each block, boundary cells included, regardless of the position of the block itself. A relaxation sweep consists in the application of the step (5.10) for each grid point, inside a loop over all the blocks after a synchronization step, as shown in the algorithm 3 .

An improvement over this kind of algorithm is a red-black treatment of grid points, as is usually done for the Gauss-Seidel type iteration. Another interesting aspects of the block based treatment arise: results show that updating one block and reintroducing immediately its data inside the loop improves greatly the convergence of the solver: it is something like a Gauss Seidel use of blocks instead of single grid points. Obviously this is a behaviour which is slowly lose

Algorithm 3 Relaxation routine pseudocode.

Procedure RELAX
while $(iter < iter_{max})$ **do**
  Synchronization point
  for $(l = 1, nbl)$ **do**
    $x(l) \leftarrow Newton\,(A(l), f(l), x(l))$
  end for
  $r(:) = f(:) - A(:)x(:)$
  $iter = iter + 1$
end while

some of its effectiveness when using parallel computations (while RB-GS can be extended to parallel without any problem, the standard GS cannot).

The guardcell prolongation interpolation strategy is the already described second order quadratic interpolation scheme, which has been tested to guarantee the second order convergence of the solver in an easy test: an analytic polynomial parabolic function ($f = ax^2+by^2+cx+dy+e$) which can be solved, with a second order discretization, to the machine precision. This behaviour has been verified on composite mesh by solving with increasing degree polynomial interpolation, obtaining the desired result (machine error) with the parabolic prolongation. The restriction does not play the same key role in the result accuracy (nor in the convergence speed), as the simple average is sufficient.

2.2 *Elliptic matching*

The special care which has to be taken in the coarse/fine interface has been investigated in Martin [61], Martin and Cartwright [62], in a similar way to the flux matching problem in the conservative advection equation. Since the way of communicating data between blocks is an exchange of Dirichlet boundary conditions, the operators working on each single block are not supposed to change near the boundaries: they can be applied inside a loop that covers all the concerned blocks, without taking care of the actual position or refinement level of the block itself. This way, the coarse and fine solution are not sufficiently linked, because the numerical fluxes, in this case the pressure gradient, evaluated at a coarse/fine interface are computed twice, once by the coarse mesh and once by the fine, and are not consistent. This causes a discontinuity in the first derivative which is $O(h_c)$, that acts like a singular charge proportional to the first derivative mismatch, and corrupts the solution preventing the attainment the fine grid precision or even convergence

$$\Delta\phi + C\delta\,(x_{cf}) \left(\frac{\partial \phi^{fine}}{\partial n} - \frac{\partial \phi^{crse}}{\partial n} \right) = f \tag{5.13}$$

In order to eliminate this artificial charge, one needs to enforce the continuity of both the variable and his first derivative over the interface, condition called *elliptic matching*. As for the advection equation, a correction on the fluxes is done exploiting the dedicated PARAMESH routines: the pressure gradient are stored on the two sides of the refinement jumps, then the integral of the coarser one, considered the less precise, is corrected by the computed difference between itself and the integral of the finer one. Actually, it is like a "counter-charge" on the right hand side with the same value as the initial mismatch. The standard Laplace operator in one dimension is

$$\left(\frac{\partial^2 \phi}{\partial x^2} \right)_i = \frac{1}{\Delta x^2} \left(\phi_{i+1} - 2\phi_i + \phi_{i-1} \right) = \frac{1}{\Delta x} \left(F_{i+\frac{1}{2}} - F_{i-\frac{1}{2}} \right) \tag{5.14}$$

with the fluxes F defined as

$$\begin{cases} F_{i+\frac{1}{2}} = \dfrac{1}{\Delta x}\left(\phi_{i+1} - \phi_i\right) \\ F_{i-\frac{1}{2}} = \dfrac{1}{\Delta x}\left(\phi_i - \phi_{i-1}\right) \end{cases} \tag{5.15}$$

The expressions for the two sides of the second derivative on a refinement jump as shown in

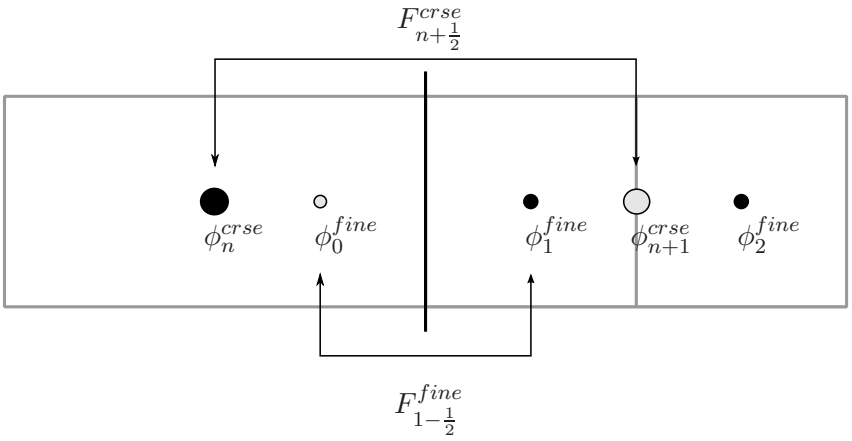


Figure 5.1: Computation of the flux on a refinement jump (in grey are the ghost points): the right flux of the last point of the left coarse block (indexed n) should match the left flux of the first point of the right finer block (indexed 1).

figure 5.1, considering that interpolated guard cells are used (grey points) and the block has n cells are

$$\begin{aligned} \left(\frac{\partial^2\phi}{\partial x^2}\right)^{crse}_n &= \frac{1}{\Delta x^{crse}}\left(F^{crse}_{n+\frac{1}{2}} - F^{crse}_{i-\frac{1}{2}}\right) \\ \left(\frac{\partial^2\phi}{\partial x^2}\right)^{fine}_1 &= \frac{1}{\Delta x^{fine}}\left(F^{fine}_{1+\frac{1}{2}} - F^{fine}_{1-\frac{1}{2}}\right) \end{aligned} \tag{5.16}$$

where $F^{crse}_{n+\frac{1}{2}}$ and $F^{fine}_{1-\frac{1}{2}}$ are defined on the same point, on the coarse-fine But the fine flux value must be imposed to the coarse grid to maintain the continuity of the $\frac{\partial\phi}{\partial x}$, so

$$\begin{aligned} \left(\frac{\partial^2\phi}{\partial x^2}\right)^{crse,match}_n &= \frac{1}{\Delta x^{crse}}\left(\frac{\Delta x^{crse}}{\Delta x^{fine}}F^{fine}_{1-\frac{1}{2}} - F^{crse}_{n-\frac{1}{2}}\right) \\ \left(\frac{\partial^2\phi}{\partial x^2}\right)^{fine}_1 &= \frac{1}{\Delta x^{fine}}\left(F^{fine}_{1+\frac{1}{2}} - F^{fine}_{1-\frac{1}{2}}\right) \end{aligned} \tag{5.17}$$

Finally the expression for the coarse grid can be written as the standard formulation plus a correction

$$\begin{aligned} \left(\frac{\partial^2\phi}{\partial x^2}\right)^{crse,match}_n &= \frac{1}{\Delta x^{crse}}\left(F^{crse}_{n+\frac{1}{2}} - F^{crse}_{n-\frac{1}{2}}\right) + \\ &\left(\frac{1}{\Delta x^{fine}}F^{fine}_{1-\frac{1}{2}} - \frac{1}{\Delta x^{crse}}F^{crse}_{n+\frac{1}{2}}\right) = \left(\frac{\partial^2\phi}{\partial x^2}\right)^{crse}_n + \delta F^{c/f} \end{aligned} \tag{5.18}$$

The effect of the *elliptic matching* can often be seen into a stagnation of the convergence; however, for some simpler problems the solver converges all the same. The solution, however, is always less accurate because of the artificial added charge on the right hand side.

2.3 Numerical tests

In order to check the resolution of the Poisson equation by the Newton iterative solver some numerical tests are performed. The ability of the solver to converge with a second order accuracy is fundamental. The chosen test function is the following:

$$\begin{cases} \Delta\phi(x,y) = -2\sigma e^C + \sigma^2 e^C \left((x-a)^2 + (y-b)^2\right), & \phi \in \Omega = [0,1]^2 \\ \phi(x,y) = \phi_{exact}(x,y), & \phi \in \partial\Omega \\ \phi_{exact}(x,y) = e^{C(x,y)} - e^{-\frac{\sigma}{2}R^2} \end{cases} \tag{5.19}$$

with $C(x,y) = -\frac{\sigma}{2}\left((x-a)^2 + (y-b)^2\right)$. It is an exponential peak located into a square unit domain: the coefficients $a = 0.5$, $b = 0.5$ set the maximum in the center of the domain; $\sigma = 500$ and $R = 0.5$ are shape parameters. The strong variation of second derivative is concentrated in the center of the domain, and so is the error. The boundary conditions are imposed Dirichlet: to the solution on the external cells the values of the exact solution are imposed. The error is defined in the two norms, a mean value and the absolute maximum. In order to deal with the different cell sizes, the mean error is not the standard mean over all the grid points, but instead an integration over the cell volume normalized by the domain volume:

$$\|\text{err}\|_1 = \frac{1}{\Omega}\sum_i^{nx}\sum_j^{ny} |\phi - \phi_{ex}|_{i,j} \cdot \Delta x_{ij}\Delta y_{ij} \tag{5.20}$$

so that for an uniform domain it coincides with the standard formulation. The second value is simply the maximal absolute value of error. In a particular case the error computation has been restricted to the active part of each single refinement level, in order to test the convergence rate for al the levels independently. In all the test cases the convergence limit is fixed to $\frac{\|residual\|}{\|rhs\|} < 1.e-14$.

Effect of the refined mesh

In this section a mesh convergence test is performed. A set of computations is performed on increasingly finer uniform meshes; then, the same computations are repeated by adding a local refinement on the domain center, where the most of the error is concentrated. The results should show a decreasing of the error each time an AMR level is added. The results in table 5.1 show

Grid	err_1	err_∞	q_∞
32^2	7.65e-4	5.07-2	-
64^2	1.82e-4	1.46e-2	1.79
128^2	4.50e-5	3.77e-3	1.95
256^2	1.12e-5	9.51e-4	1.99
32^2+1	7.34e-4	1,46e-2	1.79
32^2+2	1.83e-4	5.90e-3	1.31
128^2+1	1.12e-5	9.51e-4	2.63

Table 5.1: Mesh convergence test for different grid configurations.

the benefits of adding a new level of refinement. Clearly, adding only one level does not change the accuracy of the solution. When adding two levels, in the comparison with the corresponding

uniform fine mesh, some error in term of mean value is added. It is interesting, however, to see how the maximum value of error for this three level computation ($32^2 + 2$ in the table, for an equivalent fine resolution of 128^2) is still under the value of the immediately coarser resolution of 64^2. Solution and error are shown in figure 5.2 and 5.3; the accuracy of a refined zone is always improved by an added finer level.

Convergence rate

The aim of this tests is to check the convergence rate of the solver working on an adaptive mesh. The test function is, as before, (5.19). The mesh is refined on the center of the domain. The configuration of the mesh is fixed, the number and position of blocks not changing from one run to another. Instead, the block cell number is progressively increased, four, eight and sixteen, in order to increase the accuracy on all the levels at the same time. The solver should keep a quadratic order of convergence independently of the level, as the *elliptic matching* condition should guarantee the connexion between different grid sizes. In table 5.2 the measured maximum and mean error are presented: the measured convergence rates for the first error norm are clearly of two for all the meshes. Only a negligible decrease can be observed for the maximum error, where the value of q is nearer to 1.9 for the coarser levels. In figure 5.4 the previous results are

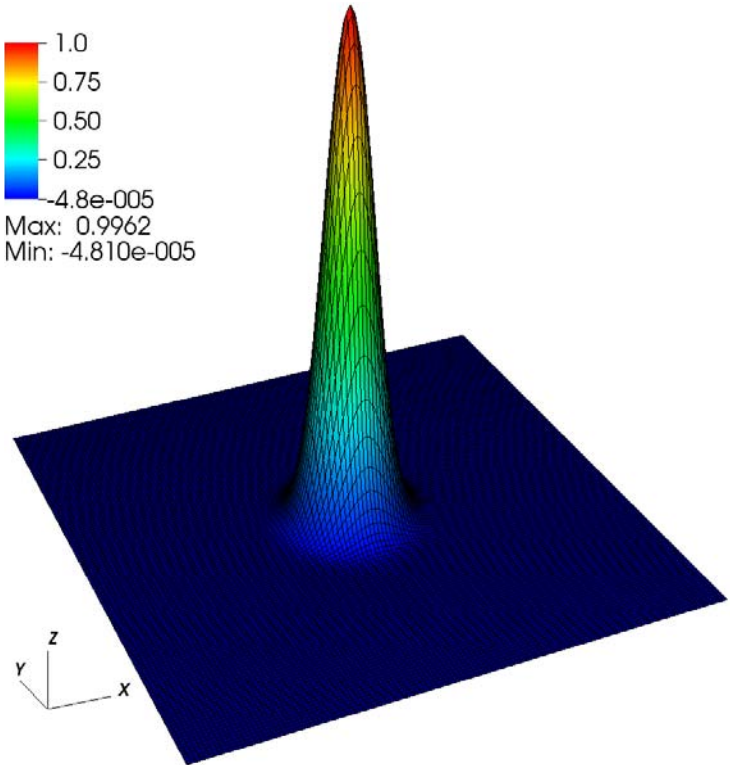
Grid	Level	err_1	err_∞	q_1	q_∞
64	1	5.33e-03	1.29e-02	-	-
	2	1.75e-02	2.60e-02	-	-
	3	2.53e-02	4.03e-02	-	-
	global	8.86e-03	4.03e-02	-	-
128	1	1.32e-03	3.70e-03	2.02	1.80
	2	4.62e-03	7.08e-03	1.92	1.87
	3	6.69e-03	1.06e-02	1.92	1.93
	global	2.27e-03	1.06e-02	1.96	1.93
256	1	3.27e-04	9.93e-04	2.01	1.90
	2	1.19e-03	1.86e-03	1.96	1.93
	3	1.72e-03	2.70e-03	1.96	1.97
	global	5.77e-04	2.70e-03	1.98	1.97

Table 5.2: Mesh convergence test for different grid configurations.

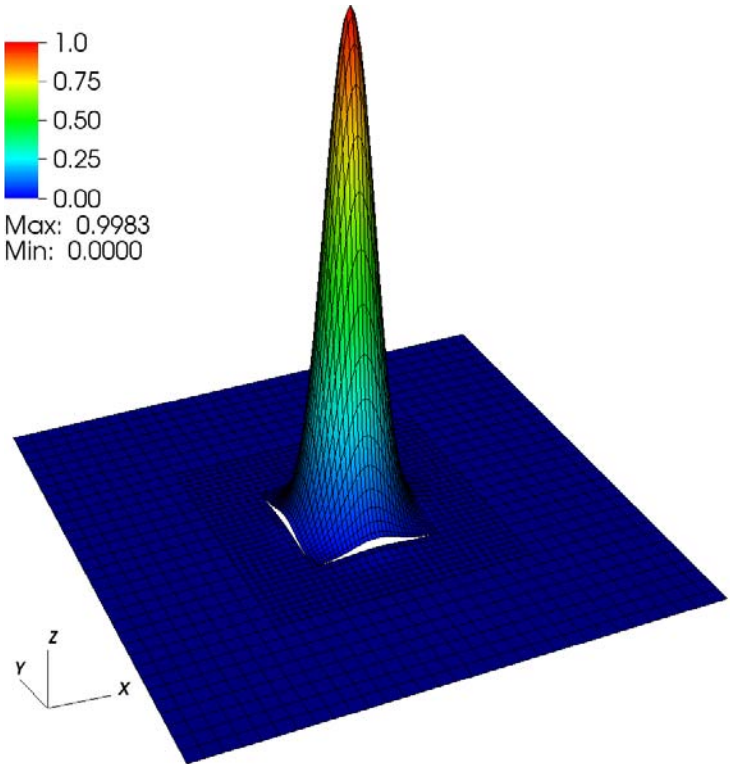
shown, and the second order behaviour can be observed.

Elliptic matching

The following test tries to put in evidence the importance of the *elliptic match* condition by confronting a computation with and one without. The behaviour of the resolution without the condition imposed is not really predictable. For the simple case presented here the convergence of the residual to its prescribed tolerance is not prevented; however the final error is quite greater than where the condition is respected. The error in figure 5.5 logically seems to grow at the refinement jumps; this in turn makes the error on the finest levels boundary to be greater, thus affecting the overall precision of the solution. In table 5.3 the two computed errors are compared. Clearly the mean error without correction is much more important than the one

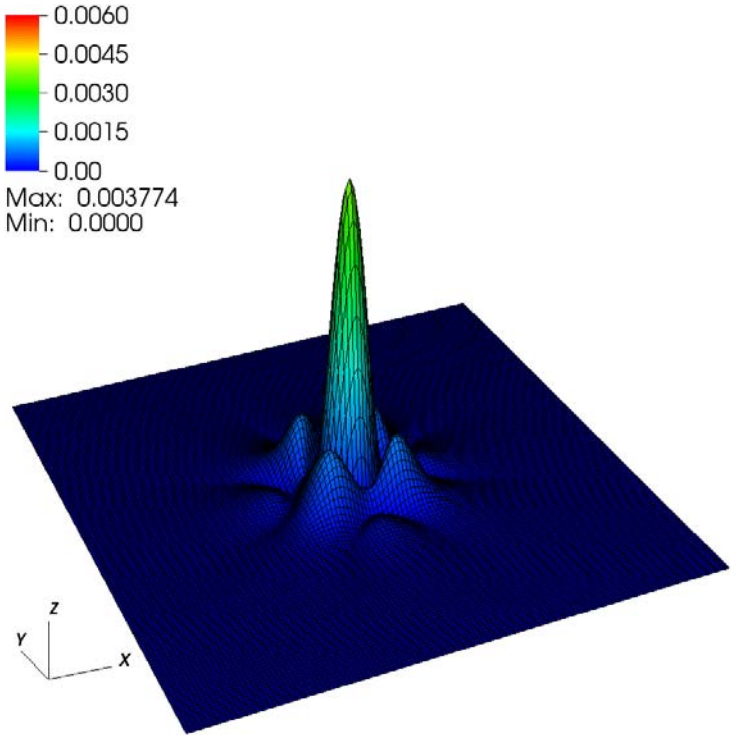


(a)

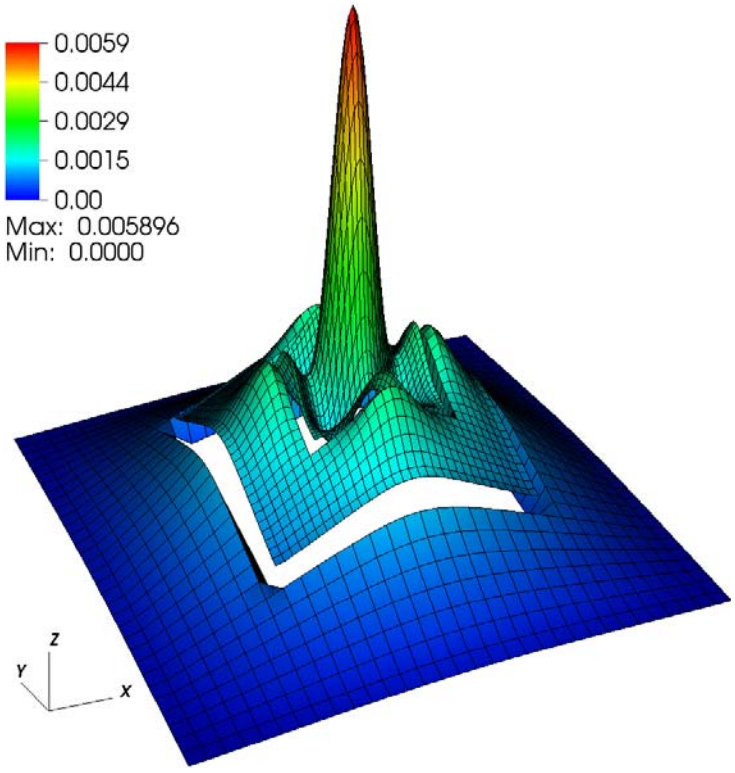


(b)

Figure 5.2: Comparison of the solution for the uniform and adaptive meshes. (a) Uniform solution, 128^2 (b) AMR solution, three levels, 32+2.

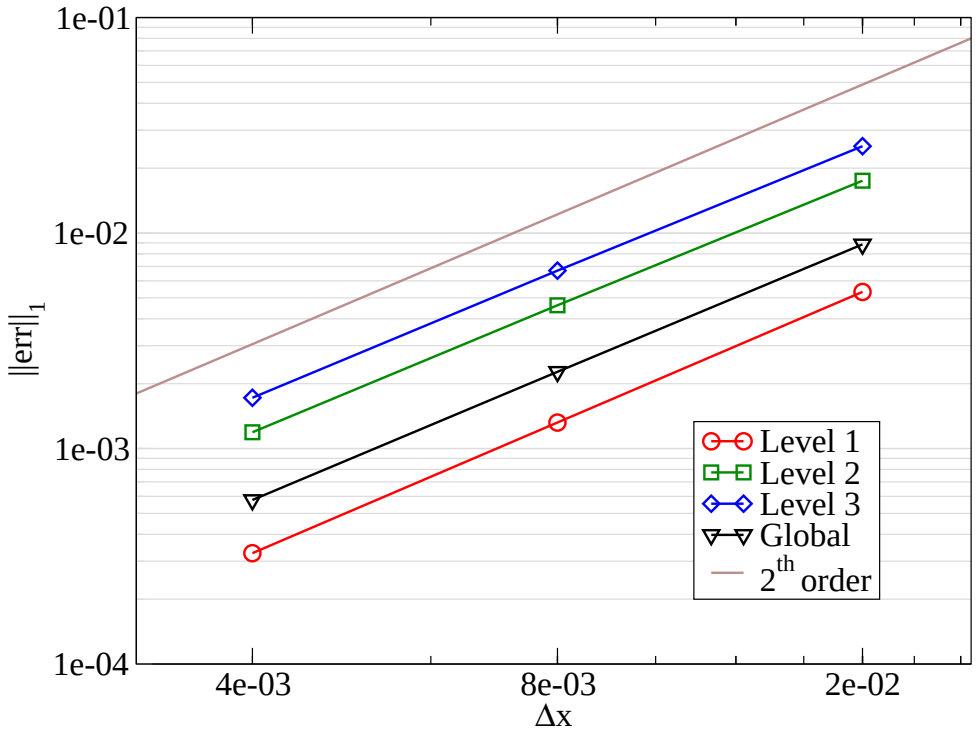


(a)

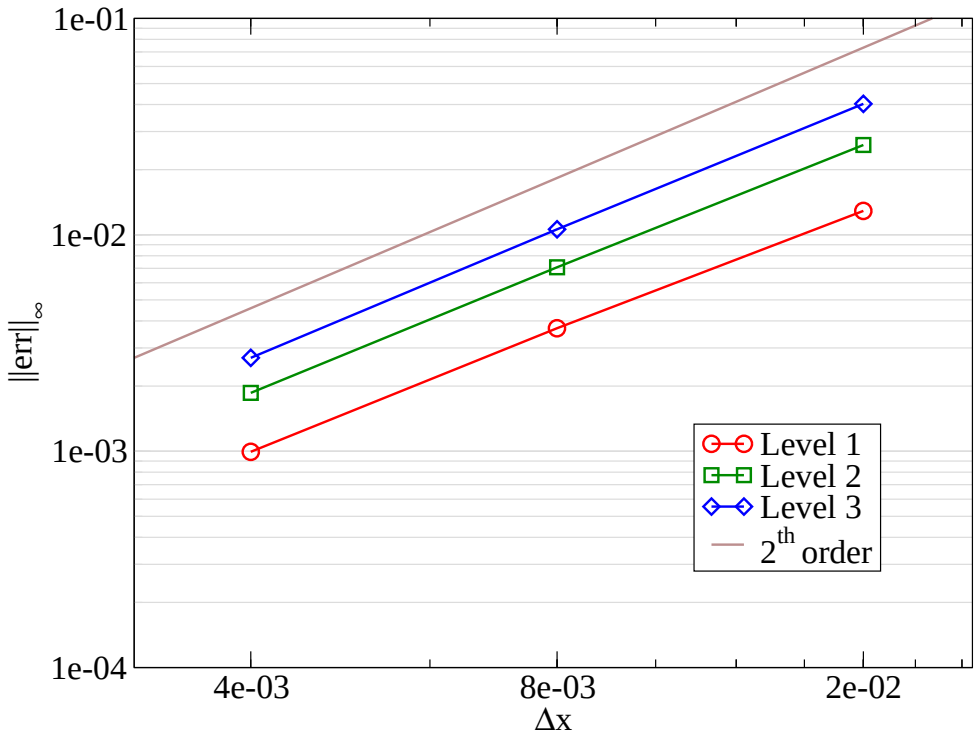


(b)

Figure 5.3: Comparison of the solution for the uniform and adaptive meshes. (a) Uniform error, 128^2 (b) AMR error, three levels, 32+2.

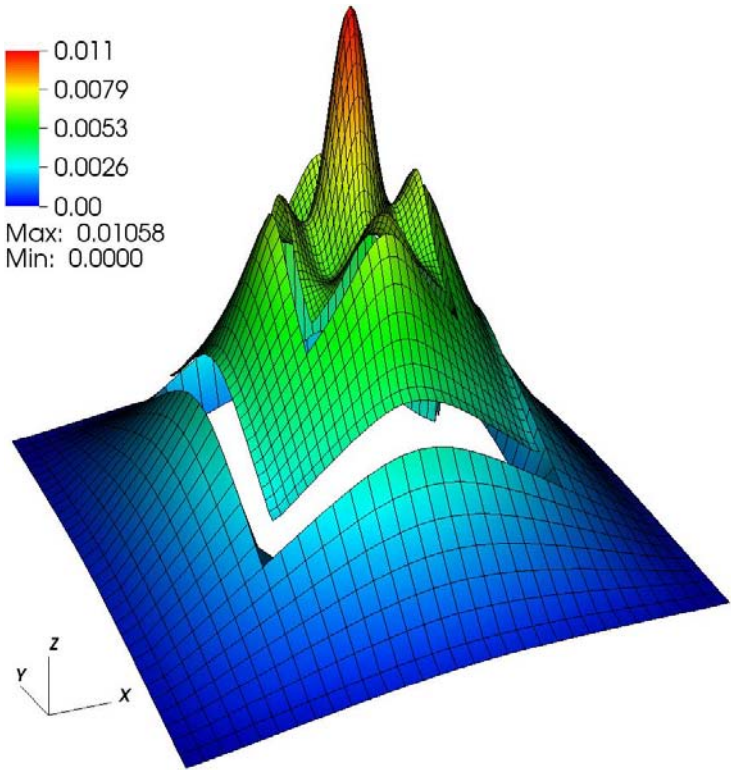


(a)

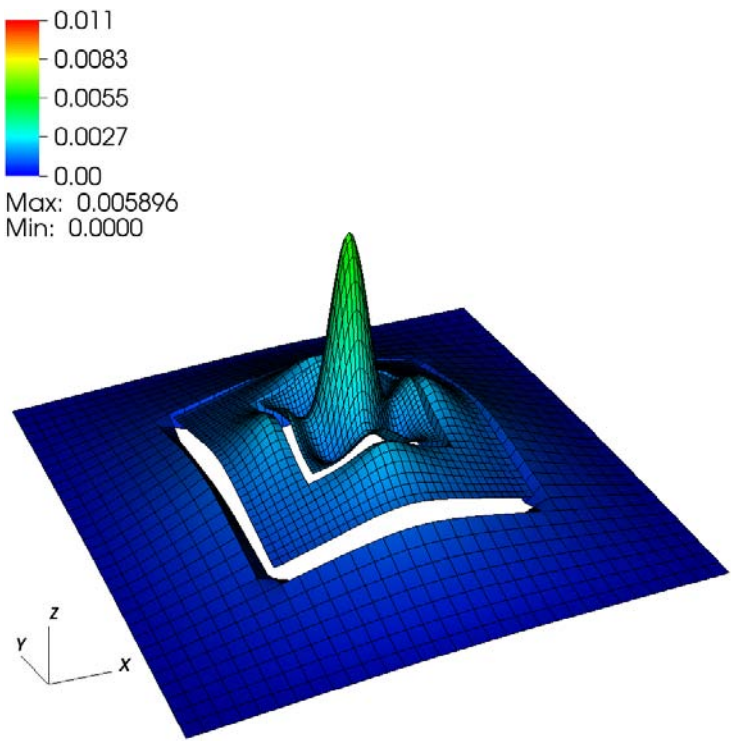


(b)

Figure 5.4: Error convergence for the exponential peak test. (a) Mean error on each AMR level and global (b) Maximum error on each AMR level.



(a)



(b)

Figure 5.5: Comparison of the error with three AMR levels, with and without coarse-fine correction. (a) Error without *elliptic matching* (b) Error with *elliptic matching*.

with the correction, being almost four times greater. There is less difference between the two peak errors, still the correction reduces of almost two times the value.

	Levels	err_1	err_∞
No match	1	1.32e-3	3.70e-3
	2	4.32e-3	7.08e-3
	3	6.69e-3	1.06e-2
	global	2.27e-3	1.06e-2
Match	1	4.32e-4	1.24e-3
	2	1.49e-3	2.20e-3
	3	2.09e-3	5.89e-3
	global	7.34e-4	5.89e-3

Table 5.3: Error in the AMR computation, with and without coarse-fine correction.

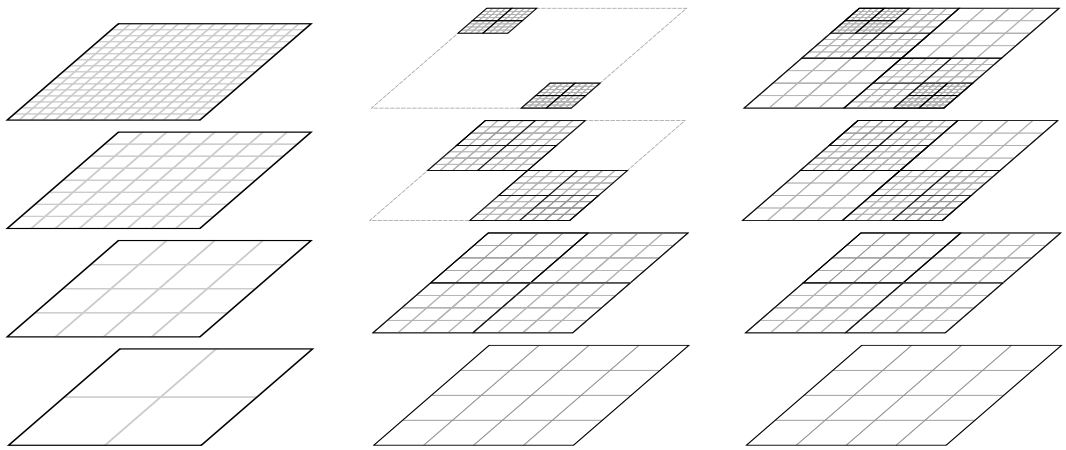
3 Multigrid with AMR

In the single phase solver a geometrical multigrid strategy (Trottenberg et al [122]) has been employed to inverse the associated linear system's matrix, due to its optimal convergence rate (up to an *O(N)* number of operations, with N representing the number of unknowns).

The main idea of the multigrid method is that what can be considered as a high wave number depends actually on the mesh size. A low wave number error on a fine mesh would be a high wave number error on a coarser one. Given the characteristic of the relaxation method to be very effective in dampening the higher frequencies of the error, but slow to reduce the lower ones, the consequence is that if the error is smoothed on different mesh sizes at the same iteration, all the wave length will be damped. In practice, if a set of increasingly coarser meshes are defined, a first approximation is given for the finest mesh; then, a restriction operator brings the residual to the coarser mesh, where it becomes the right hand side of a residual-correction form equation. A correction approximation is computed to this level, and the procedure continues recursively until the coarsest level is reached. Then the computed correction is prolonged to the finer mesh, where another relaxation sweep is applied (the correction step alone is not convergent), and so on until the finest mesh. This is the base V cycle multigrid; other "paths" between levels may define variation of the algorithm, such as the Full or the W type multigrid.

However, performing multigrid with adaptive mesh refinement requires some modifications on the "regular" algorithm, as the idea of the grid hierarchy has to be redefined. In this context, the mesh is not evolving during the Poisson solver iterations, as the grid is allowed to evolve at the end of each timestep only: it would be more precise to say that it consists of a fixed composite grid. There are, on the other hand, examples of *evolving* AMR grids during the iterative solve, as in Brown and Lowe [13]. In the same work and in Trottenberg et al [122], two main approaches are distinguished. In the first, called MLAT (for Multi Level Adaptive Technique, Bai and Brandt [6]), the grids in the multigrid hierarchy coincide with the AMR resolution levels, so that each level consists of an uniform mesh or, more precisely, of an union of grids with same mesh spacing. This method is well suited for the patch based AMR, where each patch can be solved as an independent grid (with multigrid, for example, as done by Sussman [111]), and offers the advantage of a symmetrical associated system matrix. The second method is known as FAC (Fast Adaptive Composite grid solver, McCormick [65]). In this algorithm the relaxation sweeps are performed across the entire computational domain, and one defines the

succession of multigrid levels by restricting the highest resolution sub-grids to the next lower resolution, building at each level a composite grid, that is defined as an union of blocks not all at the same refinement level. In this configuration more work is done in the relaxation sweep, but a stricter coupling between levels can be maintained. The recursive form of its pseudo code is presented in algorithm 4. If the two are told to work well, the latter has been chosen for



(a) (b) (c)

Figure 5.6: Representation of different types of multigrid: (a) Classical multigrid (b) MLAT (c) FAC.

some reasons. The first is for parallel efficiency purpose: in the PARAMESH framework the repartition of the blocks among the processors is done for the leaf blocks first, because the most of the time is spent here. So, if a relaxation sweep is performed on a fixed level of refinement, it will likely make only some processors work. Instead, if it is applied on the entire domain at each level, it can benefit from a more efficiently parallelized relaxation. The second reason is that PARAMESH multigrid routines are meant to work in a global sense, so that it would not be possible to perform "sub-multigrid" relaxation over a few blocks only, like in a patch based multigrid (Sussman [111]). Moreover, it is more straightforward to treat boundary conditions over the coarse/fine interface at each multigrid level, making the prolongation and restriction operations easier. In order to build the composite meshes, following the example of an unofficial version of multigrid for PARAMESH written by Kevin Olson, some "ghost blocks" are created. They are "clones" of the leaf blocks not at maximum refinement level, and are placed side by side to the finer blocks, so that together they cover the entire computational domain. In figure 5.6 the different use of grid blocks are showed in comparison with the standard meshes and the MLAT algorithm; the "ghost blocks" are, for the FAC algorithm, the blocks which do not have any children, and so they are copied to the finer levels to store the unknowns and the residuals.

In a two grid cycle with AMR the relaxation is performed on all the leaf blocks, and the residuals computed. Then, the residuals are restricted from the finest grid to the underlying coarse one; all the other leaf blocks copy their residuals to their underlying "ghost", completing the right hand side of the lower multigrid level. Then relaxation is performed on this level, and the correction is passed to the finer level by prolongation, where a finer grid exists, and by simple copy elsewhere. It has to be kept in mind that, here, the term level does not indicate a refinement level, but the maximum level contained in that composite grid.

In a full V-cycle, represented in the algorithm 4, the operation continues until reaching the coarsest grid, where a more precise correction is computed by a Conjugate Gradient (at a certain level, the mesh becomes always uniform). The same user made routine used for AMR

prolongation is used here. The stencil and coefficients do not change at the block boundaries because, previous to the prolongation operation, all the blocks receive guardcell data. In Martin and Cartwright [62], quadratic interpolation is found to be the minimum necessary to maintain second order accuracy (this holds for the guardcell filling operation, but not necessarily for the prolongation of the corrections). First order restriction is used for the grid : even if in the relaxation procedure using a second order interpolation seems to slightly increase the accuracy in some cases, as in the final algorithm the multigrid is used as preconditioner more than a solver, there will be no noticeable improvement. So the first order is kept, with the advantage of its intrinsic symmetry and a slightly smaller computational times. The Full multigrid Cycle

Algorithm 4 FAC multigrid algorithm for $Ax = f$, recursive formulation with V-cycle.

Procedure MG
$r_{lmax} = f - A_{lmax} x_0$
while $(r_{lmax} > toll)$ **do**
  MGLEVEL*(lmax)*
  $x = e_{lmax}$
  $r_{lmax} = f - A_{lmax} x$
end while

Procedure MGLEVEL(l)
if $(l > 0)$ **then**
  if $(l = l_{max})$ **then**
    $e_{lmax} = x$
  else
    $e_l = 0$
  end if
  $e_l \leftarrow RELAX(A_l, f_l)$
  $r_l = f_l - A_l e_l$
  $f_{l-1} \leftarrow Restrict\,(r_l)$
  MGLEVEL($l - 1$)
  $\delta e_l \leftarrow Prolong\,(e_{l-1})$
  $e_l = e_l + \delta e_l$
  $e_l \leftarrow RELAX(A_l, f_l)$
else
  $e_l = 0$
  $e_0 \leftarrow RELAX(A_0, f_0)$
end if

is another possible choice. In this case the initial condition at each iteration is restricted down to the coarsest level. Then, a rising sweep is performed; here a first full two levels V cycle takes place. Another rising sweep is done, and a second three levels V cycle is performed. This procedure is repeated until the finest level has been processed.

Both the V-cycle and Full multigrid cycles have been tested: even if in terms of iterations the FMG is faster to converge, the V-cycles has proved to be more effective in terms of computational time. Concerning the number of relaxation sweeps, there are no theoretical basis to predict the optimum value. After some trial-and-error tests, as it can be seen in the next section, a number of sweeps constant or increasing toward the coarser grids seems to give the best convergence speed when few levels of multigrid are available; only two or four sweeps are sufficient when

using a greater number of levels. Quite surprisingly, in the two phase solver the behaviour is the opposite: repeated smoothing iterations on the finest grid gives the better speed up characteristics.

3.1 Numerical test

In this section the attention is brought to the convergence speed instead of the final error of the approximate solution. Some multigrid runs with different relaxation patterns are compared. In particular, results from the two cycles, "V" and "FMG", are presented. The problem is solved on both an uniform mesh (128^2) and an adaptive one with the same finest resolution. Apart from the gain in speed due to the reduction of grid points, results reveal the behaviour of the multigrid not to be the same for the two configurations. The interest is to find the best balance between the number of iterations of the solver and the time spent by relaxation at each iteration. For all the runs, the accuracy of the solution is fixed to $\frac{\|residual\|}{\|rhs\|} < 1.e-10$. The computations are performed on a single processor.

In table 5.4 and figure 5.7 some results are presented. The two different multigrid cycles are compared with relaxation only; the relaxation patterns are the number of sweeps performed at each level. Each sweep is a red-black treatment of all the level nodes. The "p" stands for proportional, which means a number of sweeps proportional to the depth of the multigrid level: the relaxation sweeps are increased towards the coarser levels. In one case, the V cycle with only one relaxation sweep per level, the solver does not attain convergence (n/c in the table).

In the uniform mesh, the FMG seems for this problem to be the fastest method (6.9 seconds being the most favourable time measured), even though the V cycle regains quite a lot with the proportional relaxation pattern. With the adaptive mesh the iterations needed to achieve convergence are less than for the previous case. This is probably due to the natural speed-up given by the less refined levels, where the informations can travel faster. Together with the reduction of grid points, from 16384 to 2560 for the finest level, this acceleration brings the total computational time down to approximatively 3 seconds, less than half of the previous best time. Quite surprisingly, now the V cycle gains an edge, being able to reduce by a quarter the time spent by the FMG.

From this table, the most retained pattern is the proportional V cycle. It is not, anyway, a definitive choice, as the behaviour of the solver can be different depending on the problem to be solved.

4 High density ratio problems

In a preliminary development, the multigrid solver has been tested on the two phase Poisson equation. However, for problems subject to high density ratio it usually fails to converge, or needs too much time to solve the system. As found by M. Sussman [57], a density ratio of about 1:10 seems to be the uppermost limit. Some variants of the standard multigrid seem to be able to overcome this problem: in particular the *semicoarsening* method (Schaffer [101]), in which the coarsening of the grid is performed differently in the two (or three) directions. Another method is the *Algebraic multigrid* of Wagner [129]: it was developed as a more robust version of a standard multigrid, able to work with strong discontinuous and anisotropic coefficients. Differently from the geometrical multigrid, it does not set the coarser grids by a fixed coarsening strategy; instead, it dynamically makes the best choice of coarse point in order to reduce efficiently the error. For strong anisotropy problems, it can be seen as a sort of "dynamical semicoarsening". However, this means that, every call to the solver, some important time must be spent in the restriction and prolongation operators setup. In the case of repeated calls to the solver with the same matrix coefficients this would not be a problem; the interface motion, however, makes the matrix values

		UNIFORM		AMR	
	Pattern	Iterations	Time [s]	Iterations	Time [s]
RELAX	1	31363	586	14191	112
FMG	1	50	12.53	32	6.77
	2	13	5.62	14	4.79
	4	10	7.39	9	5.41
	2p	15	15.76	11	6.30
	4p	9	16.32	11	10.46
V	1	n/c	n/c	n/c	n/c
	2	54	11.7	14	1.53
	4	24	9.15	22	1.29
	2p	20	8.64	9	2.17
	4p	11	8.48	9	3.14

Table 5.4: Iterations and time to convergence for different relaxation schemes, uniform and adaptive meshes.

change at every time step, so that an AMG algorithm should continuously perform the set-up. On the contrary, in geometrical multigrid the coarser grids and the interpolation procedures are fixed and always work in the same way. For this reason the AMG was not found attractive for this particular problem.

4.1 The preconditioned Bi-CGMStab solver

Unluckily, the discretization of the Laplacian on the composite grid leads to an unsymmetrical matrix, as the interpolation coefficients appear, preventing the use of Conjugate Gradient Method. To overcome this problem, one of the most reliable unsymmetrical linear system solvers of the Krylov family, the Bi-Conjugate Gradient Stabilized Method (Bi-CGMStab, algorithm 5) of VanDerVorst [126] has been employed. Its most interesting features are the quite regular convergence, no need to build the global matrix transpose (as for the Conjugate Gradient Squared), and the simplicity of the preconditioning. As for the PCGM (Trottenberg et al [122]), there is no need to multiply explicitly the system by the preconditioning matrix M, but to compute the effect of applying the matrix M^{-1} to a vector. On the other side, within the Bi-CGMStab algorithm this operation is performed twice, so that the preconditioner is called two times per iteration. Finally, the multigrid FAC algorithm can be successfully applied as preconditioner, as CG does with standard multigrid. The preconditioning is performed by an approximate resolution of the $M\tilde{s} = s$ and $M\tilde{p} = p$ elliptic equations. There is no arresting criterion based on the residuals. Preconditioning with different relaxation patterns have been extensively tested. The best performances are given by a simple single V cycle at each call, with a number of relaxation increasing towards the finest levels; in the case of many levels of refinement (and consequently many multigrid levels), a constant number seems to work as well as the proportional.

Again, some attention has to be paid to the coarse/fine interface. The procedure is different from the *Jacobi* iterations *flux-match* correction: there is no explicit flux computation in the Bi-CGMStab. As suggested in Teigland and Eliassen [118], the elliptic matching condition can be imposed by a Dirichlet and Neumann boundary conditions coupling. In the BiCGstab algorithm, the communication between blocks happens into the matrix-vector products. Across the coarse-fine interface, a preliminary Dirichlet conditions exchange between blocks is allowed;

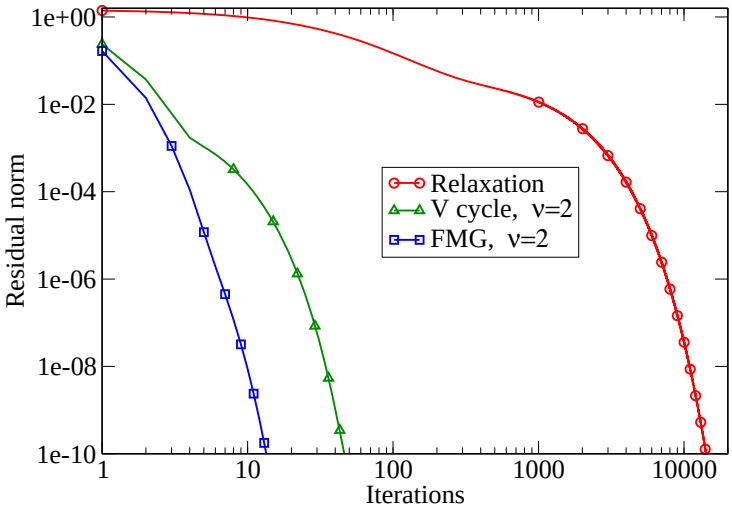


Figure 5.7: Comparison of the convergence histories for relaxation alone and the two multigrid cycles.

then the coarse grid ghost cells values are subjected to a correction to make the first derivative match the fine grid value, so that the coarse grid actually receives imposed Neumann boundary values. The numerical tests are not repeated as for the single phase solver, as the effects of the presence or absence of the corrections are the same as before; in particular, when increasing the coefficient variation, the convergence tends to stagnate, resulting in a poorly accurate solution. The smoothing routine has been changed, being now a *Red-Black Ordered Gauss Seidel* [1], that shows better smoothing properties.

The main parallel algorithm inefficiency source lies in the communication. At every relaxation step, as well as at every Bi-CGMStab matrix-vector product, guardcell data have to be exchanged. In a block-based AMR, this is the most part of the additional work imposed by the adaptive mesh because boundary conditions are needed at each block edge (and one would like to have more smaller blocks, so that he can easier isolate refined regions and reduce the total number of points). In particular, in strong density ratio problems the benefit of an increased number of relaxation sweeps would raise excessively the calls to the guardcell filling routine. A number of relaxation sweeps proportional to the multigrid level increasing towards the the finer levels seems to reduce the total number of iterations as well as the time spent on the coarser levels, where the parallelization in less effective.

The other problem lies into the Gauss-Seidel-like use of blocks. If the reintroduction of a freshly updated block into the loop can strongly speed-up the convergence, in parallel the synchronization happens only once finished all the block loop, thus reducing the acceleration. Always when increasing the parallel repartition the number of iterations grows a little bit.

4.2 Numerical tests: convergence

In this section the capability of the code to solve problems with non constant coefficients is verified. The aim of this test is to demonstrate the limits of the multigrid algorithm alone in the two phase solving. Still, even if no longer convergent, it remains the decisive speed up factor for the more robust BiCGstab.

[1] As the relaxation is used in the preconditioner only, there is no need to impose the elliptic matching condition, the final accuracy is given by the BiCGstab algorithm; neither in the convergence speed any enhancement was reported.

Algorithm 5 The preconditioned Bi-CGstab algorithm for the $Ax = b$ linear system. The matrix M is the preconditioning matrix. $M = 1$ gives the unpreconditioned algorithm.

Given $x_0 = 0$,ϵ
$r_0 = b - Ax_0$
$\tilde{r} = b - Ax_0$
while $(\|r\|_2/\|b\|_2 > \epsilon)$ **do**
  $\rho_1 = r.\tilde{r}$
  if (First iteration) **then**
    $p = r$
  else
    $\beta = (\rho_1\alpha/\rho_2\omega)$
    $p = r + \beta(p - \omega v)$
  end if
  $M\tilde{p} = p$
  $v = A\tilde{p}$
  $\alpha = \rho_1/(\tilde{r}.v)$
  $s = r - \alpha v$
  if $(\|s\|_2/\|b\|_2 < \epsilon)$ **then**
    $x = x + \alpha\tilde{p} \Longrightarrow$ EXIT
  else
    $M\tilde{s} = s$
    $t = A\tilde{s}$
    $\omega = (t.s)/(t.T)$
    $x = x + \alpha\tilde{p} + \omega\tilde{s}$
    $r = s - \omega t$
    $\rho_2 = \rho_1$
  end if
end while

The simulation consists in the resolution of the pressure field inside a static bubble centered into a L^2 domain. The solution is a discontinuous constant pressure, with a jump located on the interface. The jump theoretical value is given by the two dimensional Laplace equation $p_{int} - p_{ext} = \sigma\kappa$, with κ is the interface curvature and σ the surface tension. There is no velocity field solving, the Poisson equation only is solved, the right hand side charged with the Ghost Fluid jump conditions. In the next chapter a full Navier-Stokes resolution of this configuration is presented. In order to put the solver in "real" working conditions, three levels of adaptive mesh are added on the interface location, for a finest resolution of 128^2. The parameters are

$$\begin{cases} \rho_{int} = 10^{\mathrm{x}} \text{ kg.m}^{-3} \\ \rho_{ext} = 1 \text{ kg.m}^{-3} \\ \sigma = 0.1 \text{ N.m}^{-1} \\ L = 4 \text{ cm} \\ R = 1 \text{ cm} \end{cases} \tag{5.21}$$

which give the predicted jump $[p] = 10$. The internal density is taken to be 10^{x}, with x= $[1, 2, 3]$.

In the first test the convergence history and computational times are compared for multigrid alone, BiCGstab alone and two different preconditioned BiCGstab, respectively with four Gauss-Seidel sweeps and a single multigrid V cycle. The problem size is fixed to a three AMR levels

which give the finest equivalent resolution of 128^2. The convergence limit is fixed to $\frac{\|residual\|}{\|rhs\|} < 1.e-12$.

As shown in figure 5.8, for the lowest density ratio all the algorithms reach convergence. At the same time, the number of iterations and the computational time are very different, as can be seen in table 5.5. Quite surprisingly, the worst result comes from the multigrid. Even if "tuned" with the best parameters, the large number of iterations results into a very large computational time. The BiCGstab algorithm behaves quite better, even if it still requires a lot of iterations to converge: the total time is about fifteen times smaller. A similar result is obtained from a lightly preconditioned version of the same algorithm: the iteration number drop dramatically, but the added relaxation time roughly balances the total time. The best performances are definitely given by the multigrid preconditioning of the BiCGstab, which drop to the minimum both the iterations and total time.

When the density ratio is brought to one hundred, the multigrid alone shows even poorer performances, not being able to converge in the time and iterations limits (the convergence is still not attained at 500 seconds). The BiCGstab shows its better robustness, even if the workload is about the double as before: in this care the effect of the preconditioning begin to manifest itself, as the preconditioned versions attain convergence in about the same time as before. In the worst scenario, where the density ratio reaches one thousand, this behaviour is even more pronounced; the strong effect of the multigrid preconditioning allows the resolution of the problem within an about 20% increased time, showing the predicted robustness and rapidity of the combined solver.

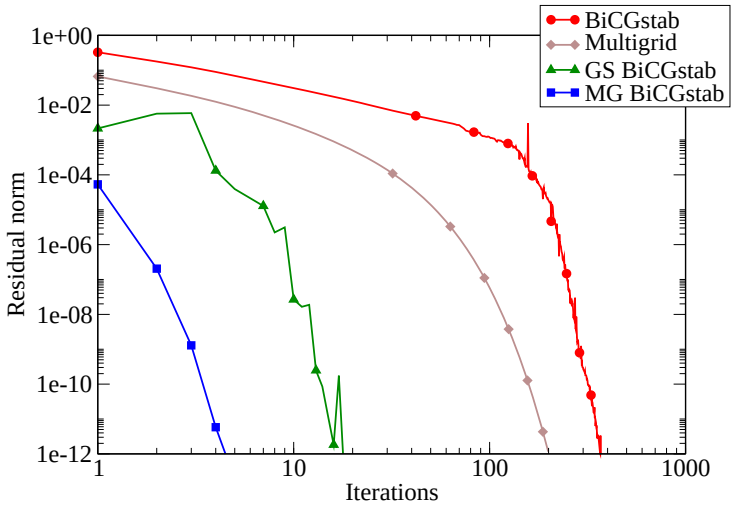
Density ratio	10		100		1000	
Algorithm	Iterations	Time	Iterations	Time	Iterations	Time
mg	201	101.39	n/c	n/c	n/c	n/c
bcg	365	6.44	872	15.78	3184	57.58
gsbcg	18	6.62	17	6.27	16	6.09
mgbcg	5	2.9	5	2.91	6	3.44

Table 5.5: Iterations and computational times for different algorithms solving the static bubble pressure field.

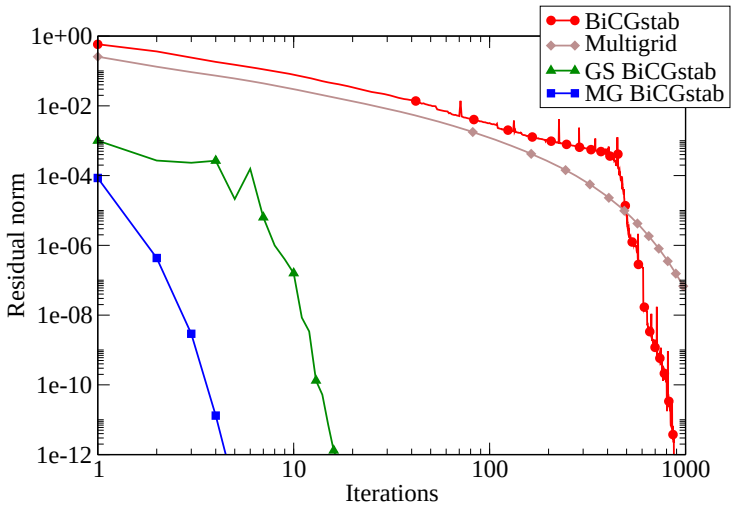
4.3 Numerical tests: accuracy

The next set of results, presented in table 5.6, is a mesh convergence test for the computed pressure field, each time adding an AMR level on the jump location. The value of q is evaluated with the global errors, given the changing number of levels. The density ratio is equal to 1000, so the solver employed is the full multigrid preconditioned BiCGstab. This is a good test for the application of the Ghost-Fluid jump conditions on a locally refined mesh.

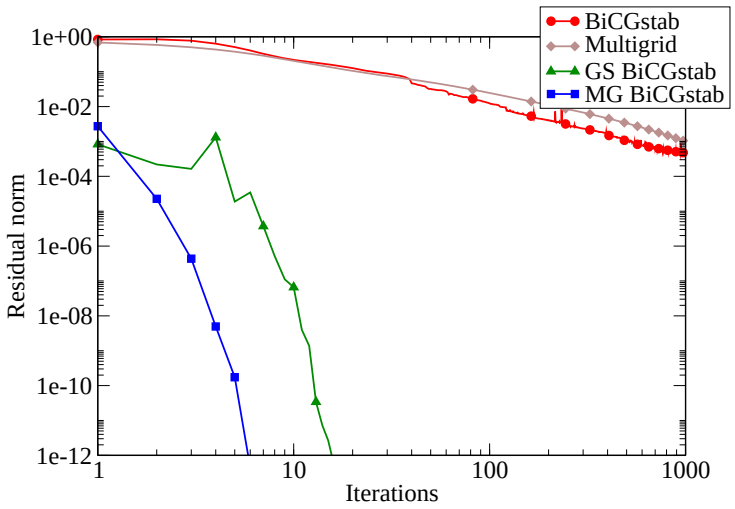
Results show a steady decrement in the error, with convergence rates near to two. The computational time increases from the 6.6 seconds for the coarsest mesh up to 11 seconds for the intermediate and finally 21.8 seconds for the finest mesh. The reduction of error is evident in the coarser meshes. The convergence rate for the finest meshes has not been computed because the reduction of surface covered by the finest grid decreases with the AMR levels; the maximum error, however, is located on the jump itself, and its convergence rate is the same as for the global mean error, showing the benefit of the refined mesh both on the interface and in the rest of the computational domain.



(a)

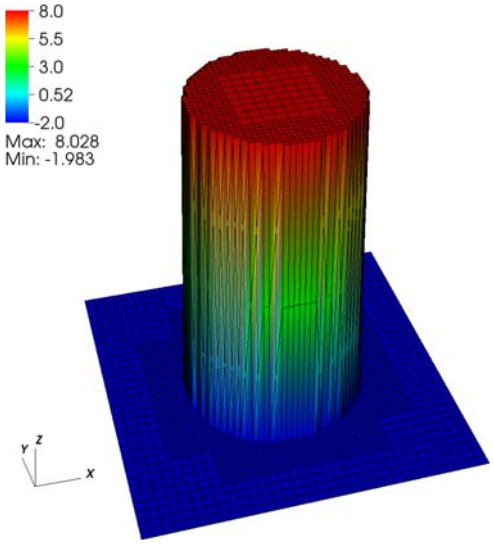


(b)

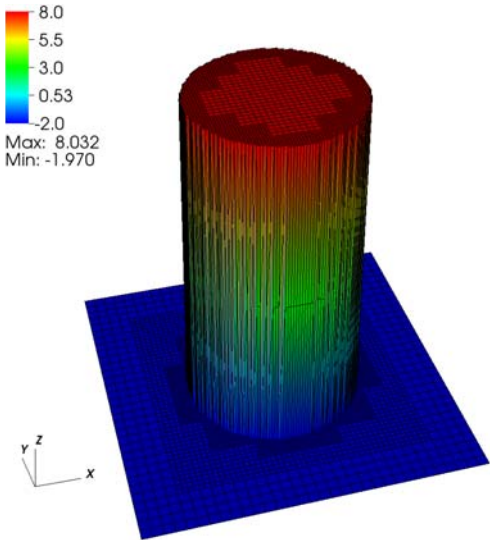


(c)

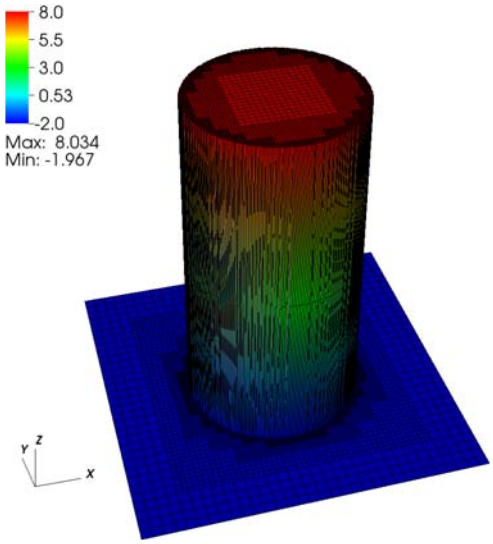
Figure 5.8: Convergence histories for different solvers. (a) $\rho_i/\rho_e = 10$ (b) $\rho_i/\rho_e = 100$ (c) $\rho_i/\rho_e = 1000$.



(a)



(b)



(c)

Figure 5.9: Computed pressure field for the static bubble test with increasing AMR levels. (a) 2 levels, 64^2 (b) 3 levels, 128^2 (c) 4 levels, 256^2.

	Level	err_1	q_1	err_∞	q_∞
64	1	1.14e-03	-	4.42e-03	-
	2	1.67e-03	-	1.04e-02	-
	global	1.37e-03	-	1.04e-02	-
128	1	1.81e-04	-	1.81e-04	-
	2	3.22e-04	-	1.05e-03	-
	3	5.04e-04	-	2.52e-03	-
	global	3.00e-04	2.19	2.52e-03	2.05
256	1	4.92e-05	-	4.93e-05	-
	2	8.47e-05	-	2.84e-04	-
	3	1.14e-04	-	4.45e-04	-
	4	1.52e-04	-	7.00e-04	-
	global	8.14e-05	1.88	7.00e-04	1.84

Table 5.6: Errors in the pressure solution for different meshes, $\rho_i/\rho_e = 1000$.

Conclusion

In this chapter the attention has been focused into the construction of a robust and efficient elliptic solver for the pressure equation. The two main issues were the utilization of the adaptive mesh, together with the difficulties bound to the high density ratios of some two phase simulations. The first approach is the construction of a Newton-Raphson iterative solver able to converge on the union of grid blocks, despite the presence of the refinement jumps. The *elliptic matching* condition is in this case imposed to link the coarse and fine solutions. This solver has been used to build a multigrid solver. In order to deal with high density ratio flows a BiCGstab solver has been used; the previously developed multigrid has been employed to accelerate its convergence speed.

Chapter 6

Validation

Contents

Introduction

This chapter is dedicated to the validation and the performances of the code. The presented configurations are mostly academical or simplified, and for many of them an analytical solution is known. If not, the parameters of the simulations are chosen in order to match well documented results from literature. This kind of simulations allows the test of the methodology and numerical methods, confirming the convergence of the solver and the accuracy of the results. In particular, the main hypothesis of this work is tested, whether the solution of the multi scale computations are as accurate as the equivalent finest resolutions. The main interest of the adaptative mesh is the gain in computational resources with negligible accuracy losses on the finest level.

The different tests try to cover all the physical phenomena expected by the model, so that converge for an arbitrary complex simulation can be expected. The tests are usually well documented in the literature. They are, for the linear advection equation: the Zalesak disk and the serpentine; for the single phase solver the Taylor-Green vortex translation and the double mixing layer; for the two phase flows, the static bubble, the damped surface wave (also known as Prosperetti's test), the Rayleigh-Taylor instability in both the linear and non linear evolutions, and finally the two dimensional rising bubble dynamics. Other tests, such as the lid driven cavity or the Poiseuille flow have been performed, but for the relative easiness of the first and

the difficulty of employing of the adaptative mesh in the second, the investigation have not been deepened and results are not presented here. All the tests but the rising bubble are presented in the reference work of Couderc [18], and the results are directly comparable. The bubble test is well documented in Sussman et al [114] and Hysing et al [44].

In the last section some AMR and parallel performances of the code are separately presented. All the computations in this chapter are performed on the same SGI ALTIX ICE 8200 architecture, Fortran compiler fce 11.1.059 and SGI MPT 1.26 libraries as for the previous chapter.

1 Linear advection equation

In this first section the performances of the code for the resolution of the hyperbolic linear advection are evaluated. This equation is naturally linked with the interface transport of the Level Set method, and presents some numerical challenges, in particular in the conservation of strong local gradients and mass. There are in literature some classical test cases, which are here re-proposed. In particular, the tests are focused on the interface local mesh refinement. As the properties of the Level Set are mostly important near the contour defining the interface, often the zero contour, one can expect good performances from a more accurate computation around this region. The hyperbolic equation is, moreover, quick to solve on the adaptative mesh, differently from the much more expensive iterative elliptic equation. The proposed tests are the Zalesak disk from Zalesak [132] and the serpentine proposed in Bell et al [8]. In both the mesh convergence test results are proposed for both the regular and the AMR code.

1.1 Zalesak disk

This simulation is the rotation of a rigid body consisting of a disk with a rectangular fence located inside. This is an interesting test because it put to the test the numerical scheme dissipation and diffusion, as it presents some very strong local gradients around the fence. The computational domain is a square $[0, 100]^2$; a disk of radius $r = 15$ is centered on the coordinates $(50, 75)$, the fence being a rectangle $[5, 25]$ located on the vertical diameter of the disk, on its lower part. The rotating velocity field is stationary, and defined as

$$\begin{cases} u\,(y) = (\pi/314)\,(50 - y) \\ v\,(x) = (\pi/314)\,(x - 50) \end{cases} \tag{6.1}$$

This velocity field makes the disk perform one complete rotation in $t = 628$ s, allowing the superimposition of the numerical solution with its initial state and thus a direct comparison with a subsequent error evaluation. The adaptive mesh simulation is obtained by refining the region around the disk (the contour only, if there is space enough the interior may be subject to de-refinement). In the AMR solution the accuracy of the solution of this shape is tested versus the same fine mesh resolution. The idea is to maintain the initial solution always inside the refined region as the simulation runs. The initial condition is restored when compared to the computed solution in order to avoid the cumulative errors due to the coarse-fine mesh jumps. In the AMR computation the referred resolution correspond to the finest level; the four tested meshes are composed of an increasing number of levels, the coarsest level being always a 16^2: two levels for the 32, three for the 64, four for the 128 and five for the 256, as shown in figure 6.1.

Advection on adaptive mesh

The error computation is a little bit tricky for this problem. In fact, the best method is to compare directly point-by-point the difference between the computed and initial conditions,

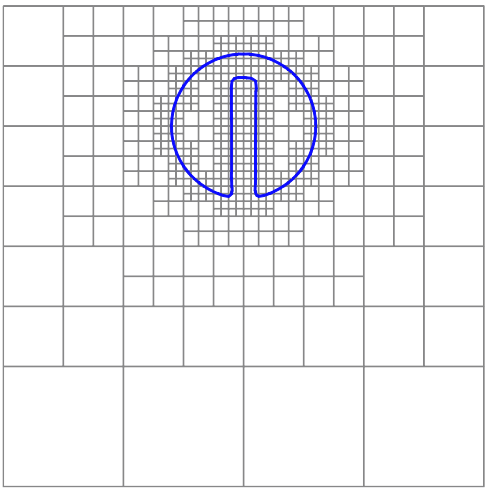


Figure 6.1: The adaptive mesh around the Zalesak disk, each square is a 4x4 block, for the equivalent finest mesh of 256^2. Solution after $t = 628$ s

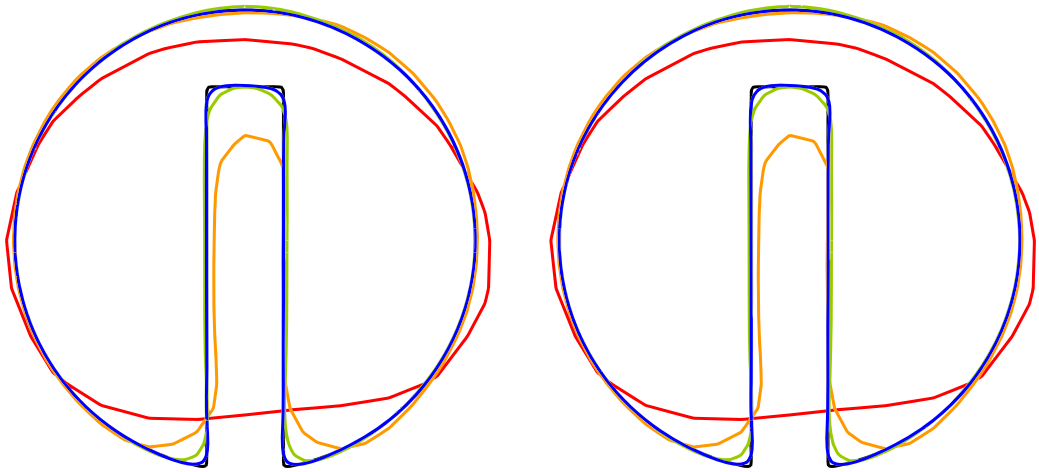
as a mass loss computation may be altered by compensations between positive and negative numerical variations. However, the effect of the refined mesh is only relevant in the proximity of the interface, where the computation is the most difficult because of the strong gradients related to the fence shape. If the global mean error is computed, it would be clearly affected by the coarser AMR levels. So, another mean should be performed over the points in the immediate proximity of the interface. Given the difficulty of finding the same set of points on both the adaptive and uniform meshes, the comparison is performed on the two first points on each side of the interface only, so that the evaluated quantity is in practice the error in the reconstruction of the zero contour. If the set of points may seem too restrictive, the graphical results can complete this measure, giving an overall idea of the accuracy of the resolution. So a "local error" and its convergence rate q can be defined as

$$\begin{aligned} \|\mathrm{err}\|_1 &= \sum_{\Omega} |\ \phi_{ij}^{ini} - \phi_{ij}^{comp}| \cdot \Delta x_{ij} \cdot \Delta y_{ij} \\ q &= \frac{\ln\left(\mathrm{err}_{2h}\right) - \ln\left(\mathrm{err}_{h}\right)}{\ln(2)} \end{aligned} \tag{6.2}$$

given ϕ_{ij} all the points where any of the product $\phi(i,j)\cdot\phi(i\pm1, j\pm1) < 0$, and Ω is the cumulative volume of all the considered cells; if the computation is extended to all the computational domain, the error corresponds to the global L_1 norm. The infinity norm $\|\mathrm{err}\|_\infty$ corresponds to the maximum error in the points set.

When making the comparison between the uniform and adaptive mesh computations in figure 6.2, it is almost impossible to distinguish any difference between the two set of computations. In both the simulations the numerical dissipation shows into the rounding of the fence angles, and becomes smaller with the increasing meshes. The base mesh able to capture the disk seems to be the 128^2. In order to put in evidence the differences between the two set of computations, tables 6.1 and 6.2 resume the two computed error, local and global. In the local errors, computed around the zero contour, a visible difference can only be seen on the finest meshes, 128^2 and 256^2, where more AMR levels are present. Still, the adaptive mesh induced error does not exceed 4% at most; neither the maximum error does.

The global error, as expected, is in case of the AMR computation a mean over all the refinement level: for these values, there is not any benefit from the adaptive mesh, even if the



(a) Uniform meshes results (b) Adaptive meshes results

Figure 6.2: Results of the Zalesak disk after one full rotation for different meshes. Black-initial condition; red-32^2; orange-64^2; green-128^2; blue-256^2

	Uniform				Adaptive			
Grid	$\|\text{err}\|_1$	$\|\text{err}\|_\infty$	q_1	q_∞	$\|\text{err}\|_1$	$\|\text{err}\|_\infty$	q_1	q_∞
32	0.8472	2.0665	-	-	0.8472	2.0665	-	-
64	0.3552	1.8366	1.25	0.17	0.3552	1.8370	1.25	0.17
128	0.1002	0.5744	1.83	1.68	0.1003	0.5792	1.82	1.67
256	0.0192	0.2723	2.38	1.08	0.0201	0.2727	2.35	1.09

Table 6.1: Mean and maximum local errors for the Zalesak disk rotation, uniform and adaptive meshes.

cell error is integrated over the cell size. From the two set of simulations the outcome is that the finest mesh precision can be maintained, for the hyperbolic equation, by keeping the region of interest on the refined zone.

Effect of interpolation

The simulations presented in the previous section about advection with AMR are performed with a fixed interpolation algorithm on coarse/fine interfaces, which consists of the symmetrical second-order prolongation (described in chapter 4) and a four points average restriction. These algorithms are called each time guardcells are filled as well as when the mesh evolves, as newly spawn children need initial conditions to be set, or blocks to be deleted need to save their data to their parent. Results show that such accuracy is sufficient to keep the fine mesh accuracy in the advection of the Zalesak disk; still, it is interesting to evaluate its influence on the final error. Five different types of prolongation-restriction have been tried:

- ▷ p0-r1: direct injection prolongation and four points average restriction;
- ▷ p1-r1: linear prolongation and four points average restriction;
- ▷ p2-r1: second order prolongation and four points average restriction;

	Uniform				Adaptive			
Grid	$\|\text{err}\|_1$	$\|\text{err}\|_\infty$	q_1	q_∞	$\|\text{err}\|_1$	$\|\text{err}\|_\infty$	q_1	q_∞
32	0.2932	4.7713	-	-	0.3218	4.3954	-	-
64	0.1057	4.0770	1.47	0.23	0.1552	4.2440	1.05	0.05
128	0.0567	3.7842	0.90	0.11	0.1163	3.9555	0.42	0.10
256	0.0431	3.6609	0.40	0.05	0.1058	3.6941	0.14	0.10

Table 6.2: Mean and maximum global errors for the Zalesak disk rotation, uniform and adaptive meshes.

- ▷ p2-r2: second order prolongation and second order restriction (not symmetrical);
- ▷ p3-r2: third order prolongation (not symmetrical) and second order restriction (not symmetrical).

In table 6.3 the computed local and global error are presented for the five cases. The local error decreases as the interpolation order is increased to first and second order, but from here its values keeps stable. The same behaviour can be seen in the global error: the third order gives worse results than the second. The restriction order is not so important: only a small benefit can be seen far from the finest level, as the global error is a bit smaller. If the symmetry property is considered, it becomes clear that the combination of a parabolic prolongation and an average restriction gives the best results, and for this reason it will be kept for all the following AMR simulations.

	Local		Global	
Interpolation	$\|\text{err}\|_1$	$\|\text{err}\|_\infty$	$\|\text{err}\|_1$	$\|\text{err}\|_\infty$
p0-r1	0.5966	2.828	3.4348	72.317
p1-r1	0.8487	3.688	2.9655	12.106
p2-r1	0.1004	0.576	0.1483	4.185
p2-r2	0.1003	0.5792	0.1163	3.9555
p3-r2	0.1003	0.5762	0.1209	4.3159

Table 6.3: Error in the Zalesak disk advection with different interpolation orders: p-prolongation; r-restriction.

1.2 Serpentine

The serpentine test for the hyperbolic linear advection equation was introduced by Bell et al [8]. Differently from the previous test, in this problem the velocity field induce a deformation of the initial Level Set condition. Starting from a circle shape, the interface deforms itself in the generation of a long ligament. The interest of this case is evident because the ligament stretches itself indefinitely as the time advances, so the numerical simulation will eventually face an under resolution situation. The code must be accurate enough to maintain the ligament even when its width approaches the cell size, and so to maintain the initial mass.

The simulation parameters are the following: a circle of radius $r = 0.15$ is centered on the point $(0.5, 0.75)$ inside a square domain $[0, 1]^2$. A stationary rotating velocity field is defined by

the potential

$$\psi = \frac{1}{\pi}\sin^2(\pi x)\sin^2(\pi y) \tag{6.3}$$

so that the components u, v are defined by the equations

$$\begin{cases} u(x,y) = -2\sin^2(\pi x)\sin(\pi y)\cos(\pi y) \\ v(x,y) = -2\sin^2(\pi y)\sin(\pi x)\cos(\pi x) \end{cases} \tag{6.4}$$

Some numerical solutions of this test problem are investigated, in particular the effect of the solution on adaptive mesh. Differently from the previous case, no exact solution are available. A reference solution obtained by an high-order simulation is employed in the graphical results. In Couderc [18] several comparisons are made between the conservative and non conservative versions of the advection equation, and the effect of the redistance on both algorithms. In this section the redistance is always active because the distance information is needed by the refinement test detector. The shapes of the advected serpentine with regular and mesh refinement

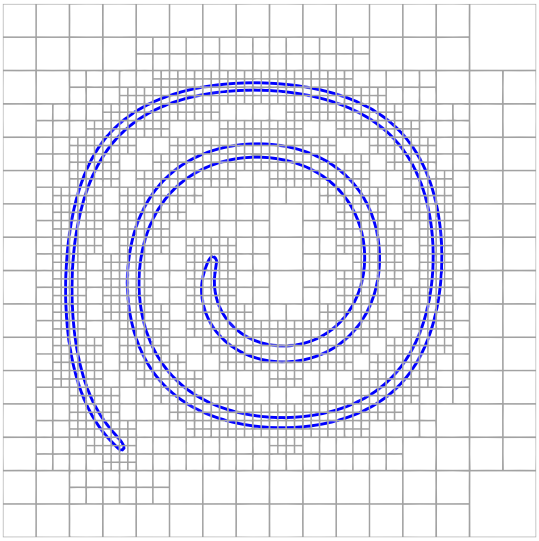
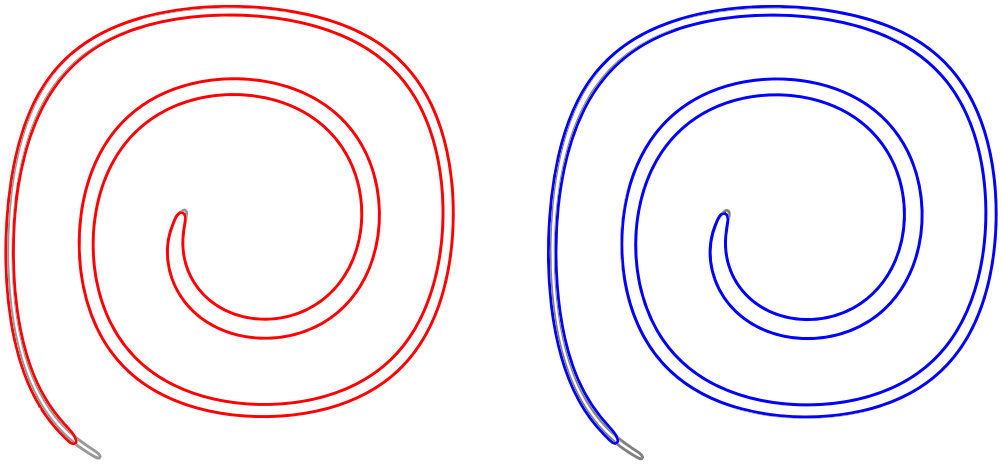


Figure 6.3: Solution of the serpentine test with four levels of adaptive mesh, 4×4 cells per block, equivalent resolution 256^2, $t = 3$s

(presented in figure 6.3)are compared in figure 6.4. No differences are visible so far. Very small contour shift are visible only within the cell scale. The adaptive mesh seems capable of correctly maintaining the ligament until the finest cell size is reached, a fundamental behaviour with a view to the simulation of the atomization process. A more quantitative test is the mass conservation check. The surface of the serpentine can be measured and compared to its initial value. An exact conservation algorithm should present no variations. This is not the case of the Level Set method. In figure 6.5 the temporal evolution of this quantity is tracked up to $t = 3$ s for both discretization schemes, conservative and non conservative. The solid markers define the adaptive mesh computation: they fall almost exactly on the underlying curbes, which define the uniform mesh computation. Confirmation is given to the graphical results: the conservation of the mass is as correct as it is for the base algorithm, the accuracy of the finest mesh is maintained.



(a) Uniform simulation (b) Adaptive mesh simulation

Figure 6.4: Numerical solution of the serpentine test, 256^2, $t = 3$s. The simulation is superimposed to a reference solution computed with an higher resolution of 512^2.

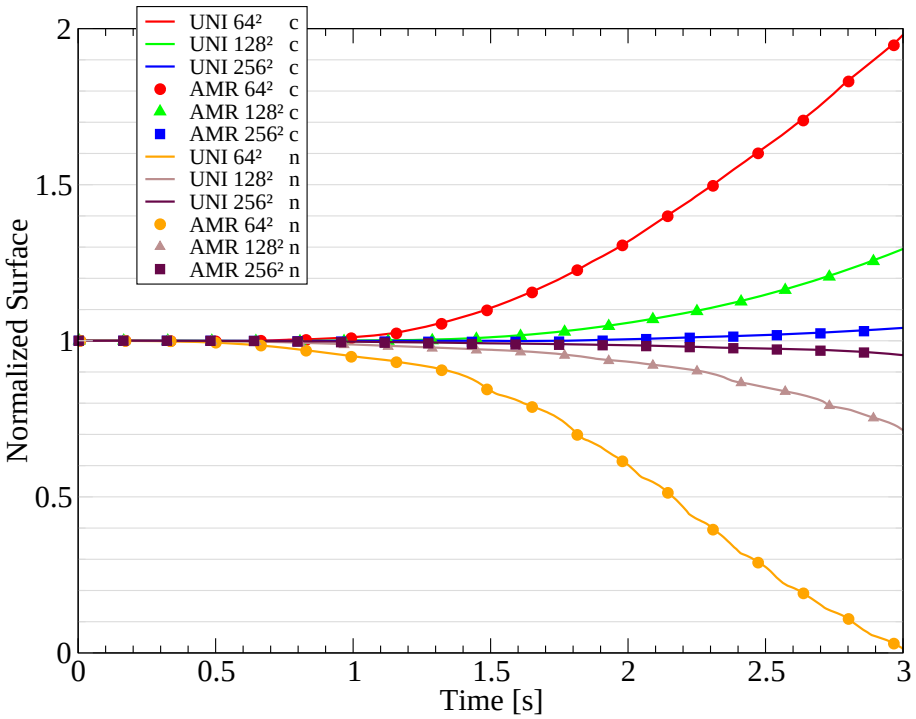


Figure 6.5: Serpent mass temporal evolution, different resolutions, $t = 3$ s. c indicates the conservative scheme, n the non conservative. The curves are from the uniform mesh, the points are samples of the AMR computations.

2 Single phase Navier-Stokes solver

2.1 Taylor-Green vortex

There are not many test cases for the incompressible viscous flows which have an analytical solution, and even less adapted to the application of the AMR strategy. For an AMR computation two different measures of error can be considered: first the error of the full composite grid solution (the error of the solution over the entire domain, correctly normalized on the cell volumes) and second the error of the solution in the region of maximum refinement alone. The importance of each measure depends on the nature of the calculation and the purpose of refinement. If the aim is purely to resolve a specific region or feature of the flow as well as possible, and one only cares about the rest of the solution as far as it affects the refined region, then the latter measure is most relevant. If, on the other hand, the goal is to compute the entire solution using adaptivity to improve the accuracy where errors are large then the former error is better suited.

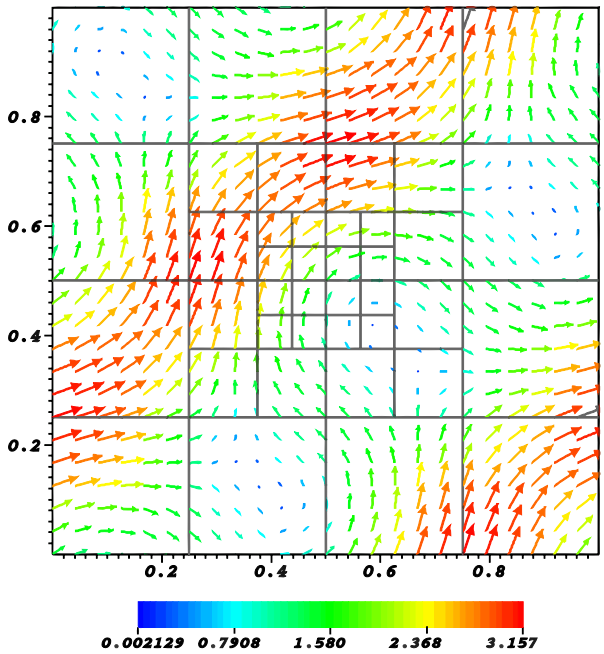


Figure 6.6: The test case of the travelling Taylor-Green vortices, with the three levels of adaptive mesh.

In this section the Navier-Stokes is tested on the Taylor-Green vortices diagonal translation, at first on an uniform domain as done in Couderc [18], then on a composite mesh as performed in Almgren et al [3]. The problem consists of a periodical square computational domain $[0, 1]$ in which the following initial velocity field is defined:

$$\begin{cases} u(x, y) = 1 - 2\cos(2\pi x)\sin(2\pi y) \\ v(x, y) = 1 + 2\sin(2\pi x)\cos(2\pi y) \end{cases} \tag{6.5}$$

The exact solution for the couple pressure-velocity is

$$\begin{cases} u(x, y, t) = 1 - 2(\cos(2\pi(x - t))\sin(2\pi(y - t))e^{-8\pi^2\nu t} \\ v(x, y, t) = 1 + 2(\sin(2\pi(x - t))\cos(2\pi(y - t))e^{-8\pi^2\nu t} \\ p(x, y, t) = -(\cos(4\pi(x - t)) + \cos(4\pi(y - t)))e^{-16\pi^2\nu t} \end{cases} \tag{6.6}$$

This corresponds to four vortices which translate diagonally in the north-east direction. ν is taken equal to 0.01.

An arbitrary three levels mesh, with the intermediate level occupying the $[0.25, 0.75]^2$ region and the finest the innermost one $[0.375, 0.625]^2$ is build to test the convergence for velocity and speed for the Navier-Stokes solver (figure 6.6). The resolution is increased without changing the mesh configuration (blocks have increasing number of cells). Two error norms are presented, the L_2 norm for the whole computational domain and a local L_2 norm for the finest level alone. They are defined as

$$e = \sqrt{\frac{1}{\Omega}\sum_{ij}\left(x_{ij} - x_{ij,exact}\right)^2 \cdot \Delta x \Delta y} \tag{6.7}$$

being x the pressure or a velocity component (any of the two), Ω, the volume of the computational domain considered, the finest level in the first case and the whole domain in the second.

The aim of this test is not to perform an error comparison between levels, but to check the conservation of the global composite Navier-Stokes solver convergence rate in presence of the adaptive mesh. In fact, when the refined grid is placed non optimally as in this calculation, the error in the refined patch is comparable to the error at the resolution of the base grid: the use of refinement is not improving the accuracy, as at the final time the flow has passed on each level of refinement, collecting error from all the refinement levels. Three different interpolation strategies are compared for the velocity (all of them detailed in chapter 4): two polynomial interpolation, linear and quadratic (parabolic), and the divergence preserving prolongation of Balsara. The aim is to find which strategy allows the conservation of the convergence rate of the base solver.

	Finest				Global			
Mesh	$\|e_{vel}\|_2$	q	$\|e_{pres}\|_2$	q	$\|e_{vel}\|_2$	q	$\|e_{pres}\|_2$	q
2th								
64^2	5.14e-3	-	6.47e-3	-	6.69e-3	-	6.80e-3	-
128^2	1.10e-3	2.23	1.81e-3	1.83	1.15e-3	2.54	2.10e-3	1.69
256^2	3.12e-4	1.82	2.76e-4	2.72	3.24e-4	1.83	4.90e-4	2.10
Bals								
64^2	1.51e-2	-	2.01e-2	-	1.98e-2	-	2.55e-2	-
128^2	3.04e-3	2.31	5.25e-3	1.94	2.92e-3	2.76	4.11e-3	2.63
256^2	7.80e-4	1.96	1.52e-3	1.79	7.62e-4	1.94	1.02e-3	2.02
1th								
64^2	2.13e-2	-	3.63e-2	-	3.64e-2	-	4.24e-2	-
128^2	2.09e-2	0.02	3.30e-2	0.13	1.48e-2	1.30	1.83e-2	1.21
256^2	1.77e-2	0.24	2.81e-2	0.23	9.62e-3	0.62	1.26e-2	0.55

Table 6.4: Error and convergence rate in velocity ($\|e_{vel}\|_2$) and pressure ($\|e_{pres}\|_2$) for the Taylor-Green vortex advection, three different guardcell interpolation strategies: parabolic, Balsara and linear interpolation. Resolution is given for the finest level.

Result are presented in table 6.4. They clearly show that the linear interpolation can not be taken in consideration, because the convergence rate is very low, lower than the possibly expected linear behaviour. The two more accurate interpolations show much better properties: the Balsara is able to keep a steady second order on velocities, while the polynomial keeps a little

bit behind, with values varying between 1.8 and 2. However, in terms of pure error the parabolic seems more accurate, with two times smaller errors. The convergence rates are the same for the finest level alone as well as for the global domain. In general, the parabolic interpolation is used for the maximal accuracy; eventually, the Balsara has to be taken in consideration for its intrinsic property of divergence preservation, a characteristic which may become necessary for more complex and unstable simulations than the validation test cases proposed in this chapter.

2.2 Mixing layer

The two dimensional simulation of a periodic mixing layer was proposed in Bell et al [8] to test the performance of a numerical incompressible Navier-Stokes solver. The simulation involves two horizontal layers moving with opposite velocity, subjected to a perturbation of a vertical sinusoidal velocity. This test is realized in a square $[0, 1]$ domain with periodical boundary conditions everywhere. The initial velocity field is:

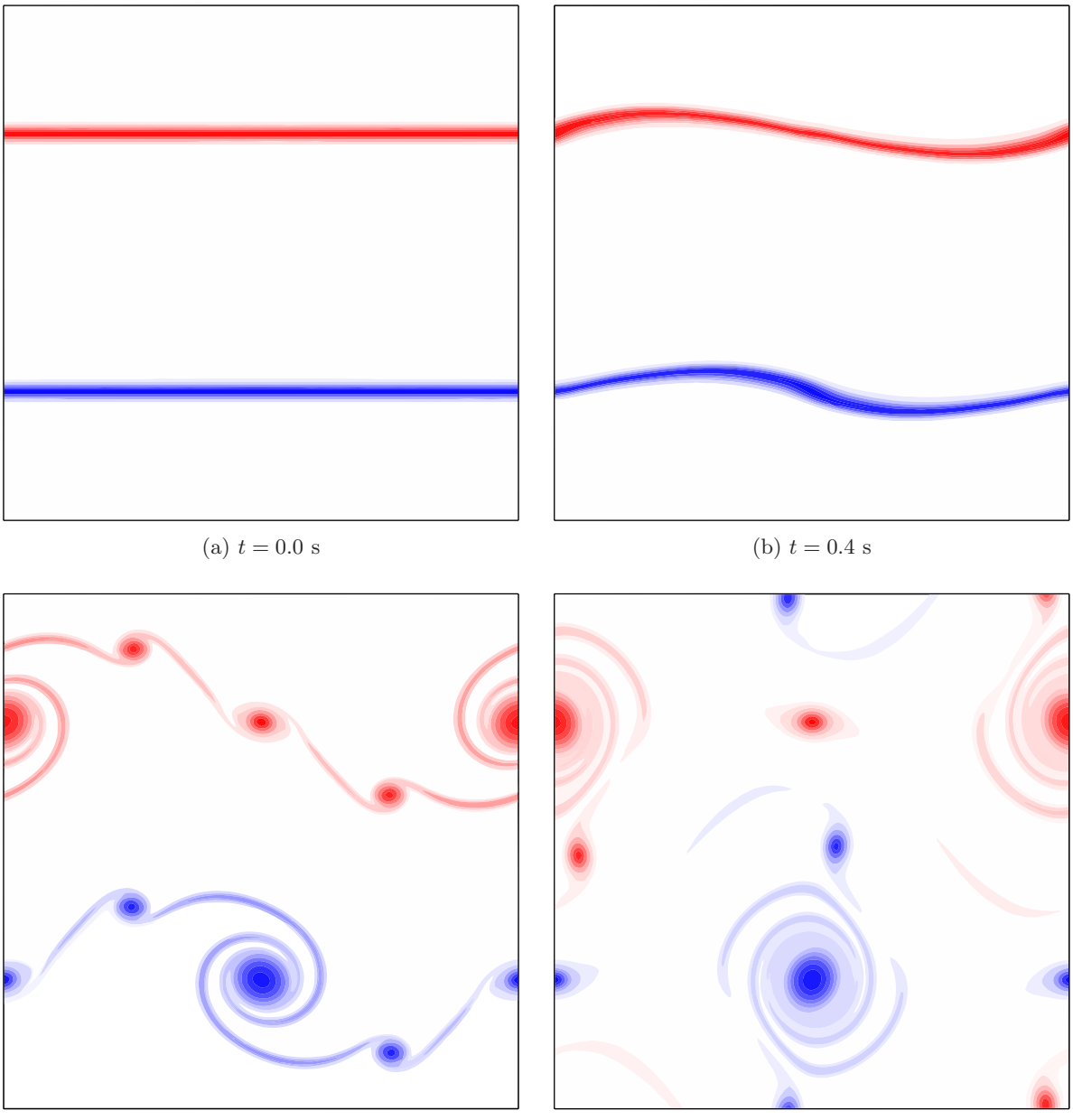
$$\begin{cases} u(x,y,t) = \tanh((y-0.25)/\rho) \ , & y < 0.5 \\ u(x,y,t) = \tanh((0.75-y)/\rho) \ , & y > 0.5 \\ v(x,y,t) = \delta \sin(2\pi x) \end{cases} \tag{6.8}$$

where ρ is the thickness of the mixing zone and δ the initial perturbation amplitude; the retained value of thickness for this simulation is $\rho = 80$, which defines a quite small mixing layer, and a perturbation of $\delta = 0.05$. The perturbation generates an instability in the mixing zone defined by the hyperbolic tangent defined profiles. The instability grows into the generation of eddies at around $t = 0.8$ s as in figure 6.7. The initial thickness of the unstable zone defined by the parameter ρ determine the size of the generated vortexes: the smaller the zone, the smaller the vortexes, and more difficult is the numerical capturing of these phenomena. There are no exact solutions for this problem, neither for the inviscid nor for the viscous case. As in Couderc [18], the interest here is not to evaluate the absolute accuracy of the solver, but instead to measure the performance of the code about the numerical dissipation of the advection schemes due to the truncation errors. Another interesting aspect of this simulation is the apparition of parasite vortices due to local under resolution (Minion et al [71]), which disappear in case of increased resolution. In particular, the effect of a local refinement on a "difficult" region of the computational domain can be put in evidence, as well as the impact of the composite Navier-Stokes solve. The parasite vortices can already be seen in figure 6.7(c), and are demonstrated in Minion et al [71] to be consequence of the small wavelength instability growth coming from the truncation errors. More dissipative schemes tend to dissipate them more, as shown in Couderc [18] for a conservative and non conservative resolution of the momentum equation.

The computations have been performed with an adaptive mesh. The mesh is refined on all the blocks showing a peak of vorticity (as it can be seen on the pictures 6.7, the regions of high vorticity are well concentrated), so that the refined mesh follows naturally the mixing zone. The hypothesis which is consequence of this meshing strategy is that the refined mesh would be able to contain the most part of the loss of kinetic energy, if compared to an uniform computation. Results and a mesh convergence test are shown in figure 6.8 and table 6.5. The kinetic energy is evaluated for all the meshes as

$$E_{kin} = \frac{1}{\Omega} \sum_{ij} (u_{ij}^2 + v_{ij}^2) \cdot \Delta x \Delta y \tag{6.9}$$

being $\Omega = 1$, the volume of the computational domain. The time histories of the kinetic energy start to decrease at about $t = 0.6$ s, approximately the moment when the eddies begin to form. After the first descent, the value tends to stabilize a little bit, mostly for the finer meshes. The



(a) $t = 0.0$ s (b) $t = 0.4$ s

(c) $t = 0.8$ s (d) $t = 1.2$ s

Figure 6.7: Simulation of a thin mixing layer, resolution of 256. Four time steps. Vorticity contours, blue=positive, red=negative.

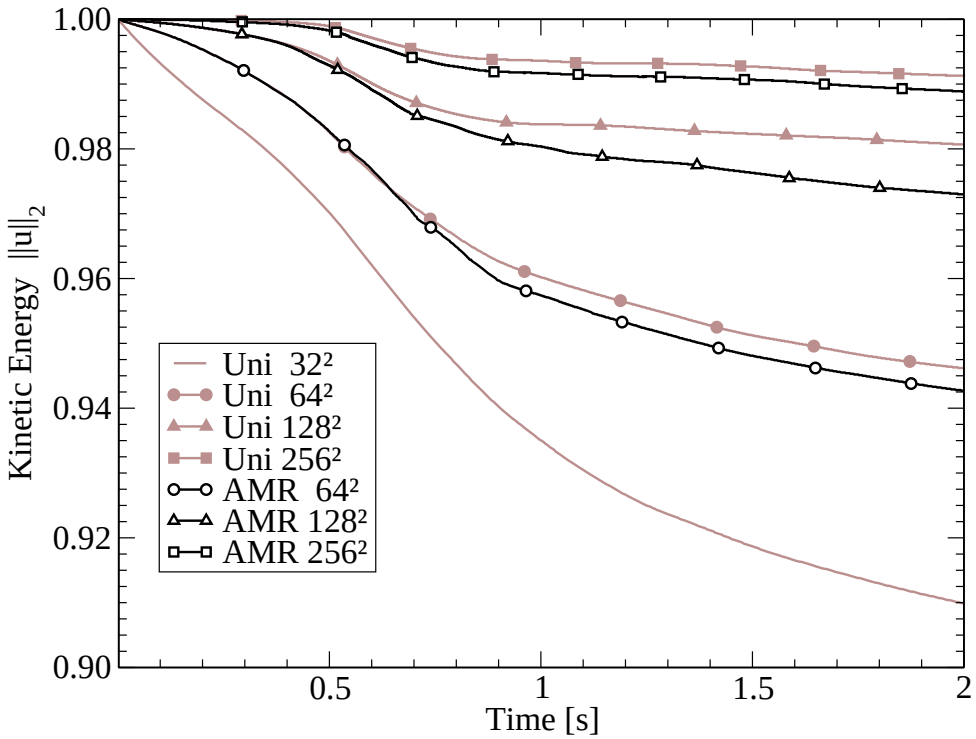
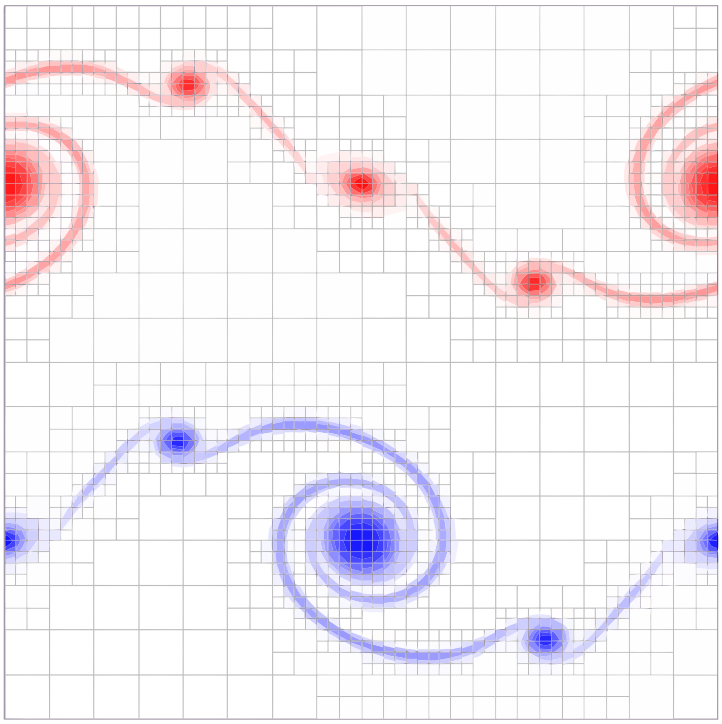


Figure 6.8: Reduction in time of kinetic energy due to the numerical dissipation, for both the single grid and the AMR computations, $t = 0.8$ s.

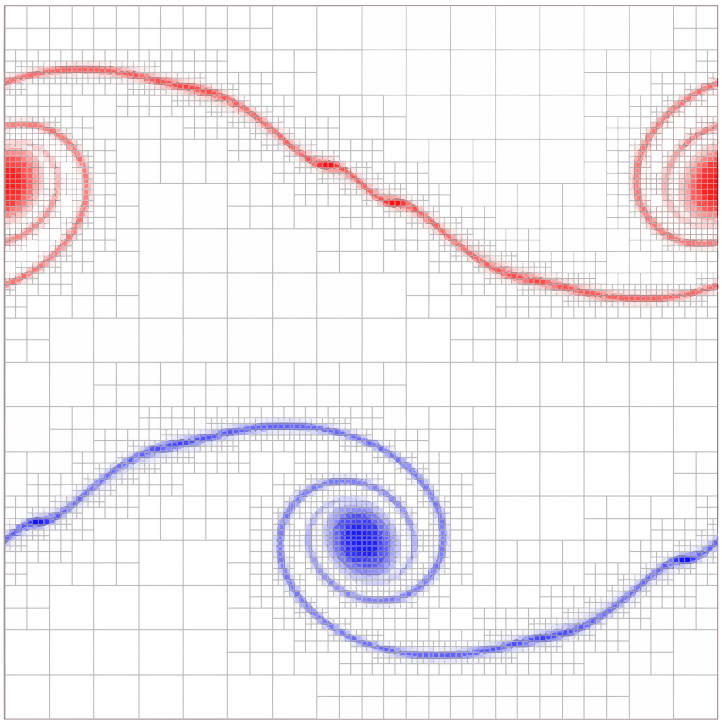
behaviour of the AMR solution seems to agree quite strictly with the reference computations. The intermediate computation, with an equivalent fine resolution of 128, shows a bigger discrepancy for unknown reasons. The computation seems, however, strongly dependent on the temporal accuracy (i.e. the CFL condition chosen). In any case the main source of dissipation can be located on the strong vorticity regions where the parasite vortexes are generated. A visual clue of the local mesh refinement effect can be seen in figure 6.9, where the solution at $t = 0.8$ s is shown for three levels of AMR (256^2) and an added fourth level (512^2): the central parasites vortices are much more reduced in the second case, while the one-quarter domain small ones have disappeared.

Mesh	Grid	$\|U\|_2$ loss [%]	q
Base	32^2	9.01	-
Uniform	64^2	5.38	0.74
	128^2	1.93	1.48
	256^2	0.88	1.13
AMR	64^2	5.73	0.65
	128^2	2.70	1.09
	256^2	1.20	1.17

Table 6.5: Kinetic energy at $t = 2$s, AMR versus single grid computation.



(a) Three levels, maximum resolution of 256^2



(b) Four levels, maximum resolution of 512^2

Figure 6.9: Effect of local refinement on the strong vorticity region: the refined mesh can significantly reduce the parasite vortices.

3 Two phase solver

3.1 Static bubble

The simulation of a round static bubble is the first interesting test for the two phase numerical method. Although very simple in theory, it can give informations about the precision of the interface treatment, in particular of the surface tension and the jump conditions. A round bubble of a dense fluid is centred in a square domain (figure 6.10), surrounded by a lighter one; there are no initial velocities and no gravity forces. The bubble is the most stable shape, as the total

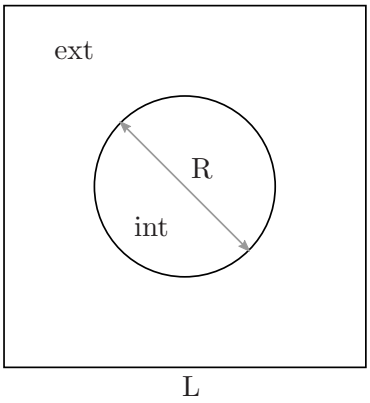


Figure 6.10: Static bubble scheme and notation.

surface of interface is minimized. In the simulation case the bubble has a higher pression than the surrounding fluid the pressure jump is imposed by the interface curvature and the surface tension product, as given by the Laplace equation

$$\begin{cases} p_{int} - p_{ext} = \sigma/R, & \text{if in 2D} \\ p_{int} - p_{ext} = 2\sigma/R, & \text{if in 3D} \end{cases} \tag{6.10}$$

The resulting pressure field in absence of gravity is constant inside and outside the bubble with a discontinuity localized on the interface (as seen in the Poisson solver results, chapter 5). In theory no convective velocity should intervene into the momentum equation. However, the difficulties to numerically compute the normal direction on a curve give rise to small errors on the pressure computation around the jump; the subsequent pressure gradient acts as a wrong correction for the predicted velocity field, generating small artificial velocities known as *spurious currents*, non zero components which arise from the first iteration, and may in some cases destroy the interface.

Two set of simulations are presented, results from the first iterations and the evolution of the currents for a long time of simulations. In each of them the adaptive mesh solution is compared to the uniform fine mesh one. The adaptative mesh is generated by refining on the zero contour of the Level Set function; blocks with 4×4 cells are employed to maximize the number of allowed levels. The fine mesh resolution varies from 32^2, where there is no room for refinement, to 64^2,128^2 and 256^2, with respectively 2, 3 and 4 level of refinement. The parameters of the

simulation, taken from Couderc [18], are:

$$\begin{cases} \rho_{int} = 1000^3 \text{ kg.m}^{-3} \\ \rho_{ext} = 1 \text{ kg.m}^{-3} \\ \mu_{int} = 10^{-3}\text{Pa.s} \\ \mu_{ext} = 10^{-5}\text{Pa.s} \\ \sigma = 0.1 \text{ N.m}^{-1} \\ L = 4 \text{ cm} \\ R = 1 \text{ cm} \end{cases} \tag{6.11}$$

Only one iteration of Level Set reinitialization is performed in order to reduce any artificial displacement of the interface. As the theoretical velocity field is zero, the error norm can be actually given by the velocity norm

$$\text{err} = \|\boldsymbol{u}\|_{L2} = \left(\frac{1}{\text{n}_x \text{n}_y} \sum_{i,j} \left(u_{i,j}^2 + v_{i,j}^2 \right) \right)^{1/2}$$

If the analytical curvature is used as input, then the solution is always precise to the roundoff error. As shown in table 6.6, the convergence rate, given by

$$q = \frac{\ln\left(\text{err}_{2h}\right) - \ln\left(\text{err}_h\right)}{\ln(2)}$$

are, for the uniform mesh computation, close to 3: this behaviour, justified in Oevermann and Berger [76], comes from the zero gradient solution of the pressure on both the sides that nullify the effects of the interface position relative to the grid. In the adaptive mesh test a slight decrease of the code performance, mainly for the finer resolutions, is visible. It has to be kept in mind that there is no lower limit to the refinement level, so that the lowest grids are kept, far from the interface, as the resolution increases. The effects of the coarser grids and the interpolation are visible in the small spurious current increase. Still, the convergence rate never drops under 2.5. The growth of the spurious currents are checked in table 6.7 for a prolonged

Mesh	Grid	$\|U\|_2$	q
Uniform	16^2	1.933e-06	-
	32^2	1.910e-07	3.34
	64^2	2.485e-08	2.94
	128^2	3.277e-09	2.92
	256^2	3.893e-10	3.07
AMR	16^2	1.933e-06	-
	32^2	2.125e-07	3.19
	64^2	3.259e-08	2.70
	128^2	5.717e-09	2.51
	256^2	9.451e-10	2.60

Table 6.6: Spurious currents norm, first iteration. AMR grid correspond to the finest level.

time of simulation. In this case the code tries to achieve an equilibrium for those non-physical

Mesh	Grid	$\|U\|_2$	q
Uniform	16^2	1.885e-03	-
	32^2	5.828e-04	1.69
	64^2	1.778e-04	1.71
	128^2	3.879e-05	2.20
	256^2	1.060e-05	1.87
AMR	16^2	1.885e-03	-
	32^2	5.676e-04	1.73
	64^2	2.181e-04	1.38
	128^2	5.545e-05	1.98
	256^2	2.953e-05	0.91

Table 6.7: Spurious currents norm, $t = 1$ s. AMR grid correspond to the finest level.

velocities, with the generation of symmetrical non zero vorticity zones near the interface. The intensity of this vorticity grows in time, with an initial steep growth, and the stabilizes near a certain value, as shown in figure 6.11. In the AMR computation the initial error grows, as for the first iteration, a little more than in the uniform mesh; still it does not rise dangerously, but it stabilizes as for the previous test. The convergence rates are now smaller, near to 1.5. In the most refined case the convergence rate drops under 1: probably here the influence of the coarsest mesh is very important, as the gas had time enough to well recirculate outside the refined zone. Other numerical tests show a certain influence from the size of the blocks: if the refinement jumps are too close to the interface (this can happen on the diagonal directions, because of the square shape of the blocks, when they are too small, e.g. 4x4), then the interpolation effect can directly affect the velocity on the interface, where the spurious currents are generated, thus increasing dramatically their intensity. The comparison of the uniform and adaptive mesh error show how an added refinement level actually improves the solution, even if for larger numbers of levels the gain becomes smaller, mainly for long simulations.

3.2 Damped surface wave

In this section the computational performances of the code are measured in the simulation of the damped wave of a surface dividing two viscous fluids, the lighter superimposed to the heavier one (figure 6.12). Without gravity, an initial sinusoidal perturbation of the interface starts to oscillate under the effect of the surface tension; the amplitude is damped in time by the viscosity. The interest of this simulation lies into the presence of both the surface tension and the viscosity, each of them equally important: if the fist behaves as a spring for the system, the second acts as damper. None of the Navier-Stokes terms is here negligible, and the accuracy of the resolution of the jump conditions determine the capability of the code to respect the theoretical solution. A square computational domain of side L with periodic boundary conditions on the vertical faces and slip conditions on the horizontal ones is used. Each half of the domain is occupied by a different fluid. The surface height is given by

$$a(t) = a_0 \cos(kx), \quad \text{with} \quad k = \frac{2\pi}{L} \quad \text{and} \quad a(0) = \frac{0.04L}{2} \tag{6.12}$$

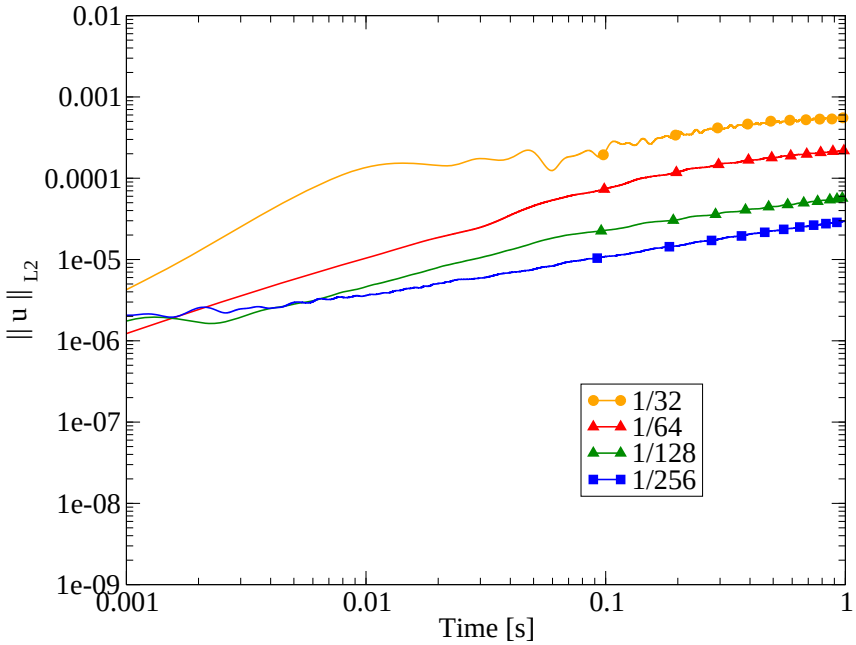


Figure 6.11: Evolution of the spurious currents for a long time simulation, $t = 1$ s, adaptive mesh.

with $L = 2\pi$. Some theoretical solutions are available. The normal mode analysis of Lamb [49] gives the dispersion relation under the infinite vertical dimension hypothesis;

$$\omega^2 = \frac{\sigma k^3}{\rho_l + \rho_g} \tag{6.13}$$

The damping of the oscillation is exponential in time when the viscosity is not negligible. If the two fluids have the same viscosity, an analytical solution of the temporal evolution of the amplitude $a(t)$ is given by Prosperetti [91]:

$$\begin{aligned} a(t) = {} & \frac{4(1-\beta)\nu^2 k^2}{8(1-4\beta)}\ a(0)\ \mathrm{erfc}\left(\sqrt{\nu k^2 t}\right) + \\ & \sum_{i=1,4}^{4} \frac{z_i}{Z_i}\left(\frac{\omega^2 a(0)}{z_i^2 - \nu k}\right)\ \exp\left(\omega^2 a(0) z_i^2)t\right)\ \mathrm{erfc}\left(z_i t^{\frac{1}{t}}\right) \end{aligned} \tag{6.14}$$

where the terms z_i are the four roots of the algebraic equation

$$\begin{aligned} & z^4 - 4\beta(k^2\nu)^{1/2} z^3 + 2(1-6\beta)k^2\nu z^2 + \\ & 4(1-3\beta)(k^2\nu)^{3/2} z + (1-4\beta)k^4\nu^2 + \omega^2 = 0 \end{aligned}$$

The Z_i comes from the cyclical index permutations of

$$Z_i = (z_2 - z_1)(z_3 - z_1)(z_4 - z_1)...$$

The inviscid oscillation frequency ω is

$$\omega^2 = \frac{\sigma k^3}{\rho_l + \rho_g}$$

and the parameter β is given by the fluid densities as

$$\beta = \frac{\rho_l\ \rho_g}{(\rho_l + \rho_g)^2}$$

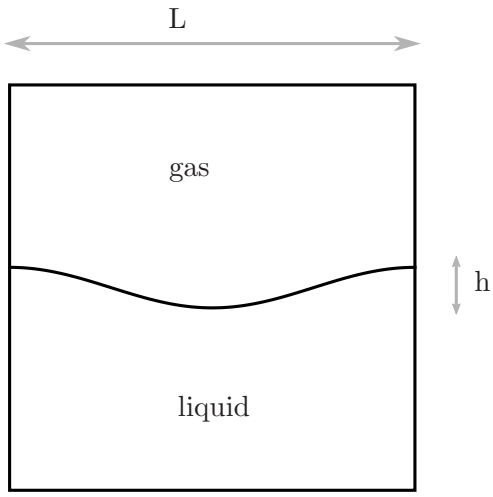


Figure 6.12: Initial condition and notation for the damped surface wave.

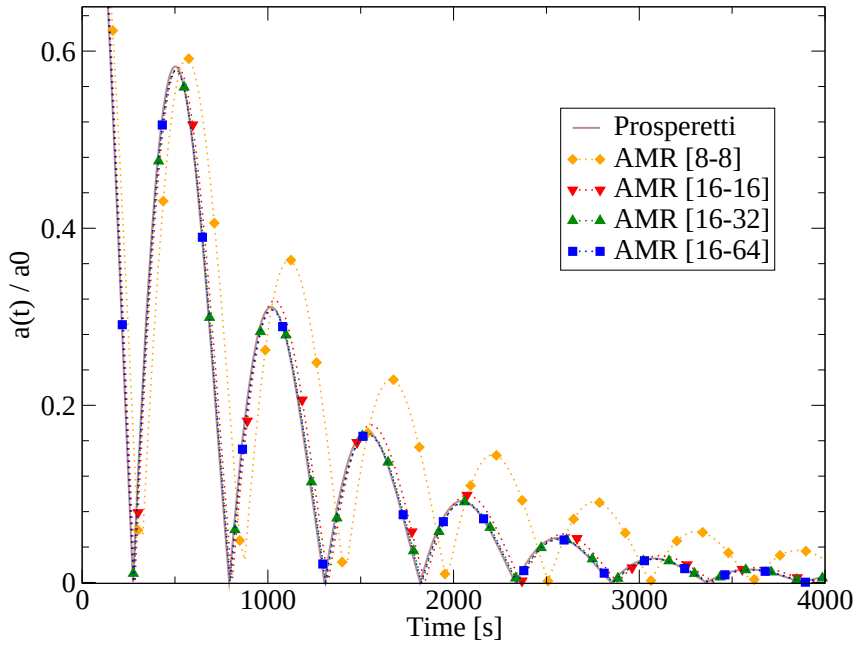
Two sets of simulations are presented, with different density ratios: in the first it is set to 1, in the second one it is raised to 1000. One Level Set redistance iteration is performed. For all the test cases the refined mesh is used, but it should be pointed out that for these simulation the mesh never changes in time: given the oscillation amplitude, there is not room enough for refinement, as the resolutions should be risen excessively in order to follow the interface with very small blocks. The influence of the connection between different grid resolutions can still be tested, leaving behind for a moment the effect of creation and removal of blocks. This will be tested more extensively in the next section. The simulation parameters are:

$$\begin{cases} \rho_g = 1 \text{ or } 1000 \text{ kg.m}^{-3} \\ \rho_l = 1000 \text{ kg.m}^{-3} \\ \nu_g = 10^{-5} \text{Pa.s} \\ \nu_l = 10^{-3} \text{Pa.s} \\ \sigma = 0.1 \text{ N.m}^{-1} \\ L = 4\pi \text{ cm} \\ h = 0.04L \end{cases} \tag{6.15}$$

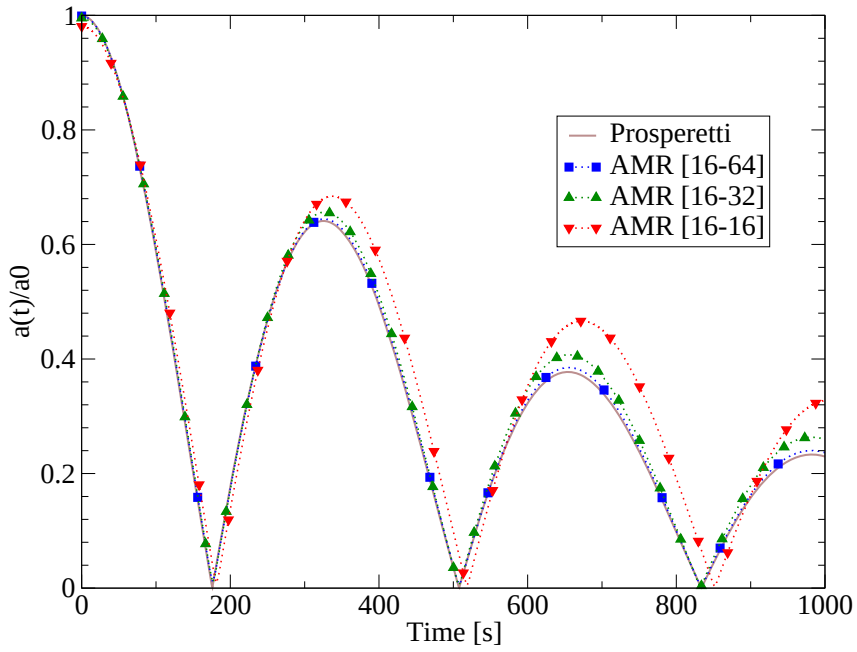
The high density ratio case is clearly the most difficult, and it has been rarely used in literature (Oevermann and Berger [76], Popinet and Zaleski [88]): the CFL condition had to be reduced up to 0.15 to avoid stability issues. In figure 6.13 a good agreement between the analytical and the computed solution is shown. The first case is clearly the easier one, as the convergence is faster: the 16^2 mesh is already capable of tracking the right wavelength of the oscillation. In the second case the more refined 32^2 and 64^2 meshes are needed to correctly capture the right amplitude. In order to quantify the error the following norm is built:

$$\|\text{err}\| = \frac{1}{N} \sum_{i=1}^{N} |a_{num}(t_i) - a_{theo}(t_i)|$$

that is the mean value of the distance of our computed value from the Prosperetti's theoretical one in equation (6.14). The simulation is stopped at time $t = 4000$ s for the first case and $t = 1000$ s for the second. Results are grouped in table 6.8. The convergence rates for the low density ratio decrease dramatically with the mesh size: probably in this simulation the effect of the initial hypothesis of infinite domain and small deformations become evident as the numerical



(a) Density ratio 1



(b) Density ratio 1000

Figure 6.13: Analytical and computed amplitude time histories for the damped surface wave test.

solution approaches quickly the exact one. The convergence falls below the unity for the finest mesh: this sublinear convergence has been also reported in Oevermann and Berger [76] as well as from Couderc [18]. Nevertheless, it seems that an extension of the vertical dimension of the domain does not improve the results. Quite surprisingly, the convergence rates for the strong density ratio are well superlinear and more or less constant with the mesh size. In any case the intrinsic precision of the code can be appreciated, and the results of simulation of similar phenomena should not be aberrant even with not so refined meshes. More troubling is the

	Grid	err	q
$\rho_l/\rho_g = 1$	8^2	0.0689	-
	16^2	0.0140	2.30
	32^2	0.0053	1.41
	64^2	0.0029	0.85
$\rho_l/\rho_g = 1000$	8^2	0.1492	-
	16^2	0.0512	1.54
	32^2	0.0140	1.87
	64^2	0.0041	1.77

Table 6.8: Results of damped wave simulation, error integrated over time.

reduction of the uppermost CFL limit for the high density ratio computation: the already small time steps due to the total explicit treatment is in these cases divided up to four for the finest meshes. The numerical method needs some improvement to overcome this problem; on the other hand the presence of an adaptive mesh does not interfere with the capability of the code to behave well even in this difficult simulation.

3.3 Rayleigh-Taylor instability

A common test problem for a two phase flow numerical method is the Rayleigh-Taylor type instability. It is easy to set up and many results are available in literature (Oevermann and Berger [76], Popinet and Zaleski [88]). This phenomenon can be observed by superimposing a heavy fluid over a lighter one. If the interface is horizontal, they are in a situation of unstable equilibrium. An initial sinusoidal perturbation of a given wavelength is subject to the destabilizing effect of gravity and the stabilizing effect of surface tension: if the destabilizing effect prevails, which happens when the wavelength and the surface tension are small, then the perturbation grows, as the heavier fluid moves down and deplaces the lighter one, generating the typical mushroom shape. With this kind of simulation the aptitude of the code to respect the solution predicted from the linear stability theory can be checked. By considering a long time simulation, the instability amplitude is allowed to exceed the limit of the linear theory approximation. The subsequent large deformations of the interface and the consequent formation of thin ligaments are shown in this simulation.

Small amplitude, linear theory hypothesis

For short time simulations, under the linear theory hypothesis, the initially sinusoidal disturbance $a = a(0)\cos(kx)$ grows following an exponential law $a(t) = a(0)\exp(nt)$, with n predicted by the linear theory relation

$$n^2 = kg\left(\frac{\rho_h - \rho_l}{\rho_h + \rho_l} - \frac{k^2\sigma}{g\left(\rho_h + \rho_l\right)}\right)$$

given the value of surface tension and the wavelength small enough to let the instability to take place and and inviscid fluids. Given $k = 1$, a limit value σ_{crit} that describes a marginal stability

state ($n = 0$) can be found. For this simulation the physical parameters are the following:

$$\begin{cases} L_x = 2\pi \\ L_y = 4\pi \\ \rho_h = 3 \text{ kg.m}^{-3} \\ \rho_l = 1 \text{ kg.m}^{-3} \\ \sigma_{crit} = 0.1 \text{ N.m}^{-1} \end{cases} \tag{6.16}$$

The interface is placed in the middle of the computational domain; periodic boundary conditions are placed on the vertical walls, slip conditions on the horizontal ones. The initial amplitude of the perturbation is $a_0 = 1.e - 8$ m. Nine different values of σ are treated, in the range $\sigma/\sigma_{crit} = [0.1, 0.2...0.9]$. The mesh is adaptive in all the simulations, as the results can immediately be compared with the analytical solution; for the value $\sigma/\sigma_{crit} = 0.5$ a convergence study has been realized, for both the uniform and adaptive mesh.

In general, the found values of n, presented in tables 6.9 and 6.10 and shown in figure 6.14, are very close to the predicted ones, in particular for high values of surface tension; however, in this region the computation is more difficult, as some stability issues appeared: the CFL condition had to be reduced when σ was closer to its critical value. As in other works, linear convergence is found for the uniform mesh test; with the adaptive mesh something in both errors and convergence rates is lost, remaining however close to the linear behaviour. The error never falls near or under the immediately coarser level.

Grid	n	err	q
32	1.498	0.053	-
64	1.525	0.026	1.033
128	1.541	0.010	1.397

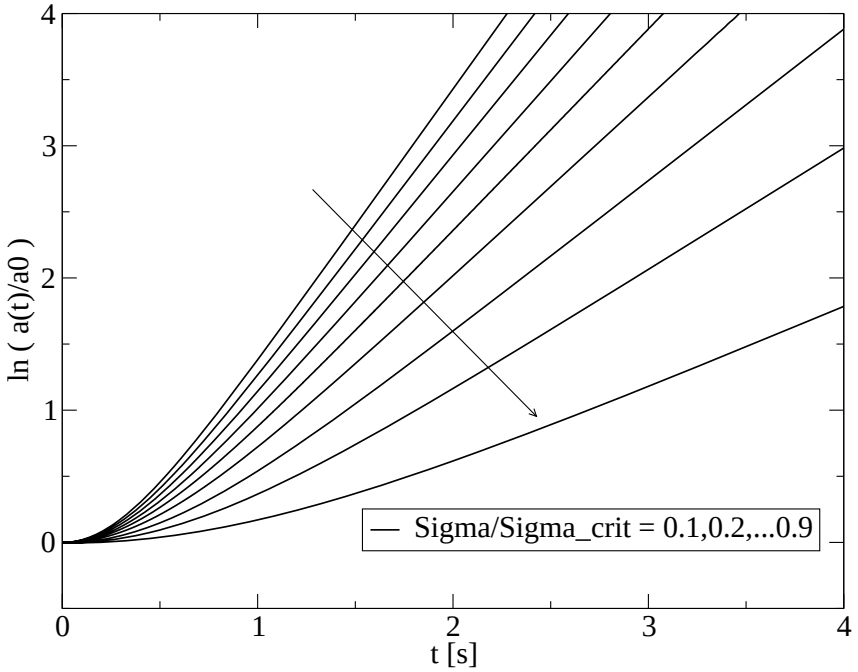
Table 6.9: Numerical computation of the growth rate for $\sigma/\sigma_{crit} = 0.5$: $n = 1.5508$, uniform mesh.

Grid	n	err	q
32	1.496	0.055	-
64	1.522	0.029	0.928
128	1.536	0.015	0.960

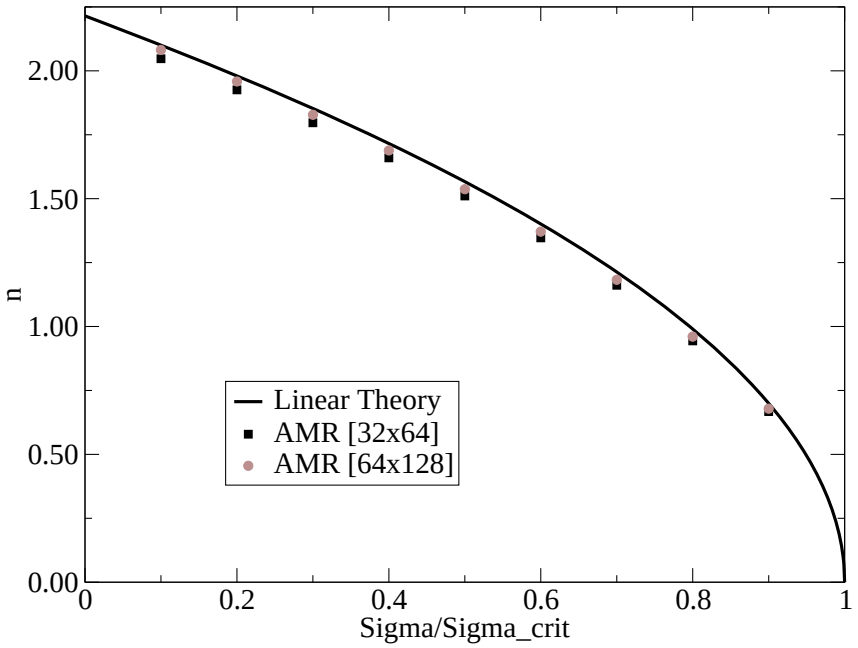
Table 6.10: Numerical computation of the growth rate for $\sigma/\sigma_{crit} = 0.5$: $n = 1.5508$, adaptive mesh.

Large amplitude, non-linear growth

In this section the evolution of the Rayleigh-Taylor instability is well outside the linear theory approximation limits. The simulation is performed in an extended domain with a height over width ratio of four. The interface is placed, as before, in the middle of the domain. The new



(a) Computed time histories of the initial deformation amplitude



(b) Computed growth rate for different values of σ and for two (adaptive) meshes

Figure 6.14: Results from the Rayleigh-Taylor instability computations.

computational parameters are:

$$\begin{cases} \rho_h = 1.225 \text{ kg.m}^{-3} \\ \rho_l = 0.1694 kg.m^{-3} \\ \nu_h = 3.13^{-3} \text{Pa.s} \\ \nu_l = 3.13^{-3} \text{Pa.s} \\ \sigma = 0 \text{ N.m}^{-1} \\ L_x = 1 \text{ m} \\ L_y = 4 \text{ m} \end{cases} \tag{6.17}$$

The new initial perturbation is $y = 0.05\cos(kx)$ with $k = 2\pi$. Those parameters are the same used in Popinet and Zaleski [88]: into his work a comparison is made between an Eulerian VOF method and a Lagrangian markers interface tracking method. In this simulation the interface sustains strong non linear deformations, the initial perturbation grows into a "mushroom" shape until 0.8 seconds. At this time the extremities begin to form two thin ligaments that quickly shrink to the grid cell size. With insufficient resolution the ligaments are broken and, subsequently, two drops are detached, so that the computed velocity field deviates quickly from the reference solution. The precise Lagrangian solution in Popinet and Zaleski [88] shows us that until $t = 0.9$ s the ligaments are kept. The first computation is performed with the same base resolution, 64×256 with four levels of AMR. At the last time step the ligaments are already somewhat fragmented, even if the shape remains. The resolution has been subsequently improved by adding an AMR level, reaching a fine resolution of 128×512: in this simulation the ligaments are clearly maintained, showing the improvement on the two phase code by a local refinement over the interface. The two solutions are shown in figures 6.15 and 6.16.

3.4 Planar rising bubble dynamics

In this section the dynamics of a planar two dimensional low density bubble immersed into a heavier fluid and subjected to gravity are investigated. If the other two phase flow tests were more oriented towards the correct reproduction of well-aimed physical behaviours, this one concerns the problem of the accuracy of the resolution of the governing equations. As there are no analytical solutions for this test case, the most common approach to validate interfacial flow codes is to examine the "picture norm". That is, to qualitatively compare the computed interface shape with reference solutions, which may come from simplified analytical expressions, others numerical simulations or experimental studies (Bhaga and Weber [10], Clift et al [17]). In this sense extensive research work has been done in creating quantitative *benchmarks* for the rising bubble problem, mainly from Sussman et al [114] and Hysing et al [44]. Some direct topological quantities are proposed, such as interface position and deformation, and also indirect ones, such as velocity. These are computed for increasing refined meshes; in order to allow a direct evaluation of the ability of the code to converge towards an accurate solution of the equations. The results obtained with the AMR code are here compared with the Hysing's simulations.

The simulation configuration

The initial configuration consists into a column of fluid, no slip conditions imposed on the horizontal surfaces and slip conditions on the vertical ones. In the lower half of the $[1 \times 2]$ domain, centered at $[0.5, 0.5]$ is the circular bubble of radius $r = 0.25$ composed of a lighter fluid ($\rho_2 < \rho_1$), all the velocities initialized to zero. The subscript 1 refers to the surrounding heavier fluid, while the subscript 2 refers to the interior of the lighter bubble (figure 6.17). The quantities are presented into a non dimensional form to assist with classification of simulations, obtained by scaling lengths and times by characteristics quantities such as the length $L = 2r_0$ and the

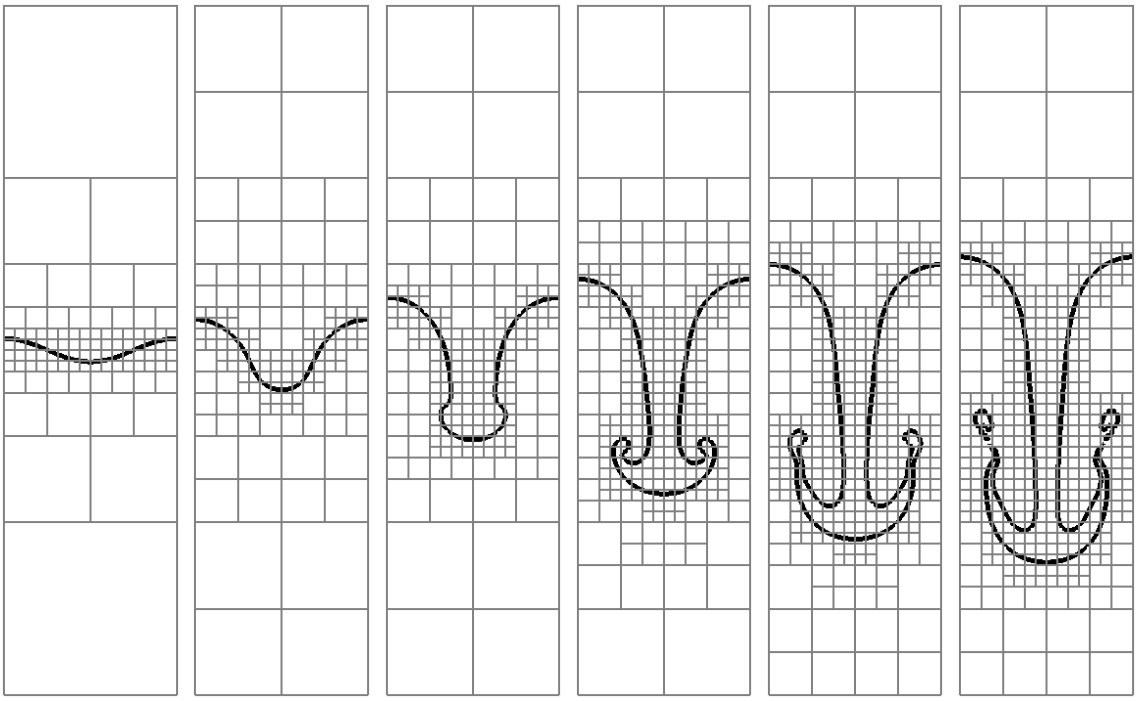


Figure 6.15: Results from the Rayleigh-Taylor instability computations, four levels of mesh, finest grid equivalent to 64x256, $t = 0.9$ s.

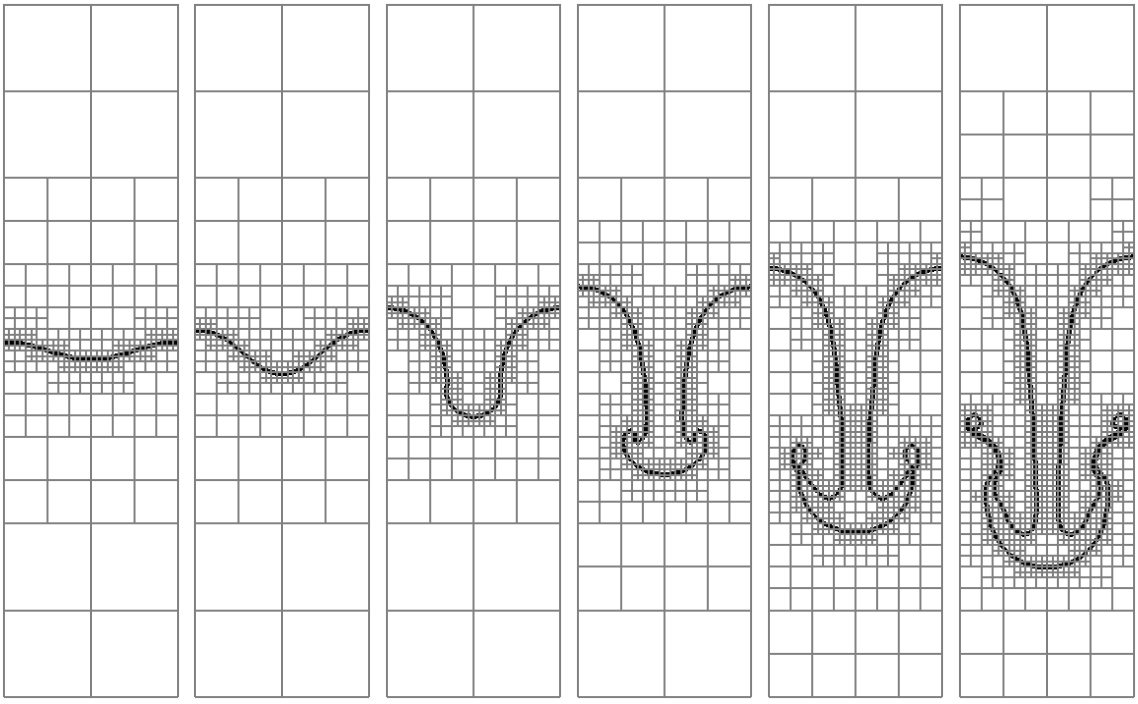


Figure 6.16: Results from the Rayleigh-Taylor instability computations, five levels of mesh, finest grid equivalent to 128x512, $t = 0.9$ s.

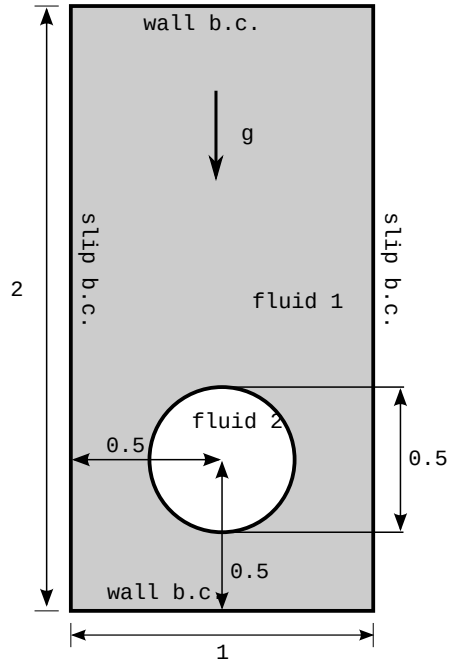


Figure 6.17: Initial condition for the rising bubble test and notations.

Test case	ρ_1	ρ_2	μ_1	μ_2	g	σ	Re	Eo	ρ_1/ρ_2	μ_1/μ_2
1	1000	100	10	1	0.98	24.5	35	10	10	10
2	1000	1	10	0.1	0.98	1.96	35	125	1000	100

Table 6.11: Physical parameters for the two test cases.

time L/U_g (where $U_g = \sqrt{g2r_0}$ represents the gravitation velocity). Two dimensionless numbers describe the regime of the bubble rise. These are the Reynolds and the Eötvös number. The first is the ratio between inertial and viscous effects, the second is the ratio between gravitational and surface tension effects:

$$Re = \frac{\rho_1 U_g L}{\mu_1} \quad , \quad Eo = \frac{\rho_1 U_g^2 L}{\sigma}$$

The viscosity and density ratios μ_1/μ_2 and ρ_1/ρ_2 help to fully classify the test cases. As in the Hysing's *benchmarks*, computations are performed for two configurations, presented in table 6.11. In the first case the density and viscosity ratio are equal to 10: the surface tension should be enough to hold the bubble together, and the final shape should be of an ellipsoidal regime. The second test is more challenging, as the density ratio reaches 1000 and the viscosity ratio 100; this bubble lies somewhere between the skirted and dimpled ellipsoidal-cap regimes indicating that break up can possibly occur (Clift et al [17]).

The evolution of the two bubbles is tracked for three time units, during which three quantities are measured. The first is the centroid of mass (x_c, y_c), of which, given the symmetry of the problem, the y coordinate only is considered:

$$y_c = \frac{\int_{\Omega_2} y \, dy}{\int_{\Omega_2} dy} \tag{6.18}$$

Ω_2 denotes the region which the bubble occupies. The second is the instantaneous rise velocity

(u, v), again y component only:

$$v_{rise} = \frac{\int_{\Omega_2} v \, dy}{\int_{\Omega_2} dy} \tag{6.19}$$

This is an interesting quantity to compute, because it does not only measure how the interface tracking algorithm behaves, but also it gives an idea of the quality of the overall solution; this means a check to the reduction of precision of the AMR grid far from the interface, and its influence on the interface itself. The last quantity is the *circularity* introduced by Wadell [128], an index of how much the shape of the bubble differs from the area equivalent circle:

$$c = \frac{\pi d_a}{P_b} \tag{6.20}$$

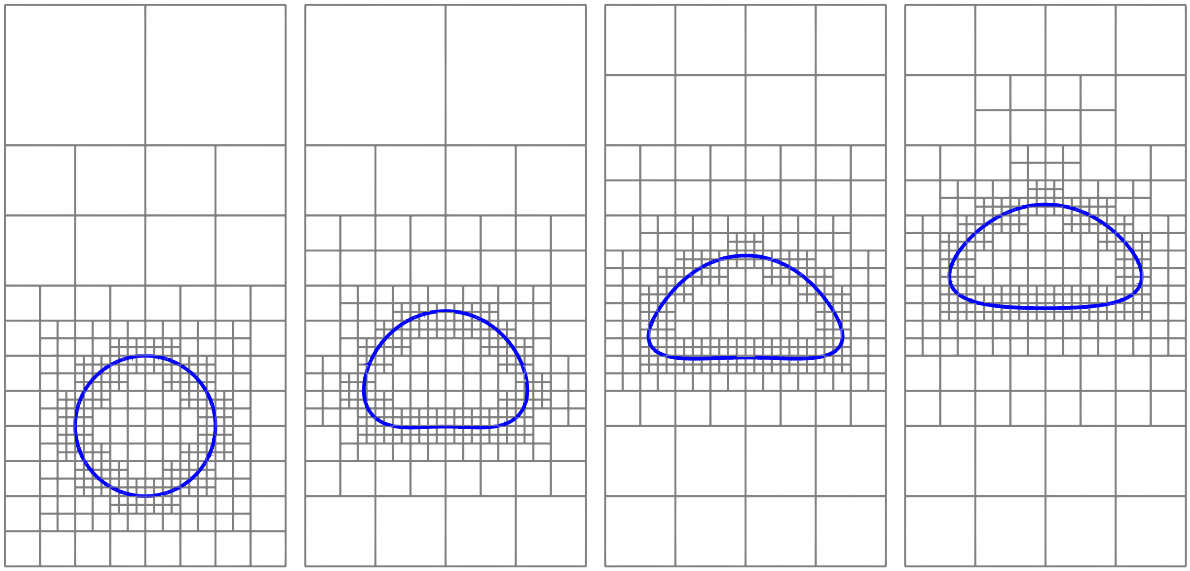
Here the numerator is the perimeter or circumference of a circle with diameter d_a, which has an area equal to that of a bubble with perimeter P_b. It starts from the maximum value of one, an then decreases as the bubble deforms. The mesh is, for all the simulations, adaptive and composed by blocks with 4×4 cells. A reference solution is computed on an high resolution uniform mesh of 256×1024. In figure 6.18 and 6.19 the shapes of the two bubble are shown at regular time intervals; the tables 6.12 and 6.13 contain the quantitative results obtained by a comparison with the reference computation. The difference between the two is calculated at the final timestep: $\text{err} = |q_{t=3} - q_{ref,t=3}|$.

Numerical results

	Mesh	value	err	q
y_c	32	1.0894	4.03e-3	-
	64	1.0863	9.39e-4	2.11
	128	1.0856	2.39e-4	1.97
	ref	1.0854	-	-
c	32	0.9313	1.02e-2	-
	64	0.9233	2.16e-3	2.23
	128	0.9206	5.17e-4	2.06
	ref	0.9212	-	-
v_{rise}	32	0.1986	2.86e-3	-
	64	0.1964	6.99e-4	2.03
	128	0.1955	1.57e-4	2.15
	ref	0.5917	-	-

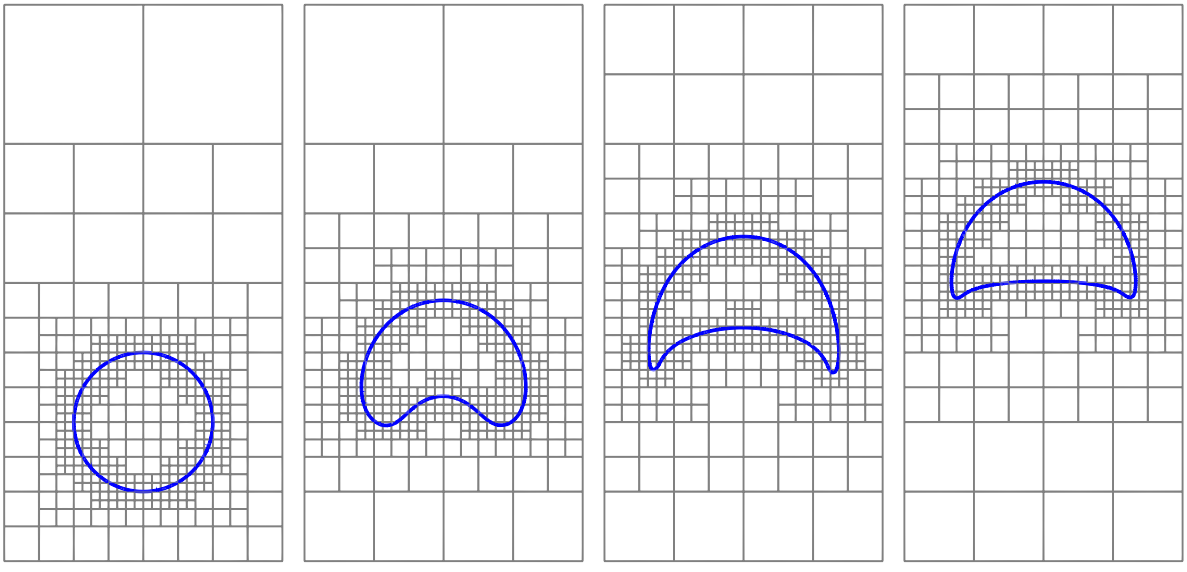
Table 6.12: Computed values for the rising bubble, test 1. Mesh value refers to the finest level. Errors and convergence evaluated through Richardson study.

In the first simulation set the bubble, being initially circular, begins to stretch horizontally. At fist it develops a dimple, but after a time it assumes a more stable ellipsoidal shape, as it can be seen in figure 6.18. In the second test case, shown in figure 6.19, the decrease of the surface tension causes the bubble to assume a more convex shape and to develop two sharp corners. The Hysing's results show two thin ligaments developing from the corners, which eventually break off and generate small satellite droplets. However, no ligaments or drops appear in the simulations here presented. The position of the center of mass for the fist bubble (figure 6.20a) is almost



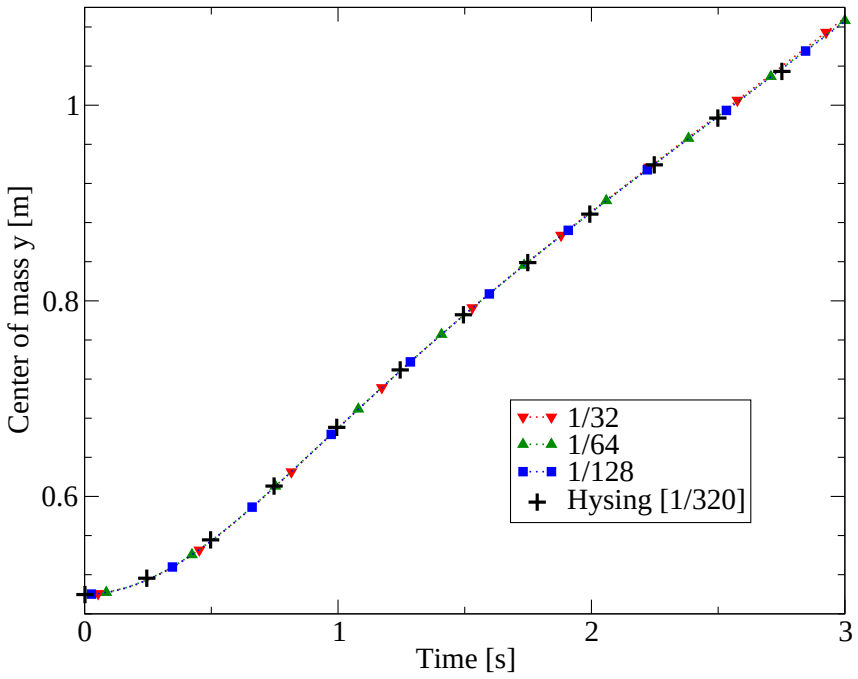
(a) $t = 0$ (b) $t = 1$ (c) $t = 2$ (d) $t = 3$

Figure 6.18: Results from the rising bubble computations, test 1, four levels of mesh, finest grid equivalent to 64×128, four different time units.

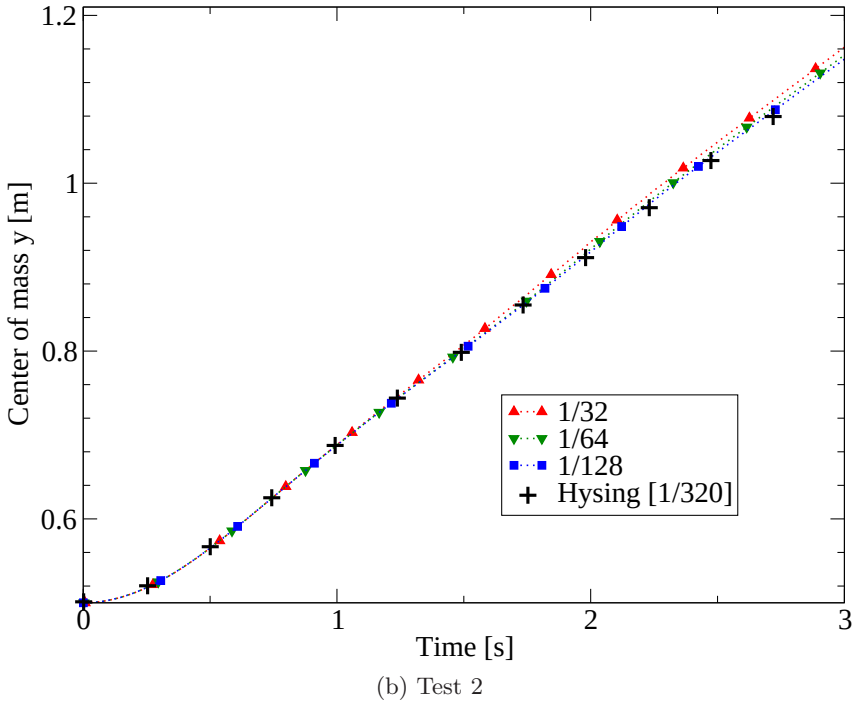


(a) $t = 0$ (b) $t = 1$ (c) $t = 2$ (d) $t = 3$

Figure 6.19: Results from the rising bubble computations, test 2, four levels of mesh, finest grid equivalent to 64×128, four different time units.

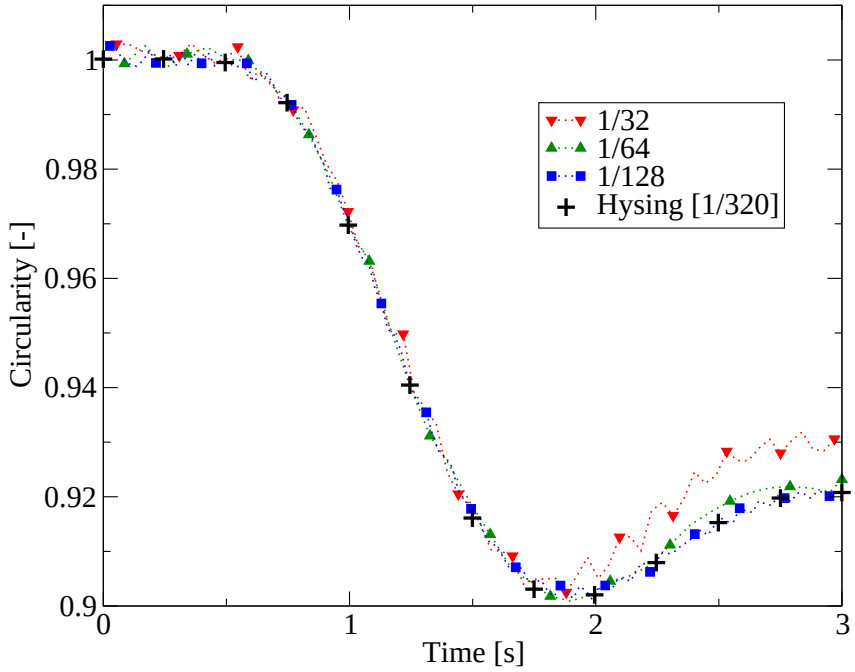


(a) Test 1

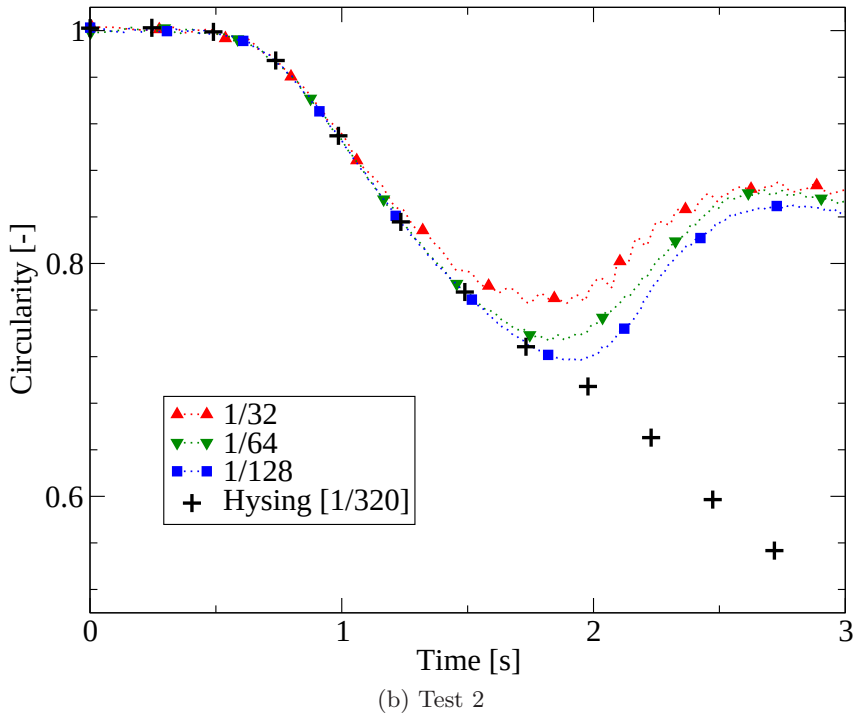


(b) Test 2

Figure 6.20: Comparison of the temporal evolution of the center of mass.

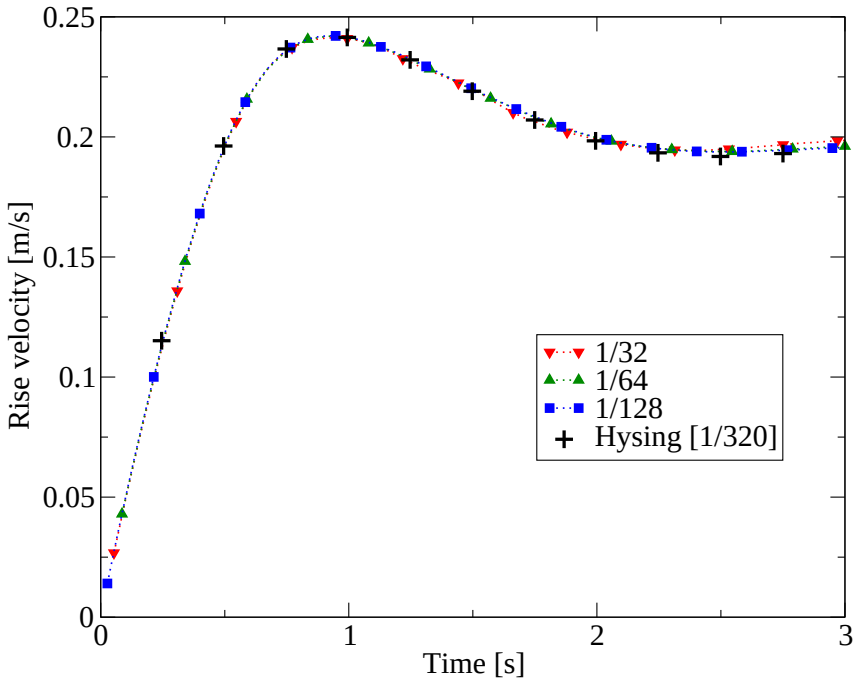


(a) Test 1

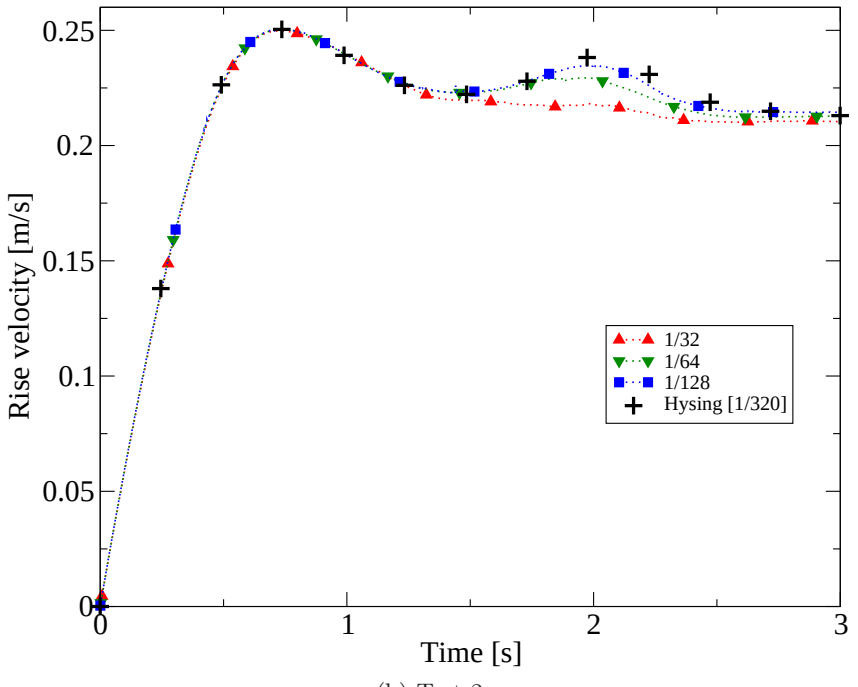


(b) Test 2

Figure 6.21: Comparison of the temporal evolution of the circularity.



(a) Test 1



(b) Test 2

Figure 6.22: Comparison of the temporal evolution of the rise velocity.

	Mesh	value	err	q
y_c	32	1.1619	1.86e-2	-
	64	1.1525	9.21e-3	1.02
	128	1.1477	4.42e-3	1.06
	ref	1.1434	-	-
c	32	0.8630*	2.41e-2	-
	64	0.8532*	1.43e-2	0.75
	128	0.8443*	5.38e-3	1.11
	ref	0.8389*	-	-
v_{rise}	32	0.2105	5.61e-3	-
	64	0.2125	3.59e-3	0.65
	128	0.2144	1.66e-3	1.11
	ref	0.2161	-	-

Table 6.13: Computed values for the rising bubble, test 2. Mesh value refers to the finest level. *Converged to the reference solution but not to the *benchmark* solution.

the same for all the meshes, and grows approximatively linearly in time; the convergence rate is quadratic for this value. The second bubble center (figure 6.20b) rise in the same manner, with a speed similar to the previous case. The difference between meshes is here more important, as it is possible to see in the plot the differences. However, the convergence is now at first order only. In the circularity computation of the test 1 (figure 6.21a) the coarser mesh shows some difficulty to follow the *benchmark* values. The finer mesh gives instead quite good results. Different is the test 2, where the code cannot track the circularity for the second test case after $t = 1.5$ s, approximately when the bubble forms the two sharp corners (figure 6.21b). It is in this location that Hysing could find, with a finite elements algorithm, the small ligaments that give birth to the small drops. The mean rising velocity is, as for the center of mass, very well captured for all the meshes in the test 1 (figure 6.22a). The convergence rates are, as for the other two quantities, quadratic. The test 2 results (figure 6.22b) show a marked approach to the reference solution as the resolution increases. The rise velocity seems a little bit underestimated, in the region near the second acceleration, after the initial velocity peak; the terminal velocity is however well estimated for all the grids. As for the other quantities, q keeps linear in this configuration.

4 Computational performances

Once the code has proven to be effective in the preservation of the fine grid accuracy in the battery of validation tests, still the main question remains, how many computational resources can be saved with the mesh adaptation and how the parallelization is effective. To give answers, some performance tests are proposed in this section. The first two tests are designed to check the reduction in computational time given by the AMR in the advection equation alone and in the full two phase Navier-Stokes solver. In particular, the difference between these two tests is the presence of the expensive elliptic solver, which can degrade the overall performances of the code.

For the AMR speed-up tests, the confrontation is done with the original DYJEAT code

of Couderc [18], configured as close as possible as the AMR code [1]. The main algorithmic difference between the two (apart from the AMR) is the elliptic solver: the first employs a multigrid preconditioned Conjugate Gradient, the second a FAC multigrid preconditioned Bi-Conjugate Gradient stabilized algorithm (described in the previous chapter). The computational cost of this second is almost the double of the first one; hopefully the reduction in the nodes can quickly overcome this weakness.

In order to give generality, the speed-up results are proposed in terms of a "Time per Cell" (T/C) or "Time per Cell per Iteration" (T/C/I) ratio. A subsequent ratio $Z = (T/C)_{AMR}/(T/C)_{UNI}$ gives an overall idea of the mean amount of work the solver does for each cell. Then, the ratio $N = C_{UNI}/C_{AMR}$ between the number of nodes of the two mesh configurations can tell when, given a value of Z, the AMR computation becomes advantageous. The answer, given the fact that a AMR node costs more than an equivalent single grid one (because of all the grid operations), is in the form "at equal finest resolution, of how many point the mesh should be reduced by the AMR to start gaining computational time?"

The section about parallel performances gives both the results from a strong and a weak scaling tests. The idea is to compare the results from different parallel run with different domain size and/or processor number. The results from the AMR code are expected a little bit worse than for a single grid solver, but the expectation are to still be able to partition and solve efficiently large domains. The parallel performances are checked on the full solver, with some hints on the impact of the multigrid preconditioner on the overall performances.

In all the tests the computation time only is measured using the MPI_WTIME() command, no initialization or checkpointing included. The compilation options are strictly the same for both codes. The computations in the AMR tests are performed on a single processor to isolate the speed-up given by the mesh refinement alone.

4.1 Adaptive mesh speed-up

Advection equation

In this section the AMR performance for the advection equation applied to Zalesak disk problem (cfr section 1.1) are presented. For the four different resolution the computational times and cells number are measured, and the mean time the code spent per cell is given. The AMR value is expected to be clearly higher because of the added AMR operations: guardcell filling, interpolation, grid modification, and the plain fragmentation of the data vectors in blocks. The disk is advected for a full rotation, the final computational time is kept for confrontation.

	Uniform			Adaptive				
Grid	T [s]	Cells	T/C [ms]	T [s]	Cells	T/C [ms]	N	Z
32	1.08	1024	1.05	4.06	832	4.88	1.23	4.63
64	6.61	4096	1.61	15.28	1888	8.09	2.17	5.02
128	46.93	16384	2.86	63.32	4672	13.6	3.51	4.73
256	365.00	65536	5.57	274.99	10240	26.9	6.40	4.82

Table 6.14: Measured times per cell for different uniform and adaptive meshes.

As it can be seen in table 6.14 and figure 6.23 , as more AMR levels are added, the gain in

[1] The same test could have been done using the AMR code set to use uniform meshes, but the real test is to confront with the standard "lighter" non AMR code.

cells becomes more evident in the ratio $N = C_{UNI}/C_{AMR}$. The ratio Z between the two times per cell, AMR over uniform, give an idea of the time per cell increase: about five times more for the AMR. In this test case the regridding operations happens very frequently, causing the AMR time per cell to be quite high. This explain how the AMR total time becomes smaller only in the most refined case when the reduction in points ($N = 6.4$) exceeds the AMR load on computational time ($Z = 4.8$), as shown in figure. The constant value of Z shows how the grid operations do not become more expensive in large scal simulations. The overall result is that a reduction of 75% of nodes allows the AMR computation to gain an edge over the uniform mesh one.

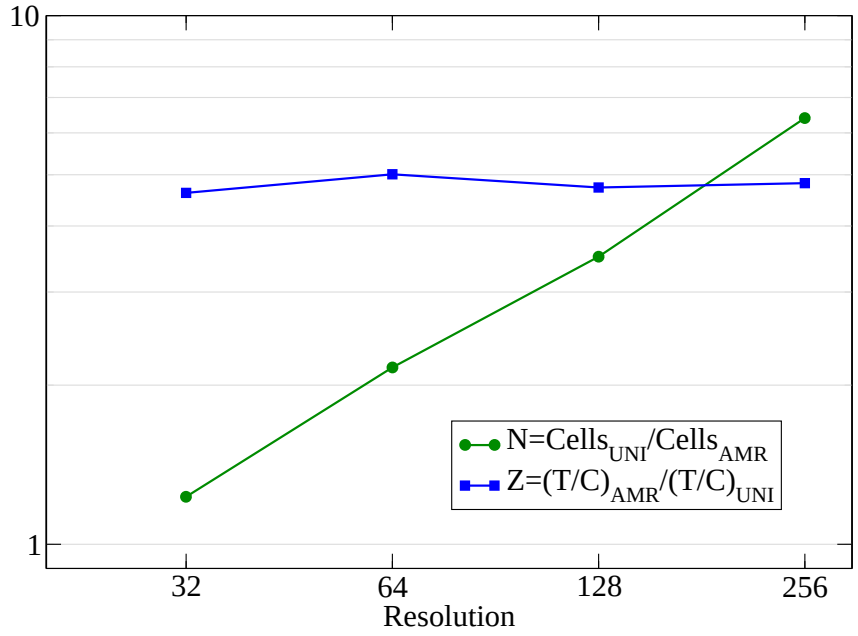
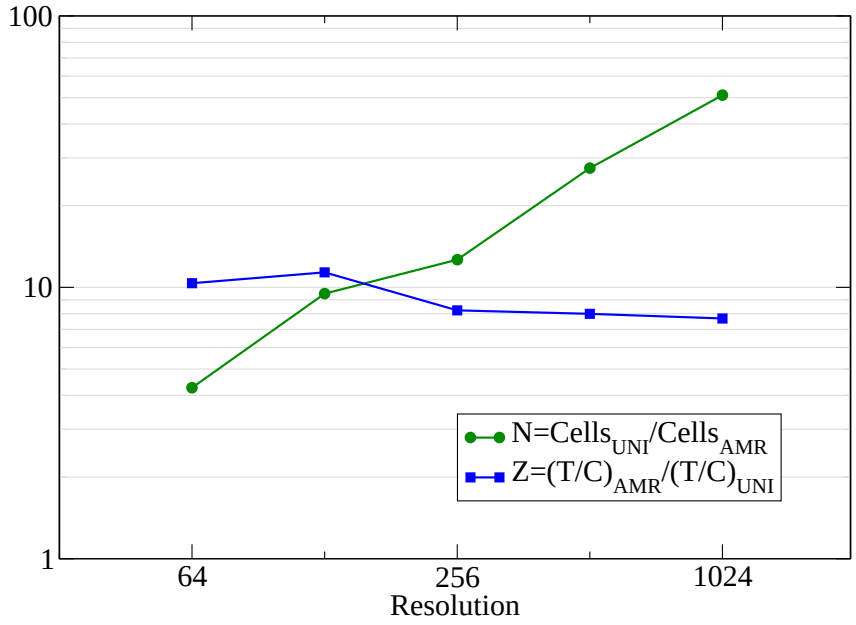


Figure 6.23: Plot of the cells ratio N and time ratio Z against the resolution for the Zalesak disk computation. The crossing point is where the AMR computation becomes advantageous, $T_{AMR} < T_{UNI}$.

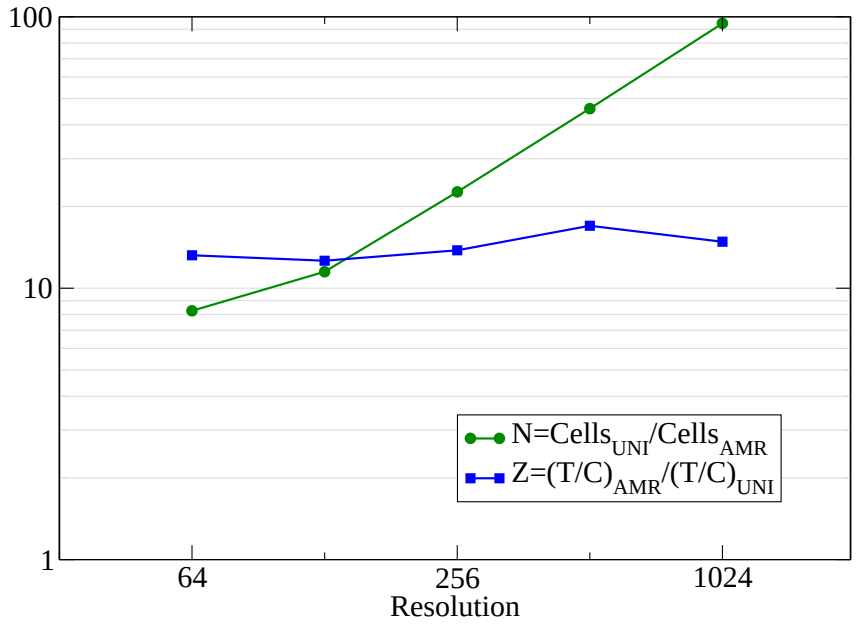
Two phase Navier-Stokes

In this section the performances of the full two phase Navier-Stokes solver are checked in terms of computational time. The comparison is done at equal mesh size for the finest AMR mesh. The proposed test case is the Rayleigh-Taylor non linear instability simulation (cfr section 3.3), in which it is quite easy to isolate a region to be refined (the interface). The other characteristic of this case is the frequency of the regridding, not as frequent as for the Zalesak disk advection test but still significant. As the resolutions considered are quite high, the simulations have run for a fixed number of iterations (1000), so that the value of mean time per cell (T/C) is obtained by dividing the total computational time by this value. Another aspect taken in account in this test is the size of the blocks in term of nodes. The block based quad-tree imposes a fixed size of the sub grids. The smaller are the blocks, the easier is to refine a precise region (the cell based quad-tree, being a degeneration of the block based where a block is a cell, excels in this). However, at the same time the larger is the total number of blocks to be treated, each of them with its own set of guardcells (three on each boundary node for the WENO scheme); also, the solution vector becomes more fragmented in the physical memory, reducing the efficiency of the data processing. An added positive consequence of the reduced block size is that the coarsest

block correspond to the bottom grid of the multigrid: at the same fine resolution, a smaller block size adds a multigrid level, with possible convergence benefit for the elliptic solver if a small number of levels is used.. The blocks tested here have 4×4 and 8×8 cells. The smaller block size has been used in most part of the validation tests in order to maximize the adaptation of the mesh; the larger one seems more equilibrate and is used in the main application in the next chapter.



(a)



(b)

Figure 6.24: Plot of the cells ratio N and time ratio Z against the resolution for the Rayleigh-Taylor disk computation. (a) 8×8 cells per block (b) 4×4 cells per block.

In table 6.15 and figure 6.24a the results from the 8×8 blocks are presented. The time

per cell per iteration for the uniform code is almost constant for all the meshes, with a small increase only for the finest mesh; the same value for the AMR code is about eight times higher for the finer resolutions (a quite high value, it is four for the advection equation test, where there is a lot more regridding). The same results for the smaller blocks are similar (table 6.16

	Uniform			Adaptive				
Grid	Cells	T/I [s]	T/C/I [ms]	Cells	T/I [s]	T/C/I [ms]	N	Z
64	16384	0.14	8.83e-3	3840	0.35	9.15e-2	4.3	10.4
128	65536	0.56	8.58e-3	6912	0.67	9.75e-2	9.5	11.4
256	262144	2.28	8.68e-3	20736	1.48	7.14e-2	12.6	8.2
512	1048576	9.25	8.82e-3	38144	2.69	7.05e-2	27.5	8.0
1024	4194304	39.14	9.33e-3	82176	5.89	7.16e-2	51.0	7.7

Table 6.15: Computational times for the two codes for the presented test case, same interface resolution, 8×8 cells per block.

and figure 6.24b), but not surprisingly the time per cell is a little bit higher than in the previous one. On the other hand, the higher values of N reveal that the overtake of the gain threshold can be achieved earlier in the mesh refinement. The solution is probably better in the first case, however, as the finer mesh is more extended. It is also true that problems with very difficult elliptic solve may shift upwards the Z ratio, rendering the AMR computation less favourable.

	Uniform			Adaptive				
Grid	Cells	T/I [s]	T/C/I [ms]	Cells	T/I [s]	T/C/I [ms]	N	Z
64	16384	0.145	8.83e-3	1984	0.23	1.16e-1	8.3	13.2
128	65536	0.563	8.58e-3	5696	0.61	1.08e-1	11.5	12.6
256	262144	2.276	8.68e-3	11584	1.38	1.19e-1	22.6	13.8
512	1048576	9.250	8.82e-3	22848	3.42	1.49e-1	45.9	17.0
1024	4194304	39.146	9.33e-3	44352	6.14	1.38e-1	94.6	14.8

Table 6.16: Computational times for the two codes for the presented test case, same interface resolution, 4×4 cells per block.

As conclusion, a reduction of eight times the computational nodes is something not trivial, but it can be achieved with modest efforts in higher resolution computations, especially in three dimensions. The AMR technique here presented is definitely worth for large scale simulations in term of computational time. This can be stated also by remembering the saving on processors in parallel computations, as the eight times reduction in nodes means also a reduction of eight times the cores needed for the same computation (if the parallelization has the same efficacy in both cases), allowing ideally eight simulations to run at the same time (as for a parametric study).

4.2 Parallel performances

In the subsequent section the parallel performances of the code are tested. The two types of parallel study are realized, the strong and the weak scaling. In the first the domain is fixed, so that the maximum number of points a single processor can handle are given; then some computations are performed with an increasing number of processors. The ideal reduction of time, that would be achieved if the communication time were zero, is equal to the number of processors, as the domain become more partitioned among the CPUs. In the second case the domain addressed to a single processor is fixed; the dimension of the problem grows together with the number of cores. The expected computation time should be almost constant.

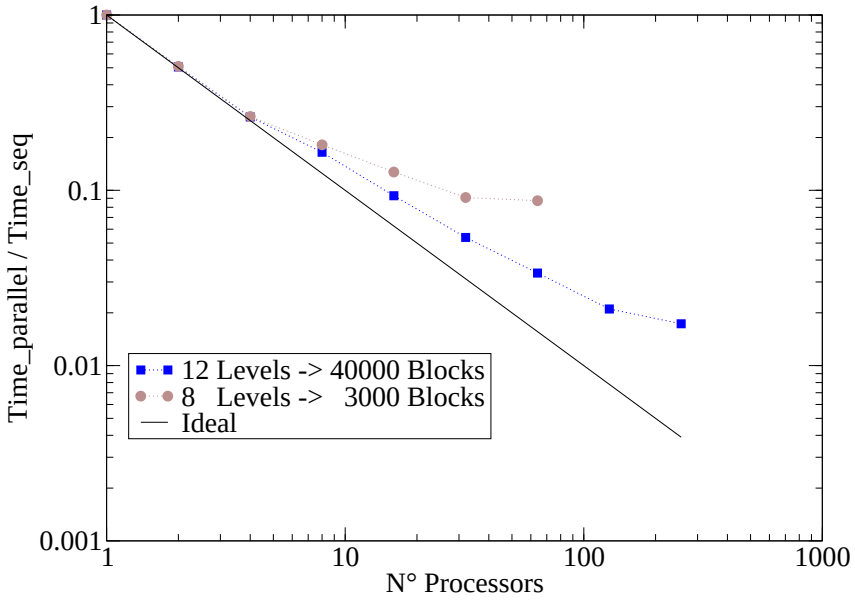
Strong scaling

For the strong scaling test a square domain composed by 12 levels of an adaptive mesh has been built (the chosen test case is the damped surface wave oscillation, cfr 3.2). Each block is composed by 8^2 internal points plus 132 ghost points (three cells on each direction, corners included). When run on a single processor, this mesh requires an allocation of at least 40000 blocks to build the whole tree (actually not all the blocks are active, but PARAMESH needs some room for moving data). The resulting fine resolution is 16384^2. The computational domain is then distributed among an increasing number of processors, from 2 up to 256, each time their number doubled. The computational time is measured for a fixed number of iterations; the mean iteration time is kept. For each processor number increase, the maximum allocated block are accordingly reduced (approximately divided by two, figure 6.25b). The test has been repeated with a maximum level of 8 (resolution 1024, 3000 initial blocks) in order to compare the limit value of points per processor which stops the speed-up.

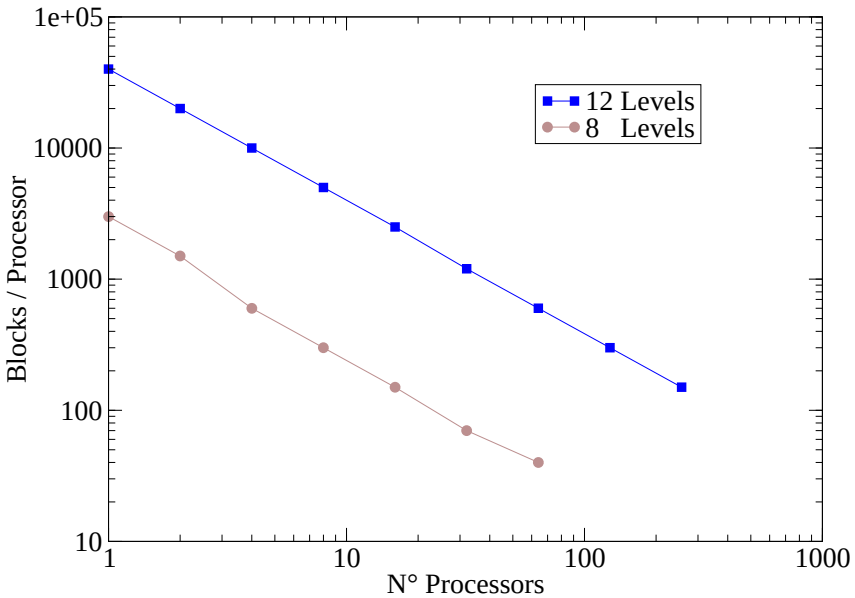
The results in figure 6.25a show a good constant decrement in the computational time, even if not the ideal one. The speed-up limit seems reached with 256 processors; the time still decreases from the 128 test, but quite far from the ideal division by two. This result can be expected, as in the AMR code the communication is more complicated than an uniform mesh code alone, as the quantity of exchanged data may vary in function of the mesh configuration. Moreover, a particular "idle processors" problem arises from the multigrid preconditioning: a more precise evaluation of this issue is given afterwards. As expected, in the 8 levels test the limit is reached sooner, at 32 processors. The two results agree reasonably well in the limit number of blocks per processor: if the last point of the first test and the next-to-last of the second one are kept as limits, the number of blocks reach its limit at around 100 blocks, corresponding to 6400 points. If the next-to-last point is kept for the fist case, this values jumps to around 300 blocks or 19200 points. This is an interesting value because in the experience of Couderc [18] the limits of the sub domains size is reached with 128^2 mesh points per processor, corresponding to 16384 points.

Weak scaling

In this section the size of the domain assigned to a processor is kept constant at 200 blocks, the number retained from the previous tests. The resolution is increased together with the number of processors working. The mesh is kept uniform, so that adding a refinement levels provokes four children to be spawn over each old leaf block: the active points are multiplied by four and so are the processors. The behaviour of the elliptic solver could change with the refinement level, so the iterations instead of the precision are fixed. The same test has been repeated with the adaptive mesh, thus losing the correspondence between the number of points and processors increase. Results from this test are shown in figure 6.26. The time per iteration is not really constant: the time per iteration rises slightly, from about 0.4 to 1 s approximately for respectively 1 and 256 processors. It has to be pointed out, however, that the multigrid



(a) Speed-up ratio for the two test cases



(b) Corresponding blocks per processor allocated. Each block contains 64 active points

Figure 6.25: Strong scaling speed-up test.

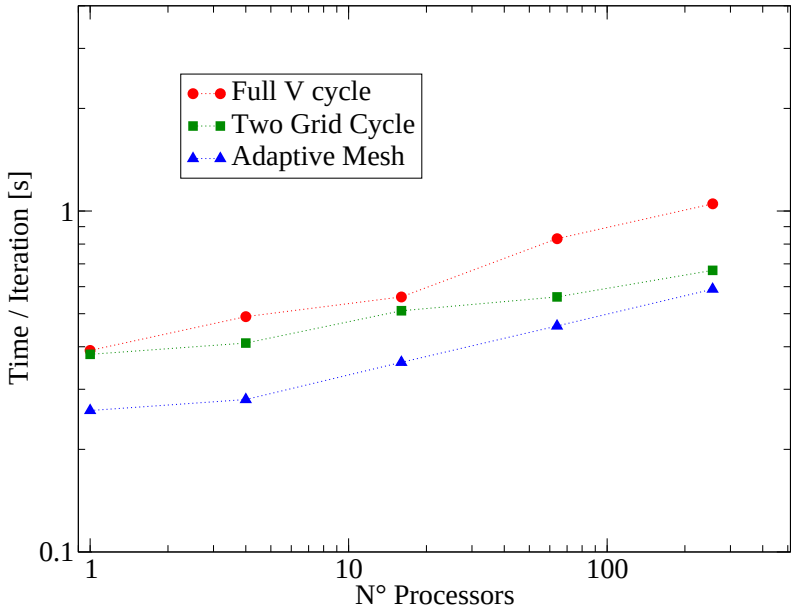


Figure 6.26: Weak scaling tests. The first two curves refer to an uniform mesh computation; the last one to a fully adaptive mesh.

algorithm utilizes more levels (so more points) at the finer resolutions. The use of the adaptive mesh does not change the general behaviour, but the computational time obviously decreases. Finally, the multigrid full V-cycle has been replaced by a two grid cycle, in order to evaluate the

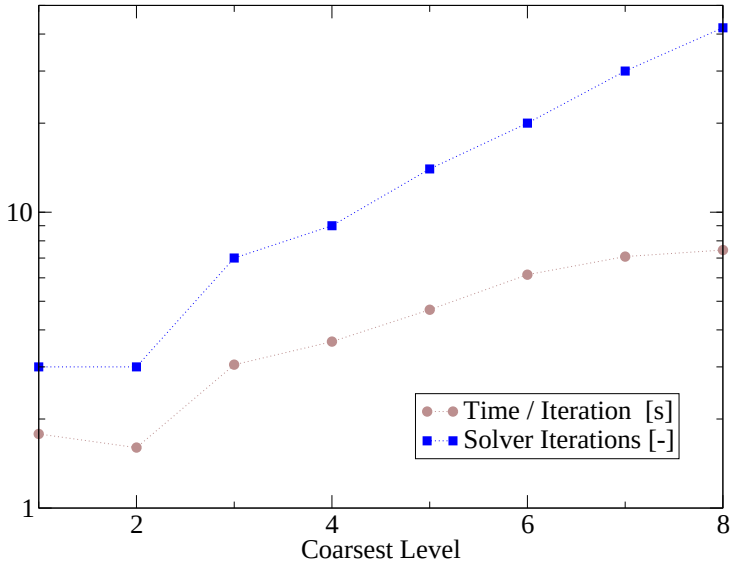


Figure 6.27: Multigrid coarsest level test: augmentation of iterations and computational time due to the decreasing of total multigrid levels.

loss of speed-up caused by the multigrid. This is caused by the PARAMESH peculiar utilization of the AMR tree for the multigrid: for each descent step the hidden parent block are defined as active blocks and used for the relaxation; however, at the coarser levels the number of active

blocks approaches and eventually surpasses the number of processors, making an increasing number of those to wait. As can be seen in the graph, the computational time is now much more constant. This result confirms the limits of the preconditioner scalability. One obvious solution would be to stop the descent to a finer multigrid level. However, this would reduce the multigrid efficiency, as it depends on the number of levels employed. A simple test has been done to check the influence of the coarsest level on the convergence speed: a computation with a fixed precision is performed with an increasing coarsest level, which corresponds to a decrease of the total number of multigrid levels. The results in figure 6.27 show how the loss of even a single level may raise the number of global solver iterations, and consequently increase dramatically the computational time. This block based multigrid characteristic may even be an advantage, because usual fixed mesh codes are limited in their coarsest grid by the mapping of the domain in sub-domains.

Conclusion

In this chapter a set of validation tests have been performed to evaluate the accuracy and performances of the code. The tests cover the resolution of the linear transport equation, the single phase Navier-Stokes solver, the full two phase solver and check the speed up given by the AMR and the parallelization. The code has proven capable to maintain the fine grid accuracy in all tests, as well as the convergence rates with negligible losses. The speed-up given by AMR shows gain in time respect to the original single grid solver once reduced by four times the number of nodes in the transport equation, and about eight times in the two phase solver; these results are, however, quite sensitive to the performances of the elliptic solver alone. The parallel performances show an overall efficient parallelization, but they are similarly limited by some design weaknesses of the multigrid preconditioner.

Chapter 7

Primary Atomization of a Liquid Sheet: Simulations with AMR

Contents

Introduction

This last chapter focuses on the application of the AMR code to the problem of the primary atomization of a two dimensional liquid sheet. This series of numerical simulations follow the path described in Couderc [18], Couderc and Estivalezes [19], where a planar configuration of the sheet has been considered to reproduce the initial two dimensional aspects of the instability, supported by the large amount of experimental results at disposal in literature. It is clear that the primary atomization is globally a three dimensional mechanism which involves instability in the two planes, the longitudinal and the transversal one. It is on the other hand known that the starting oscillation evolves in the flow direction, characteristics which allow a preliminary study in two dimensions alone. In a second time the transversal instability start to generate the longitudinal ligaments.

The overall goal of this thesis is to develop a numerical tool which allows the accurate representation of this phenomenon, that presents itself as definitely multi-scale. The dimensions of an injector can be measured in tens of millimetres; the amplitude of the injection vein only about one tenth of a millimetre. If the complexity of the interfacial flow has to be represented in a direct numerical simulation, it is clear that an accurate meshing of the sheet would not allow the

representation of a realistic domain. It seems from Couderc [18] that the consideration of a reduced computational domain does not worsen the capturing of the global oscillation frequencies; on the other hand the simulation of the whole atomization requires a domain which expands quickly in time. If all those considerations are valid in two dimension, in three dimensions they become even more true.

The numerical simulation in this chapter are aimed to find the values of global oscillation frequency for a low resolution sheet, using the adaptive mesh to reduce the time and processors resources. At the same time this can be considered the final validation test of the modified code, a definitive confirmation of the ability of a composite adaptive mesh to correctly capture the phenomenon without any physical "deviation" given by the under resolution of the flow far from the interface. In the second part a very high resolution sheet simulation is performed to overcome the limitations coming from the weak resolutions of the previous case, and a first full two dimensional atomization (sheet oscillation, ligaments break-up, droplet generation), with realistic parameters, is observable.

1 Sheet configuration

The configuration of the liquid sheet has greatly improved the ability to observe, measure and comprehend the physical mechanisms of the disintegration. Being the width of the sheet largely superior to its thickness, the initial configuration can be considered two dimensional near to the injector zone. For this reason a certain number of bi-dimensional stability studies have been performed. These studies can be quickly become too complex to be solved, and may be obliged to neglect viscosity and the non linearity of the waves evolution. The numerical tool can in this case reveals all its capabilities in the study of the instabilities.

The main interest in the simulation of a two dimensional sheet is the possibility to isolate the beginning of the destabilization of the initial configuration, being this phenomenon longitudinal only. These simulations put in evidence the formation of sinusoidal deformation waves without any interference coming from three dimensional phenomena. The numerical study allows the realization of temporal simulations as well as spatial ones: the first follow the temporal evolution of a single wavelength of a deformation wave, while the second ones try to represent a real injector situation. These latter are obviously more interesting because nearer to experiments, but they are also more difficult to realize because of the transition time in which the different instabilities start to develop, as well as for the necessity to represent wider computational domains and to impose boundary conditions which can be problematic.

The configuration of the sheet is retrieved from Couderc [18], in order to perform a final validation of the code face to a complex unsteady computation as the atomization is. Two of the most relevant results in his work were the construction of a diagram like the one established by Mansour and Chigier [59] or Lozano et al [56], where the global oscillation frequencies coming from their experiences are allocated in the three zones representing three different atomization regimes, and the diagram which shows the linear relation between the frequency and the gas velocity (depicted again in figure 7.1).

1.1 Computational domain

This set of simulations takes place into a square domain, where injection conditions (fixed $u(y)$ velocity profiles centered on the injection zones, wall everywhere else) are assigned to the left wall and outflow condition on the other walls (figure 7.2).

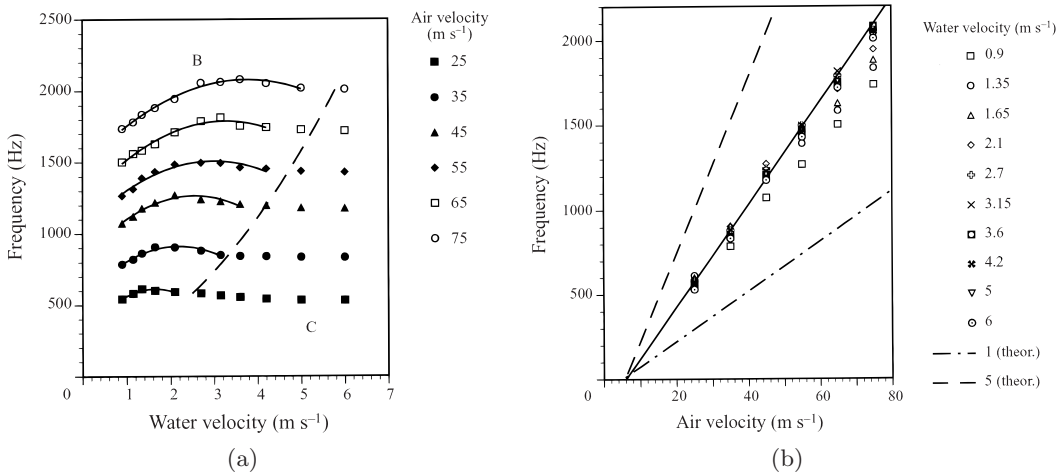


Figure 7.1: Results from Lozano et al [56] (in Couderc [18], Fernandez [31], Trontin [121]) representing the global oscillation frequency versus the liquid injection speed, for different air pressures. (a) Frequency against air velocity for different liquid velocities (b) Frequency against liquid velocity for different air velocities.

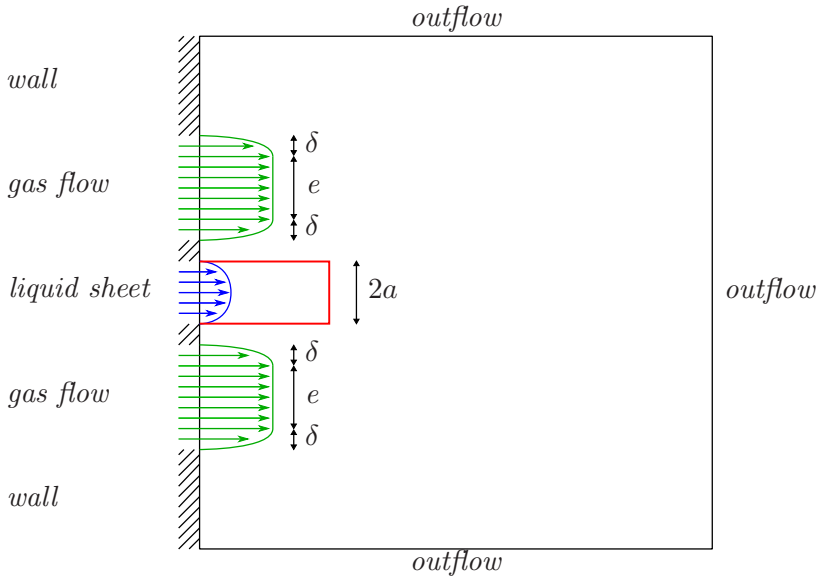


Figure 7.2: Configuration of the sheet simulation, recalled from Couderc [18].

The velocity profiles

The velocity profiles for the injection have been chosen to recall the real atomizers geometries. The liquid is injected through a 300 μm to 1 mm thick fence, which usually spans hundreds of times these values in order to minimize the boundary effects. Given a typical Reynolds number always inferior to 1000 (based on the boundary layer thickness), the liquid flow can be approximated by a planar laminar Poiseuille profile:

$$u(y) = \frac{3U_l}{2a^2}\left(a^2 - y^2\right) \tag{7.1}$$

where a is the sheet half amplitude and U_l the mean incoming flow velocity this ranging from 1 to 3 m.s^{-1}. The velocity profile for the gas is a bit more tricky, because different gas injection

mechanisms and different regimes have been employed in literature, not many informations available on the shape of the boundary layer. If the gas vein thickness is of the same order of magnitude of the sheet, the gas regime may range from a Poiseuille to a fully developed turbulent flow. Even if the turbulence is not considered in these simulations, the gas velocity profile is finally a turbulent profile approximated by the Polhausen fourth order polynomial solution

$$\frac{u(y)}{U_g} = 2\frac{y}{\delta} - 2\left(\frac{y}{\delta}\right)^2 + \left(\frac{y}{\delta}\right)^4 \tag{7.2}$$

where δ is the boundary layer thickness and U_g the undisturbed velocity, in the simulations presented in this section ranging from 15 to 40 m.s^{-1}.

1.2 Solver restrictions

In Couderc [18] at least two main problems arise in the set up of the simulation, especially when trying to simulate air/water flows. The first is bound to the velocity range to be treated. The experiences employ velocities up to 80 m.s^{-1}, corresponding to Reynolds numbers up to 3000 (always calculated with the thickness of the boundary layer). So much high values describe a convection dominated flow, its complexity boosted by the non linearity of the Navier-Stokes equations. The DNS becomes in this case quickly hampered by small cell sizes, which decrease even more for the needs of the interface transport. The outflow boundary conditions pose another problem of stability when confronted to this situation. For these reasons the Reynold number has been lowered artificially by an arbitrary increase of the fluids viscosities.

The second problem concerns the elliptic equation for the pressure computation. A 1/1000 ratio determines a much more complex interfacial evolution, and may hamper the code stability and the mass conservation in the Level Set advection. Moreover, the difference of fluids densities transfers to the system matrix coefficients, resulting in a dramatic increase of the conditioning number as the two densities start to diverge (an illustration of this result can be seen in the chapter 5). This increase stacks with the increase given by the non symmetry of the matrix in the adaptive mesh computations, so that this problem become even worse.

2 Low resolution simulations

The first simulations are performed with a relatively low resolution and few levels of refinement in order to speed-up the computation and allow a set of measurements with different injection parameters. The full capability of the AMR are exploited later with some higher resolution simulation, at first with a double resolution in a wider domain, in order to catch any difference in the predicted oscillation frequency and to take a glance to what happens outside the computational domain (thus effectively checking the effects of the boundary conditions). It is reported in Couderc [18] that even at the low resolutions the frequencies are correctly captured by the solver.

2.1 Simulation parameters

The retained parameters for this simulations are, after the solver limitations considerations, are reported in table 7.1. The air density represents a 10 bar pressurized air. This implies an increased momentum compared to the atmospheric conditions, which results in a more rapid manifestation of the oscillation, and a smaller break-up length. The dynamic viscosities have been risen as told in the solver limitations analysis. Still a comparison with the experience can

Physical parameters	Dimensions
$\rho_g = 12.26$ kg.m^{-3}	$L_x = 3 \times 10^{-3}$ m
$\rho_l = 1000$ kg.m^{-3}	$L_y = 3 \times 10^{-3}$ m
$\mu_g = 1.78 \times 10^{-3}$ Pa.s	$a = 150 \times 10^{-6}$ m
$\mu_l = 1.132 \times 10^{-2}$ Pa.s	$\delta = 3 \times 10^{-4}$ m
$\sigma = 7.28$ N.m^{-2}	$e = 1.5 \times 10^{-4}$ m
$U_g = 30$ m.s^{-1}	$\Delta x_f \approx 11.72 \times 10^{-6}$ m
$U_l = 2$ m.s^{-1}	$\Delta x_c \approx 93.75 \times 10^{-6}$ m

Table 7.1: Parameters for the low resolution sheet simulation, resolution 256^2, four AMR levels.

be performed by using the non dimensional numbers:

$$\begin{cases} Re_l = \dfrac{\rho_l U_l a}{\mu_l} & = 26 \\ Re_g = \dfrac{\rho_g U_g \delta}{\mu_g} & = 62 \\ We = \dfrac{\rho_g (U_l - U_g)^2 a}{\sigma} & = 20 \\ M = \dfrac{\rho_g U_g^2}{\rho_l U_l^2} & = 2.76 \end{cases} \tag{7.3}$$

The initialization of the sheet can be performed more or less arbitrarily: the sheet can be set already present into the domain or, conversely, be injected during the computation by the boundary conditions. This latter is the preferred configuration, because the nodes are kept at the minimum by the adaptive mesh in this phase (the interface occupies a very small part of the domain), so that the computation is fast. The mesh consists in four levels of AMR, with 8×8 cells per block, and is depicted in figures 7.4.

2.2 The sheet instability

Even though no perturbation is given to the initial condition nor in the injection profiles (which are constant in time), the solution loses its horizontal symmetry very soon. The instability starts to manifest itself when the advancing sheet front has hardly reached half of the domain (figures 7.3). The first effect is the acceleration of the sheet under the shearing effect of the air flow, which provokes its stretching. Then the sheet starts waving inside and outside the air flow. A strong interaction between the movements of the high velocity gas veins and the slower liquid sheet can be seen, which causes the final regime of sinusoidal oscillation to establish. The instability is so strong that simple numerical errors are sufficient to destabilize the symmetry. The velocity profiles at the injection nodes do not respect the balance of forces on the interface, as it can be seen from the viscous stress forces evaluation:

$$\tau_p = \begin{cases} \mu_l \dfrac{\partial u}{\partial y} = \dfrac{3\mu_l U_l}{a} & = 455 \text{ N.m}^{-2} \text{ in the liquid} \\ \mu_g \dfrac{\partial u}{\partial y} = \dfrac{2\mu_g U_g}{\delta} & = 356 \text{ N.m}^{-2} \text{ in the gas} \end{cases} \tag{7.4}$$

This unbalance of force is very likely the trigger of the instability, as the Ghost-Fluid tries to impose an exact equilibrium on the interface.

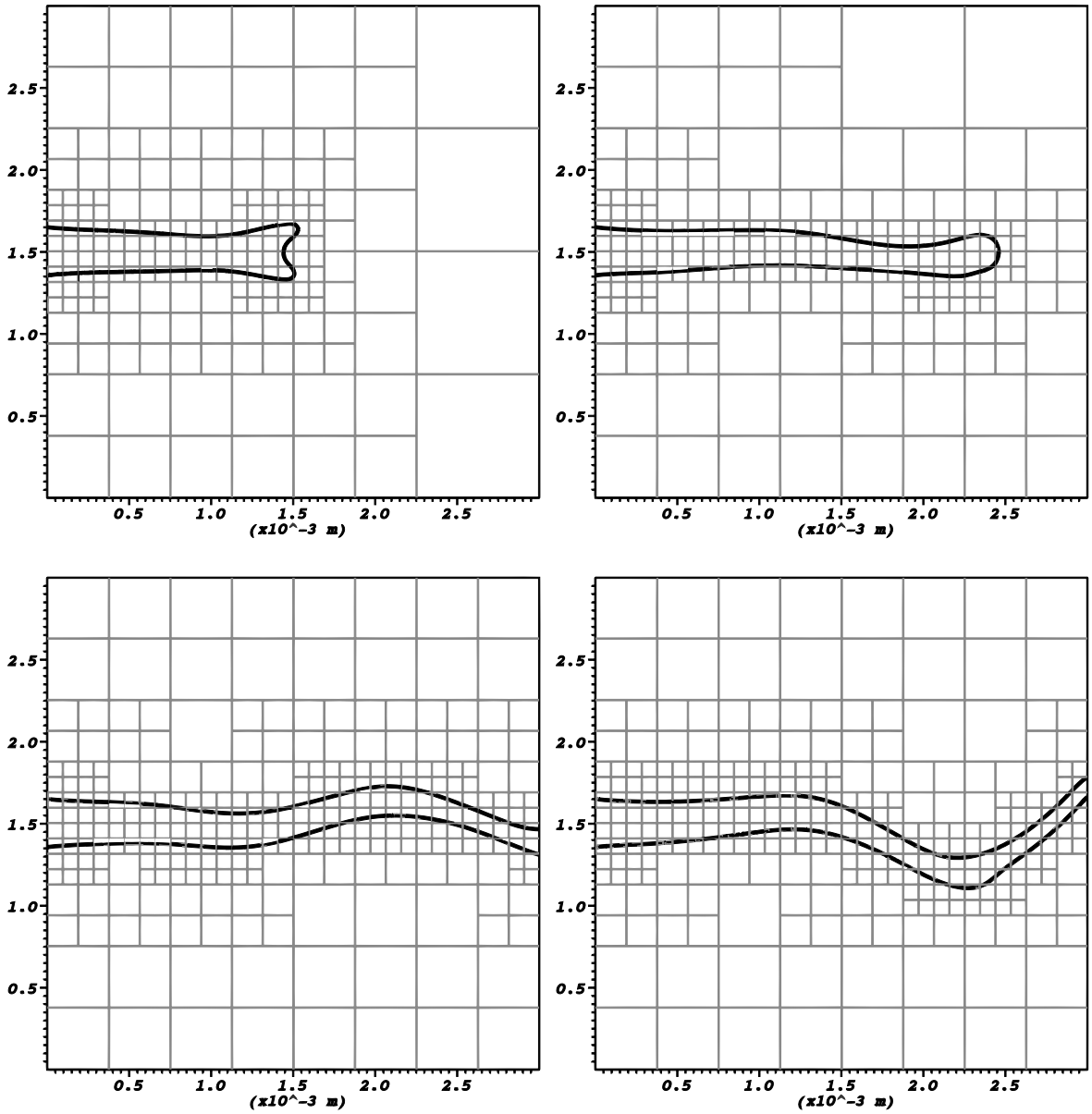


Figure 7.3: Initial deformation of the planar sheet, mesh blocks configuration (squares are 8×8 cell blocks). $\Delta t = 2.5 \times 10^{-4}$ s.

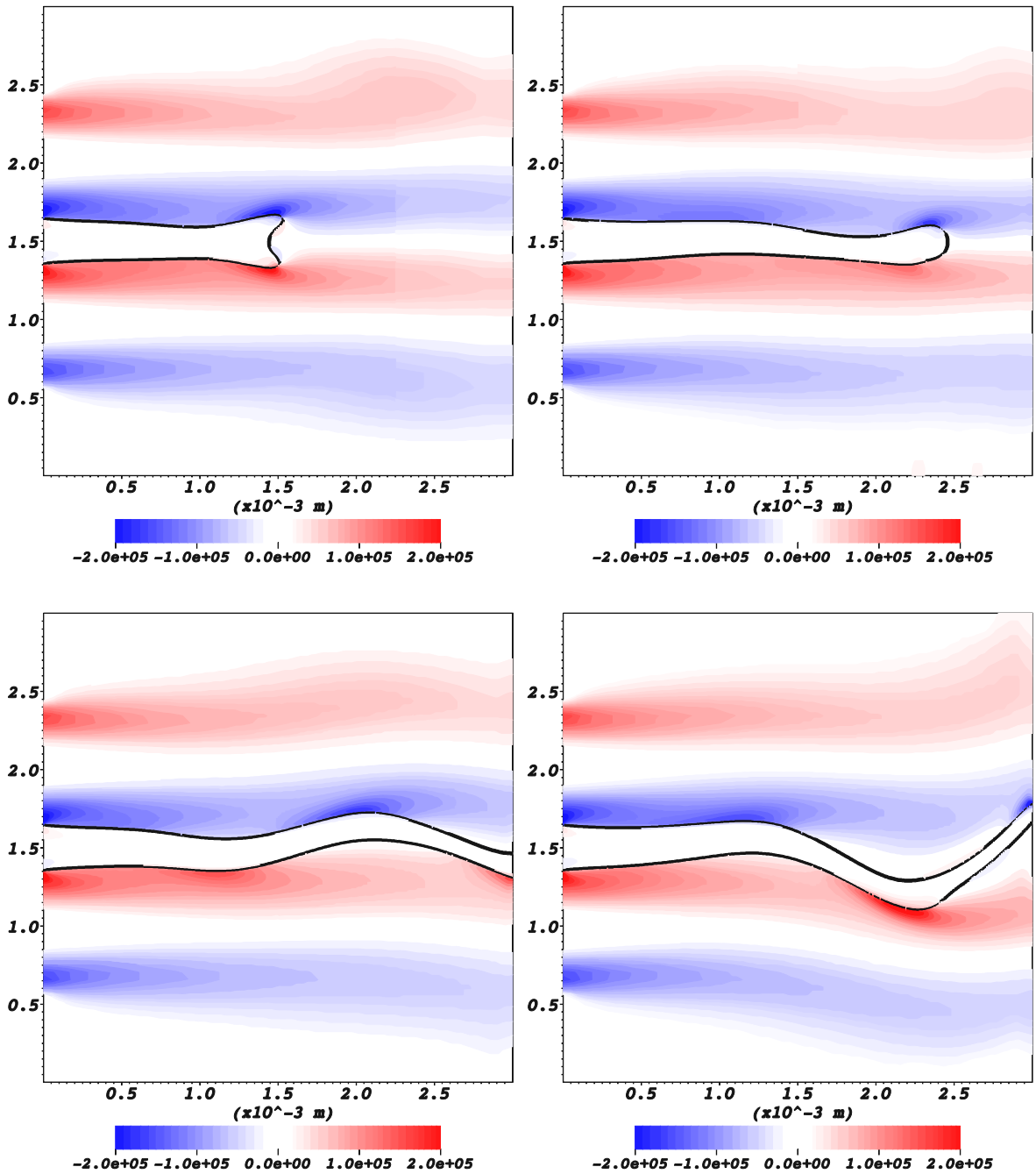


Figure 7.4: Initial deformation of the planar sheet, vorticity field. $\Delta t = 2.5 \times 10^{-4}$ s.

After the observation of the development of the instability, figure 7.5 (with the corresponding mesh 7.6)gives a glance of four different instants in time, where the primary oscillation can be observed fully developed. As already mentioned before, one of the most evident aspects of the sheet evolution is the stretching of the sheet thickness. The reason behind this is quite simple: the liquid velocity profile is altered by viscous diffusion; as the boundary layers are similar, the higher momentum in the gas due to its superior speed accelerates the liquid sheet, which is forced to shrink to conserve the mass flow.

The second interesting effect visible in the sheet oscillation is the accumulation of liquid at the top of the oscillation. This is due to the deformation of the local velocity field diverted by the presence of the liquid interface, which acts like a wall (no slip conditions are impose on the interface). The clump of fluid is bound to the sheet by a thin ligament which stretches until break-up, which usually happens near to the clump itself. The final behaviour is an intermittent release of liquid packages, upwards and downwards, downstream. In the higher resolution simulation (later in this chapter), however, the clump does not form; instead, a secondary atomization process originates in these locations a cloud of small droplets, triggered by small local instabilities which generate tiny recirculating ligaments at first.

The vorticity contours show the peaks of vorticity located into the gas boundary layers, which show the behaviour of the high speed air streams above and beyond the sheet. Local accumulation of vorticity are behind the liquid packages, due to the local acceleration of the gas imposed by the tangential condition of the curved interface. Downstream the liquid structures the gas forms slower wake whirling structures. These liquid-gas interactions are fundamental in the generation of the oscillations: the sheet modifies the gas flow by acting like a wall. But, differently from a real wall, it can move under the feedback of the fluid, until an equilibrium is found.

2.3 The oscillation frequency

In order to measure the oscillation of the sheet, the ordinate of the upper sheet wall has been evaluated at different fixed abscissas, its value stored for each time step (or every few time steps), as presented in figure 7.7. The sinusoidal form of the signal time history is expected to appear in the proximity of the injector, so the abscissas considered are each half sheet thickness in the range $x/a = \{1, 2..., 8\}$.

The resulting signals show a very constant sinusoidal movement of the sheet, with downstream increasing amplitudes. It will be clearer, however, that this behaviour is simplified by the relatively low resolution, as a lot of very tiny interfacial interactions are missed with such a cell size. The advantage is that it becomes easier to measure the oscillation frequency thanks to the smoothness of the signal, and the global oscillation phenomenon is physically correctly captured, cleaned out of other "noisy" instabilities. The frequency of the oscillation is very similar for each signal; the measured value is 2645 Hz, a value quite higher than experimental ones, but consistent with the 10 bars pressure condition and the increased shearing forces (due to the overestimated viscosities). As it can be seen from results of air-water experience realized in Fernandez [31] shown in figure 7.8, the trend is an increase of frequency which follows an increasing injection pressure. Some different oscillation regimes can be found by changing the parameters such as the boundary layer thickness or inflow velocities of both air and water. This research work has been already performed in Couderc [18], so it is not the aim of this work to replicate its results if not for testing/validation purposes. In this sense, some indicative results only are presented, as a small parametric study of the inflow velocities. The solver-sheet configuration is not yet totally satisfactory, in the sense that a lot of quite "annoying" hypothesis must be done to find approximately realist results; the next logical step will be to improve the numerical method, and to exploit the AMR to realize higher resolution and full three dimen-

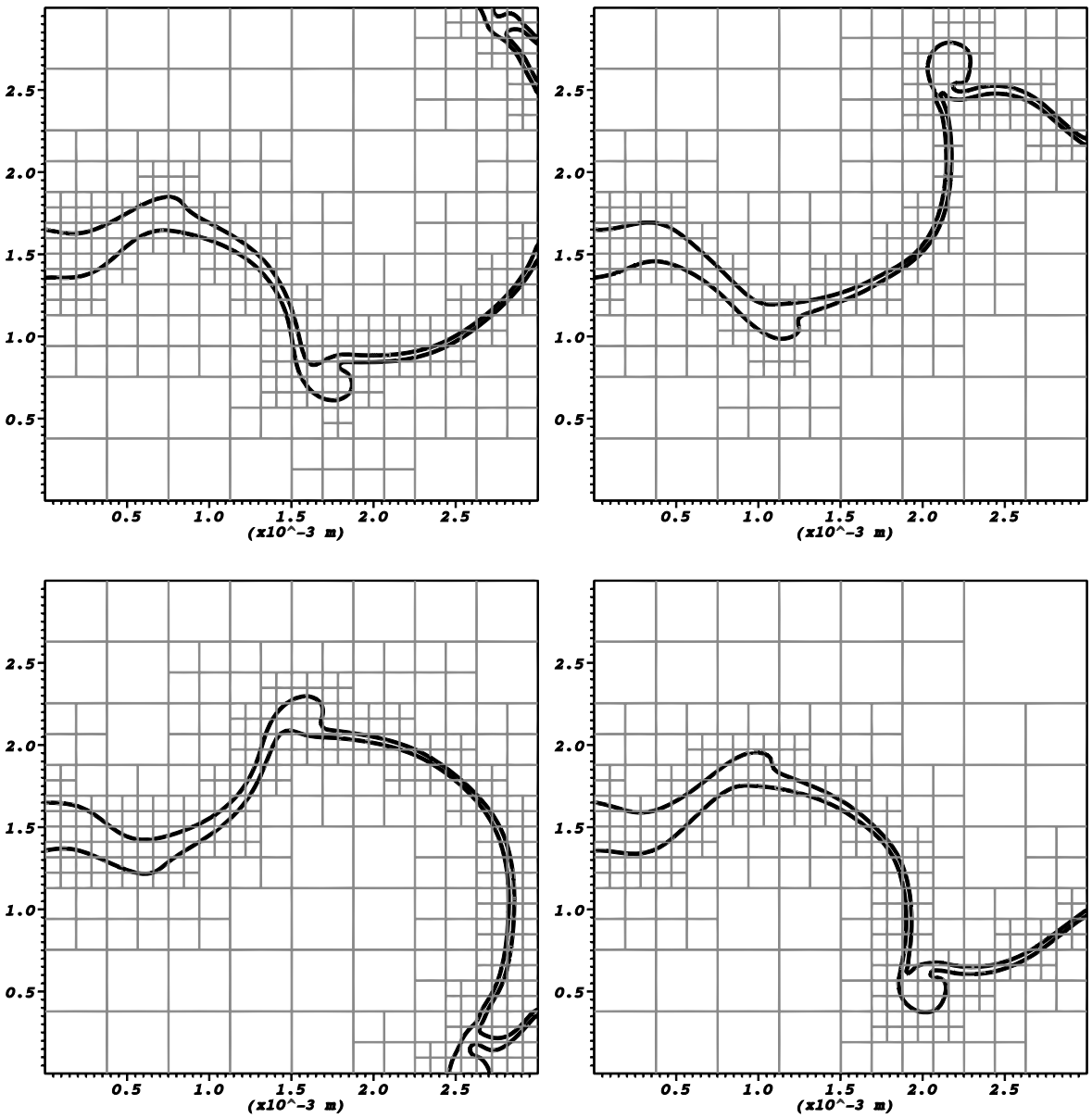


Figure 7.5: Steady oscillation regime of the planar sheet, mesh blocks configuration (squares are 8×8 cell blocks). $\Delta t = 1.25 \times 10^{-4}$ s.

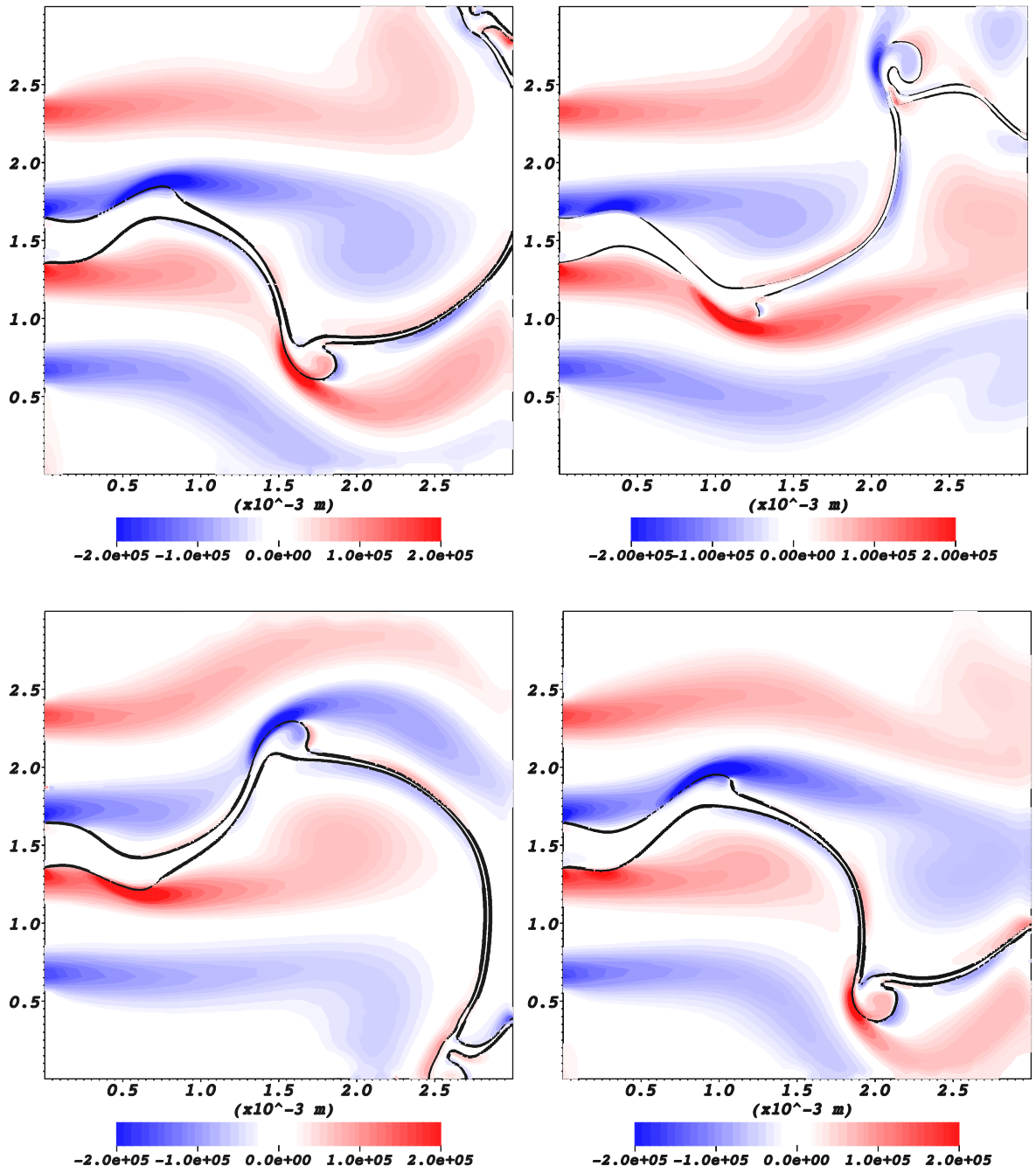


Figure 7.6: Steady oscillation regime of the planar sheet, vorticity field. $\Delta t = 1.25 \times 10^{-4}$ s.

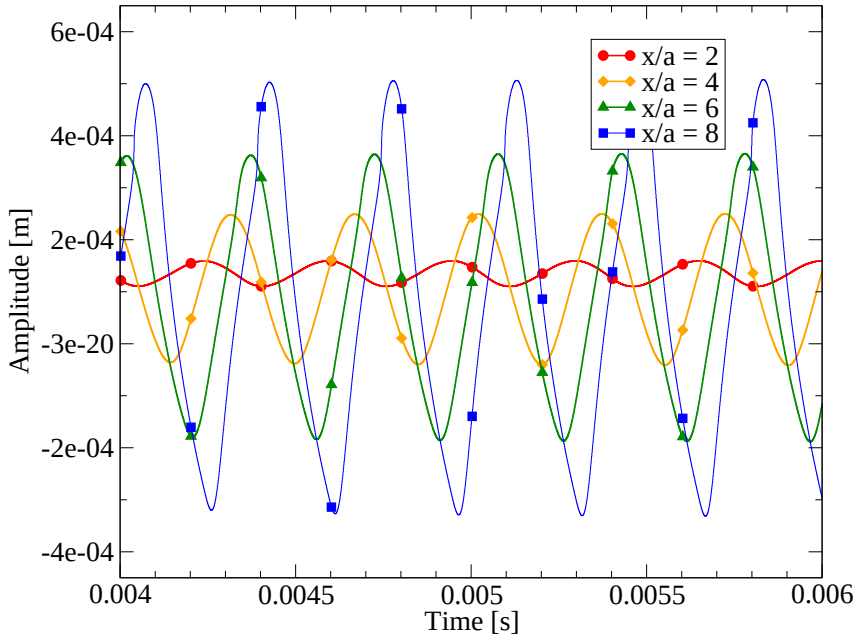


Figure 7.7: The measured signal of the sheet oscillation, different abscissas.

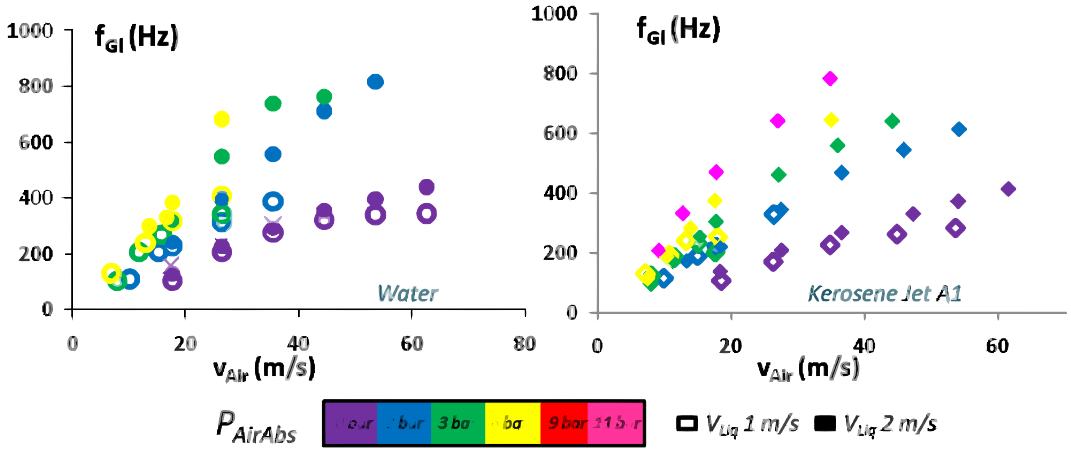
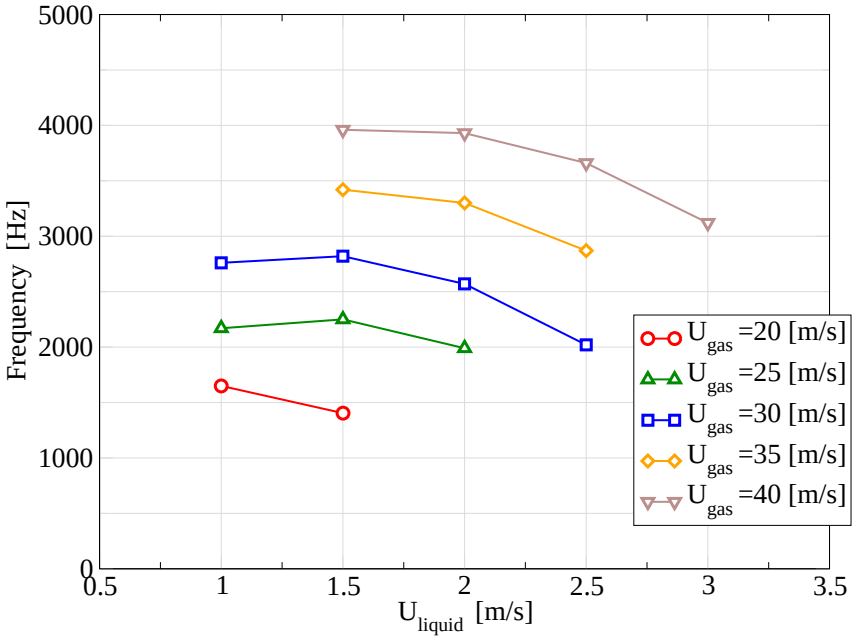
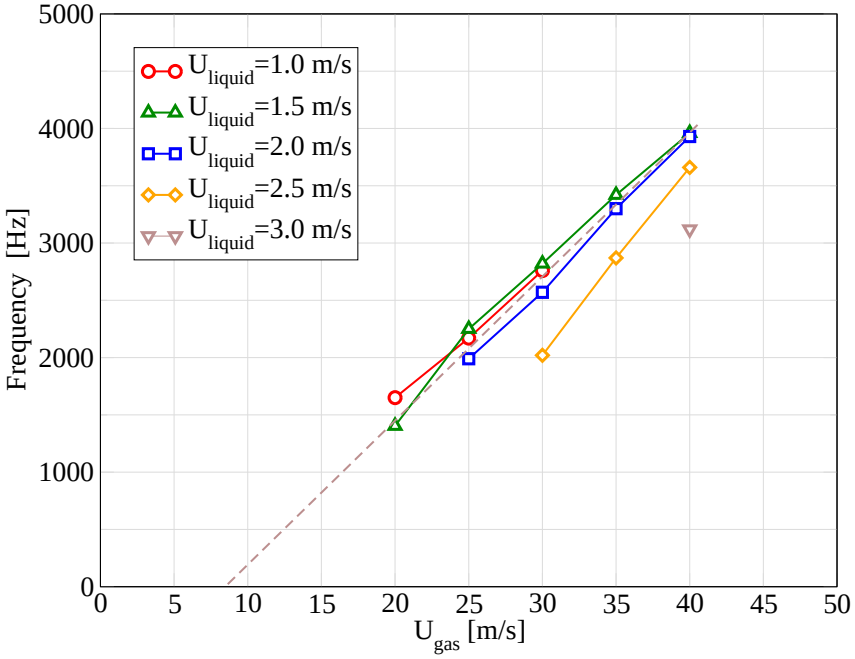


Figure 7.8: The computed oscillation frequencies for the experiences in Fernandez [31], realized at the LACOM laboratory of ONERA.

sional simulations. It is known in literature that the gas-liquid velocity ratio is a fundamental parameter in the disintegration process. In particular, a linear dependency between frequency and gas velocity can be found; it is less clear the influence of the liquid injection, however. A set of simulations is presented in figure 7.9a The results are coherent with literature, as Lozano et al [56], reprised in figure 7.1a. Mansour and Chigier [58, 59] distinguished on a similar graph the three regime zones defined by couples gas-liquid velocities, where only the intermediate one (called "B") allows the global oscillation (all the parameters in figure 7.9a belong to zone B). For higher values of liquid velocity (zone "C"), the presence of dilatation waves forbids the development of sinusoidal ones; it is reported by Couderc [18] that in the numerical simulations nothing happens, the flow is stable, and the simplified model does not reproduce the real behaviours. For slower values (zone "A"), there in no more global oscillation in experiments. [110] defines here the "stretched streamwise ligament breakup" zone, where no streamwise instabilities have the



(a)



(b)

Figure 7.9: The computed oscillation frequencies for different injection velocity of both air and water.

time to develop: the transversal ones grow too rapidly, generating an instantaneous formation of ligaments and droplets. Being the numerical simulation two dimensional, the transversal instability cannot clearly exist. Couderc [18] found in this zone a different regime from the "B" zone, where more wavelengths coexist and the oscillation process is less coherent. However, no more investigation is done in this zone. It is clear, however, that the most advantageous region for the atomization remains the "B" zone. The second graph in figure 7.9b shows the same results

as before but in function of the gas velocity. As expected, the dependency is approximatively linear. A tendency line (dashed) is drawn among the points, cutting the zero slightly before 10 m/s. Only a small decrease in the frequencies can be observed for increasing liquid velocities.

2.4 Increasing resolution and domain

In this section the resolution has been increased as well as the computational domain with the help of the AMR to reduce the workload. The interest is to see the evolution of the sheet with less influence from the boundaries. Another test which has been performed is the comparison between the two kinds of injectors in figure 7.10, of which the simulations show some qualitative and quantitative differences. The physical parameters for this more accurate simulation ($\Delta x_f \approx$

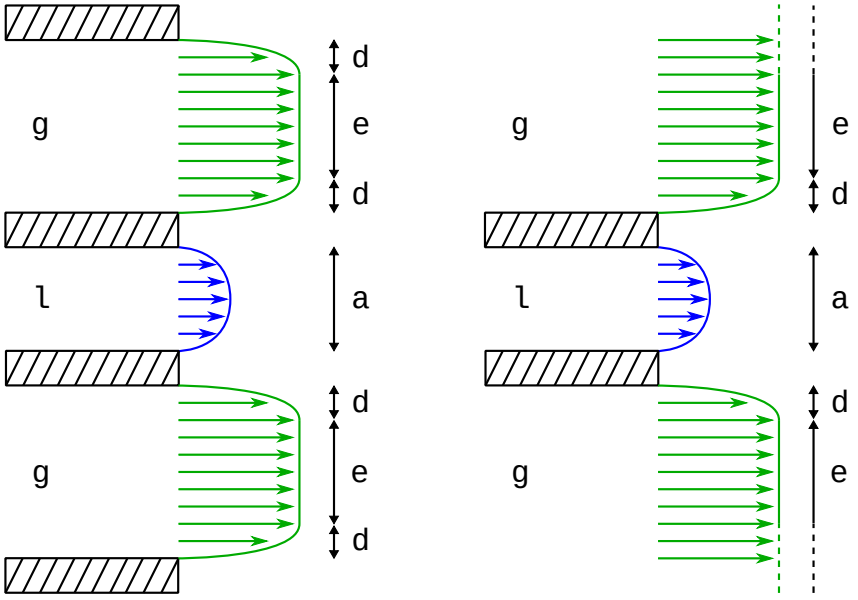


Figure 7.10: Two configuration of injector, with (a) confined and (b) unlimited gas injections.

5.86×10^{-6} m) are the same as before; the geometry presents some differences:

Physical parameters	Dimensions
$\rho_g = 12.26$ kg.m^{-3}	$L_x = 6 \times 10^{-3}$ m
$\rho_l = 1000$ kg.m^{-3}	$L_y = 6 \times 10^{-3}$ m
$\mu_g = 1.78 \times 10^{-3}$ Pa.s	$a = 150 \times 10^{-6}$ m
$\mu_l = 1.132 \times 10^{-2}$ Pa.s	$\delta = 3 \times 10^{-4}$ m
$\sigma = 7.28$ N.m^{-2}	$e = 1.5 \times 10^{-4}/inf$ m
$U_g = 30$ m.s^{-1}	$\Delta x_f \approx 5.86 \times 10^{-6}$ m
$U_l = 2$ m.s^{-1}	$\Delta x_c \approx 187.5 \times 10^{-6}$ m

Table 7.2: Parameters for the low resolution sheet simulation, a more accurate equivalent resolution 512^2, six AMR levels, double domain.

Confined injection

In the confined injection simulation the results (figure 7.11) are similar to the lower resolution simulation, confirming the mesh convergence for the global oscillation problem. The initial destabilization occurs very soon, caused by an entering flux which can be seen on the lower wall around 3 mm from the injection. It seems to be originated by a recirculating flow coming from the initial step of the sheet shape, which impacts the outflow condition lower wall. It is a bit strange because it is a very thin channel of high speed gas; however, it does not seem to perturb the establishment of the oscillations. The oscillation correspond to the description done in the previous section. An interesting detail visible in this simulation is the evolution of the fluid packages, which now tend to evolve towards a ligament rolling up. The amplitude of this oscillation exceeds the upper and lower limits of the domain. The computed frequency is 2663 Hz, actually very similar to the value computed before.

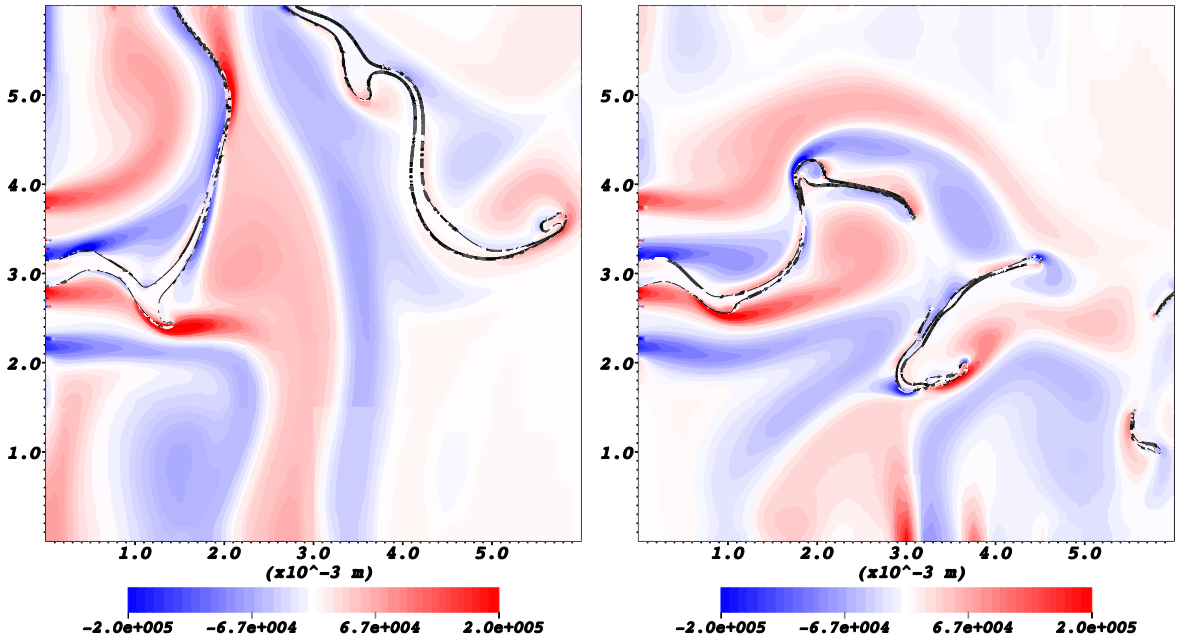


Figure 7.11: Destabilization and fully developed oscillation for the confined injection configuration.

Full injection

This case presents some important differences respect to the previous one (figure 7.12). First of all, the gas flow is less likely to be shifted by the movement of the sheet; for this reason the oscillation is much more confined vertically than before. The break-up seems to happen on average before that in the previous case, about one millimetres upstream. The sheet presents in general more fragmented, with smaller structures and more scattered droplets. The oscillation frequency is higher, with a value of 3494 Hz.

The conclusion is that the different configurations do not change radically the mechanism of primary atomization, but they have an important impact on the global oscillation frequency and on the development of the sheet atomization.

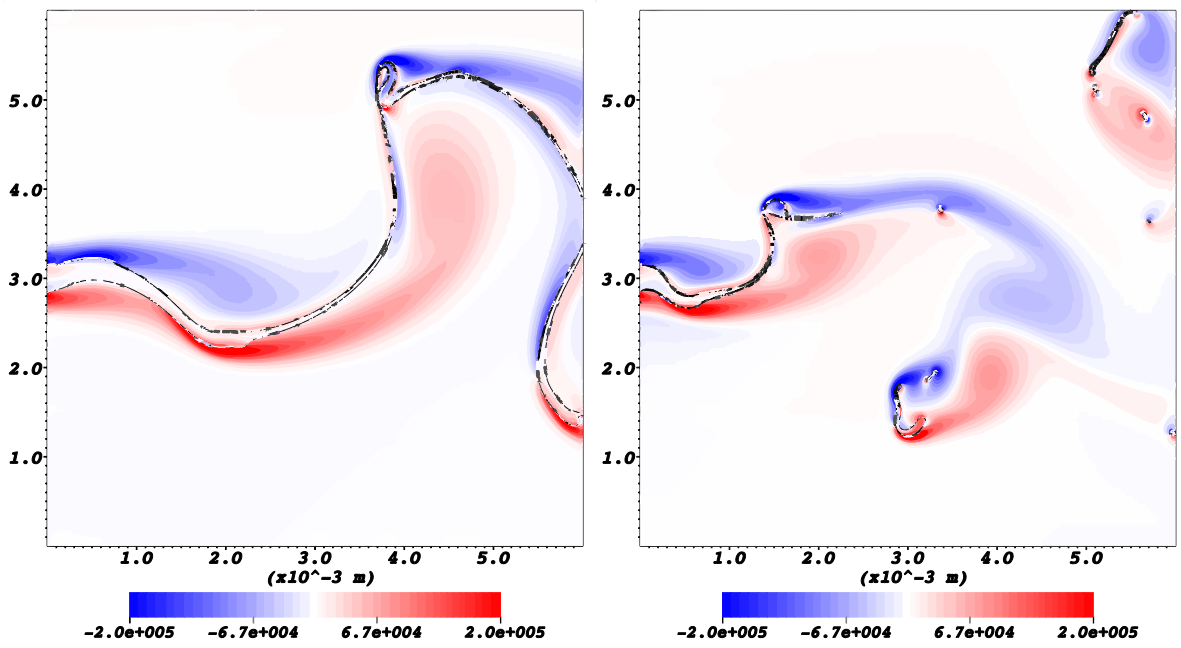


Figure 7.12: Destabilization and fully developed oscillation for the full injection configuration.

3 High resolution simulation

3.1 Simulation configuration

This section describes a very high resolution simulation ($\Delta x_f \approx 9.76 \times 10^{-7}$ m) of a two dimensional sheet atomization, with realistic injection parameters. The parameters are chosen to match as much as possible the experiment environment, in particular those of Fernandez [31], the bank represented in figure 7.13. The injection involves 10 bar pressurized air and kerosene. The domain is more than five times the one of the first simulations, reaching 16 mm. The injection is confined is some centimetres. The final parameters are the following: Again, no

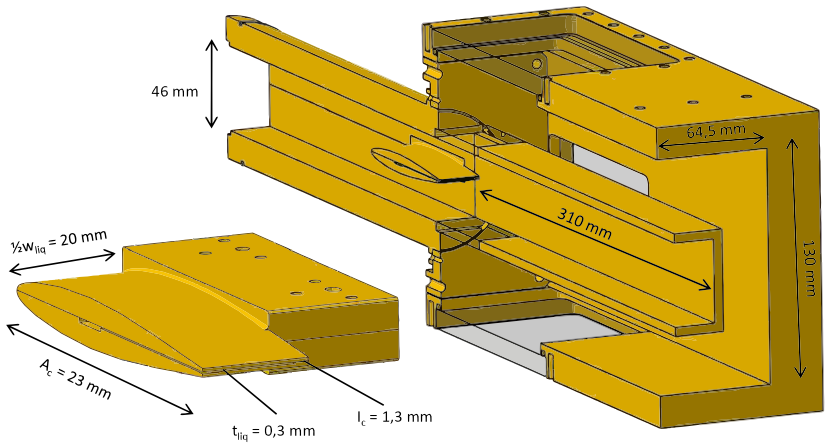


Figure 7.13: ONERA Lacom experimantal installation, from Fernandez [31]: *airblast* planar injector.

perturbation is given to the initial condition, but the instability take place very soon as for the

Physical parameters	Dimensions
$\rho_g = 12.28$ kg.m^{-3}	$L_x = 16 \times 10^{-3}$ m
$\rho_l = 820$ kg.m^{-3}	$L_y = 16 \times 10^{-3}$ m
$\mu_g = 18.27 \times 10^{-5}$ Pa.s	$a = 150 \times 10^{-6}$ m
$\mu_l = 2.3 \times 10^{-3}$ Pa.s	$\delta = 1 \times 10^{-3}$ m
$\sigma = 2.3$ N.m^{-2}	$e = 2 \times 10^{-2}$ m
$U_g = 30$ m.s^{-1}	$\Delta x_f \approx 0.98 \times 10^{-6}$ m
$U_l = 2$ m.s^{-1}	$\Delta x_c \approx 500 \times 10^{-6}$ m

Table 7.3: Parameters for the ultra-high resolution sheet simulation, equivalent resolution 16384^2, 12 AMR levels, large domain.

previous simulations. The same non dimensional configuration of the flow is evaluated:

$$\begin{cases} Re_l = \dfrac{\rho_l U_l a}{\mu_l} = 107 \\ Re_g = \dfrac{\rho_g U_g \delta}{\mu_g} = 20164 \\ We = \dfrac{\rho_g (U_l - U_g)^2 a}{\sigma} = 63 \\ M = \dfrac{\rho_g U_g^2}{\rho_l U_l^2} = 2.76 \end{cases} \tag{7.5}$$

The mesh is composed by twelve levels of refinement (figure 7.14).

3.2 Results

Figure 7.15 shows the initial deformation of the sheet. As the viscosity are not altered, the shearing effects are reduced, resulting into a different interaction between the sheet and the air flows. The sheet walls are subject to different shearing instabilities (which can remind the Kelvin-Helmholtz, figure 7.19a) which give a "humped" shape, mainly to the upwind side of the sheet. The fore of the sheet is subject to a down step-like recirculation, which immediately starts the sequence of ligament formation, stretching and break-up, with subsequent formation of droplets. Some wake vortexes like those found on steps can be seen, well defined as they detach and travel downstream. It is possible to detect von Karmann wake vortices behind the bigger droplets.

In the second image of 7.15 a ligament has formed between 3.5 and 5.5 mm: in a three-dimensional analogy it would remind the formation of the membranes in the cellular and in the stretched ligament break-up; obviously the two dimensional nature of the simulation does not allow any transversal transfer of mass, so the analogy ends here. It is at the end of this thin structure that the ligament breaks (as it did for the low resolution simulations, just before the liquid packs). It is clear that the liquid packages do not have time to form, but instead they undergo instant break-up in droplets as soon as they are generated, thus defining a better atomization quality than expected before.

As the simulation carries on, the first oscillation starts. The sheet begin to accelerate upwards, while it continues to undergo a secondary atomization at the end and contemporaneously along its body, as shown in 7.16. A preliminary hypothesis on the small instabilities which occur in this phase suggest the possible presence of Rayleigh-Taylor instabilities (7.19b) when the

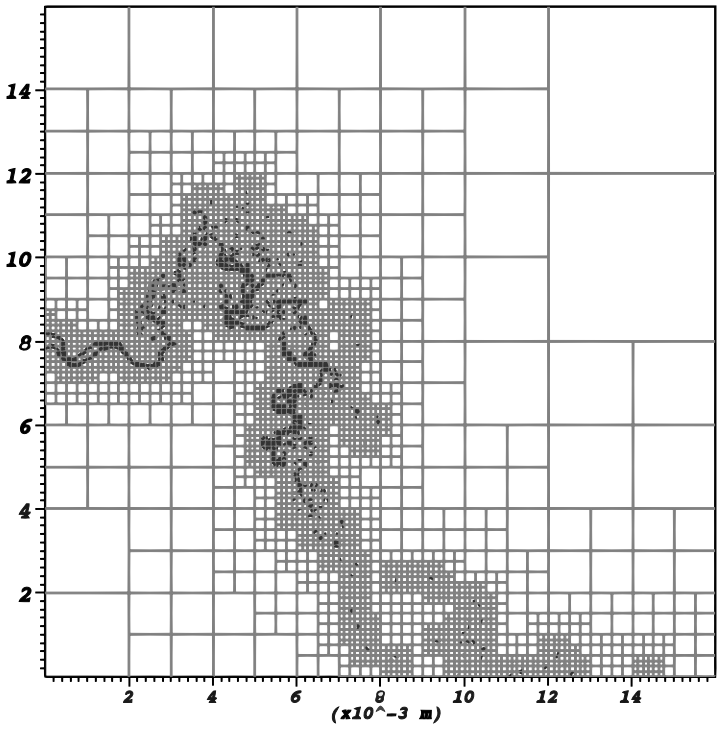


Figure 7.14: High resolution sheet, first eight levels of refinement (three more finer levels are not shown, squares are 8×8 blocks).

liquid accelerates upwards or downwards (the downstream sheet wall is an interface subjected to inertial acceleration in the heavy-light direction). It is still to be confirmed than the wavelength and the acceleration parameters allow such an instability to take place. The asymmetrical shape can be justified by the rightwards fluids motion.

Image 7.16 shows the accomplishment of the period, when the sheet starts its downwards motion. In this picture it is clear that the oscillation creates fine dispersions of droplets in all directions, some coming from thin ligaments breaking up (as for a Rayleigh-Plateau instability), others from local recirculations. The ligament breaks at a distance of approximately 3-4 mm from the injection, a value which is not aberrant with the experiences of Fernandez [31]. The droplet average size (which has been only estimated, not rigorously calculated) is around 20 μm, underestimated face to the experimental values of 30-40 μm (figure 7.17), but the lack of three-dimensional effects can partially explain this difference. A droplet formation mechanism is shown in figure 7.18. The ligaments stretches and breaks, generating in the process two separate droplets and an intermediate smaller structure. If in figure 7.18a the size of the final droplets is several times the ligament diameter (a behaviour well documented in literature), in figure 7.18b it seems to be more due to a numerical effect, as confirmed by Menard [66]. It is still unclear whether the Level Set model without break-up models is sufficient to correctly capture these very small scale phenomena at those cell sizes. Figure 7.20 shows the increasing level of detail given by the multi scale grid, from the whole sheet to the single droplets. Figure 7.20 (d) in particular show the finest mesh covering the droplets: they are captured with at lest eight-ten cells, and are maintained until they exit the computational domain (or collide with other structures).

In figure 7.21 an image from an experience of kerosene injection into 11 bar air pressure is shown, with $U_l = 1$ m/s, $U_g = 30$ m/s. It is clear from that image the three dimensionality of the atomization process, in which the primary atomization leads to the formation of streamwise

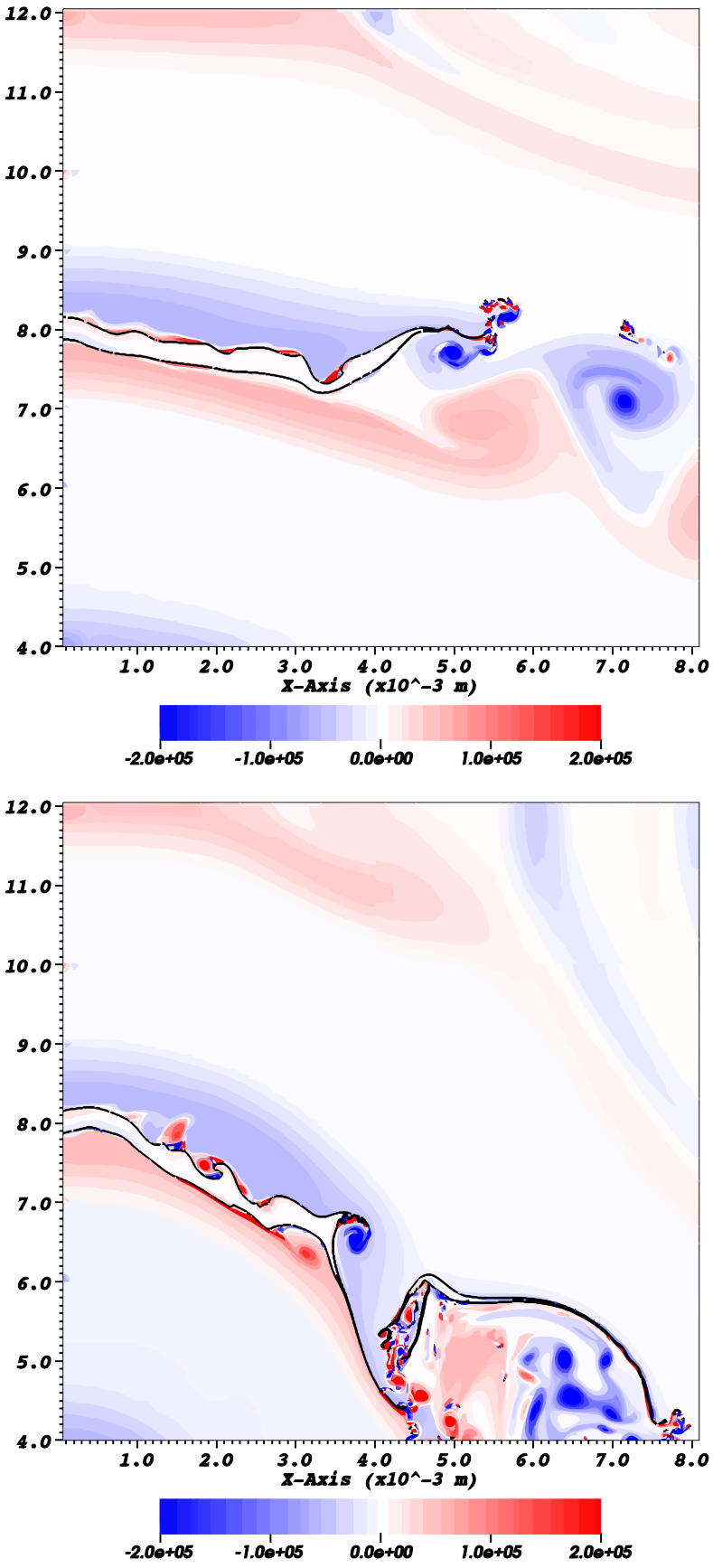


Figure 7.15: Results from high resolution liquid sheet disintegration, primary atomization. Initial destabilization and ligament break-up.

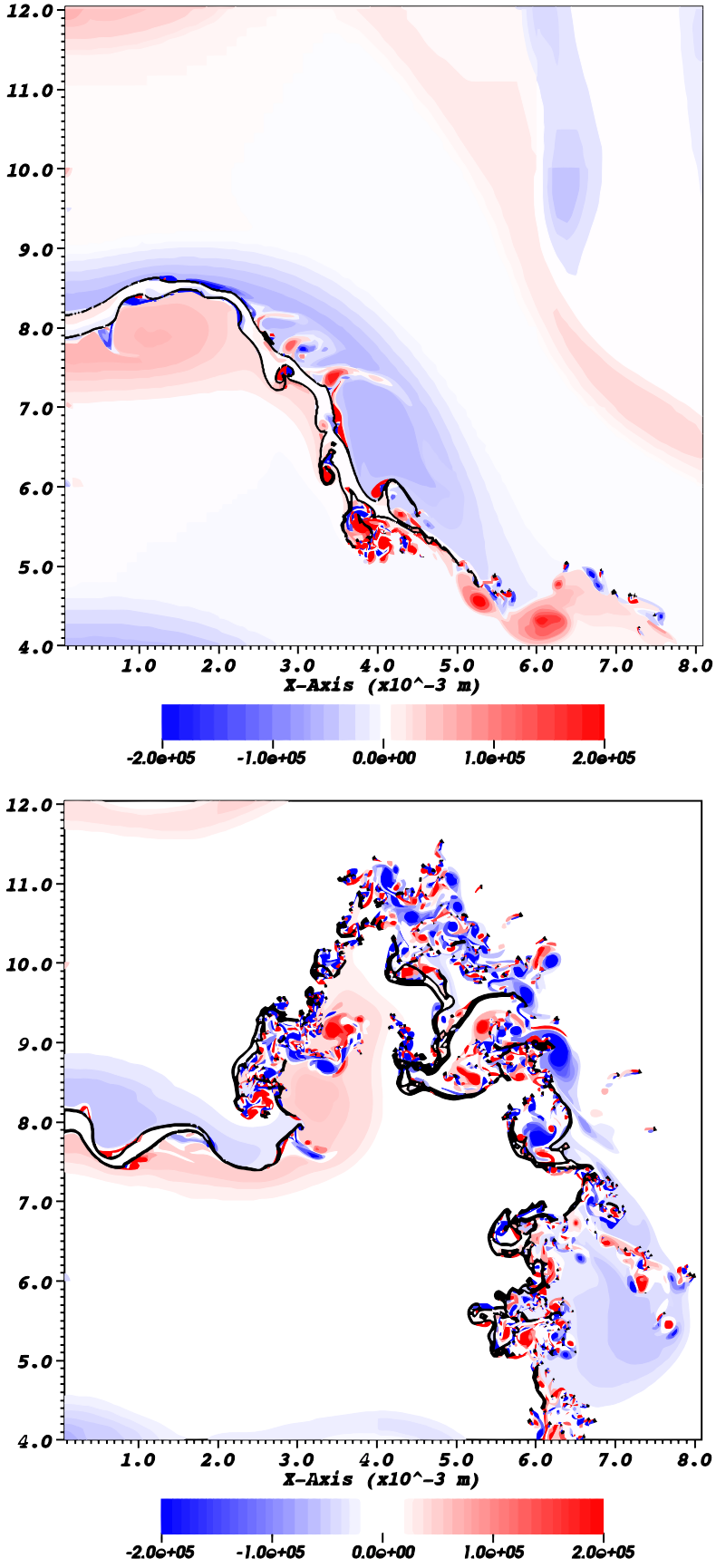


Figure 7.16: Results from high resolution liquid sheet disintegration, primary and secondary atomization. Formation of the droplet cloud.

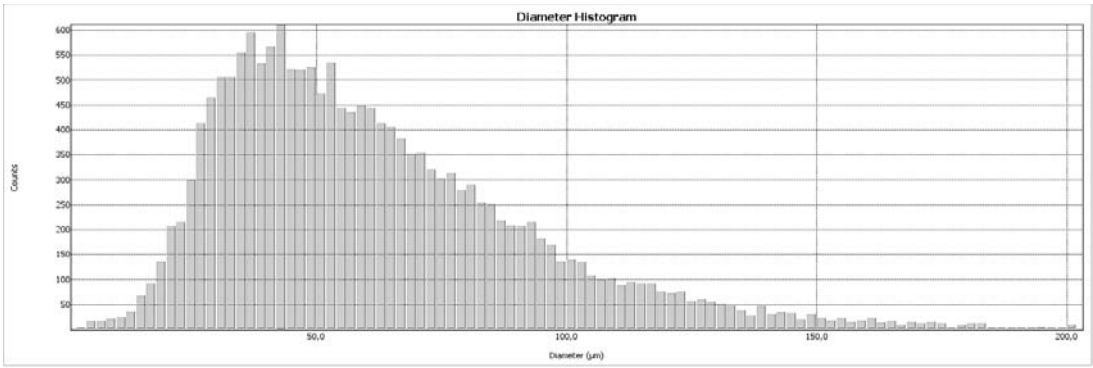
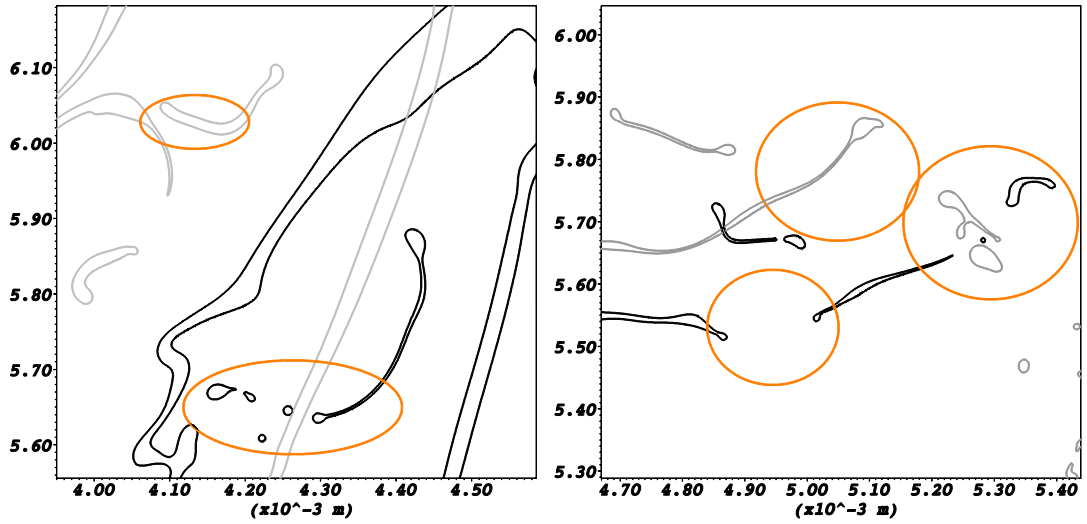


Figure 7.17: Granulometry of a kerosene sheet, $U_l = 1$ m/s, $U_g = 30$ m/s, 11 bar pressure.



(a) (b)

Figure 7.18: Two examples of ligaments break-up: the ligament stretches and break, resulting into two separate structures with the formation of satellite droplets. Two different instants: from grey to black time is advancing.

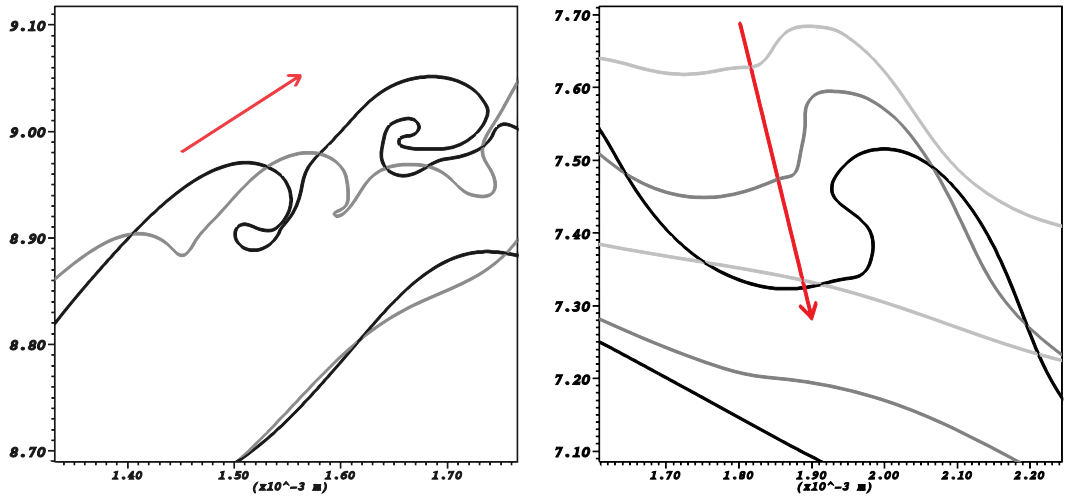
ligaments. In consequence, the early secondary atomization is less pronounced than in the simulation: the transversal mass transfer is an important part of the disintegration process. In general, it seems that several of the smallest structures in the simulation do not exist in the real process, mainly along the sheet body.

Even if the gas injection section is much greater than in the low resolution simulations, the liquid seems to penetrate well into the air stream, confirming what obtained for the previous simulations.

3.3 Notes on computational performances

In this section an quick overview of some reference computational times are given, in order to show the advantages of the mesh adaptation and the capability to perform simulations with very fine resolutions.

256^2 simulation: for about ten periods ($t \approx 0.005$ s), around 18 hours of computation on 8 processors. Average block number (oscillation already established) around 450 blocks of 8×8 cells for $450 \times 64 = 28800$ nodes (uniform mesh value $256 \times 256 = 65536$, ratio=2.27).



(a) (b)

Figure 7.19: Details of the instabilities which develop on the surface of the sheet. (a) A Kelvin-Helmholtz like instability (b) A Rayleigh-Taylor like instability.

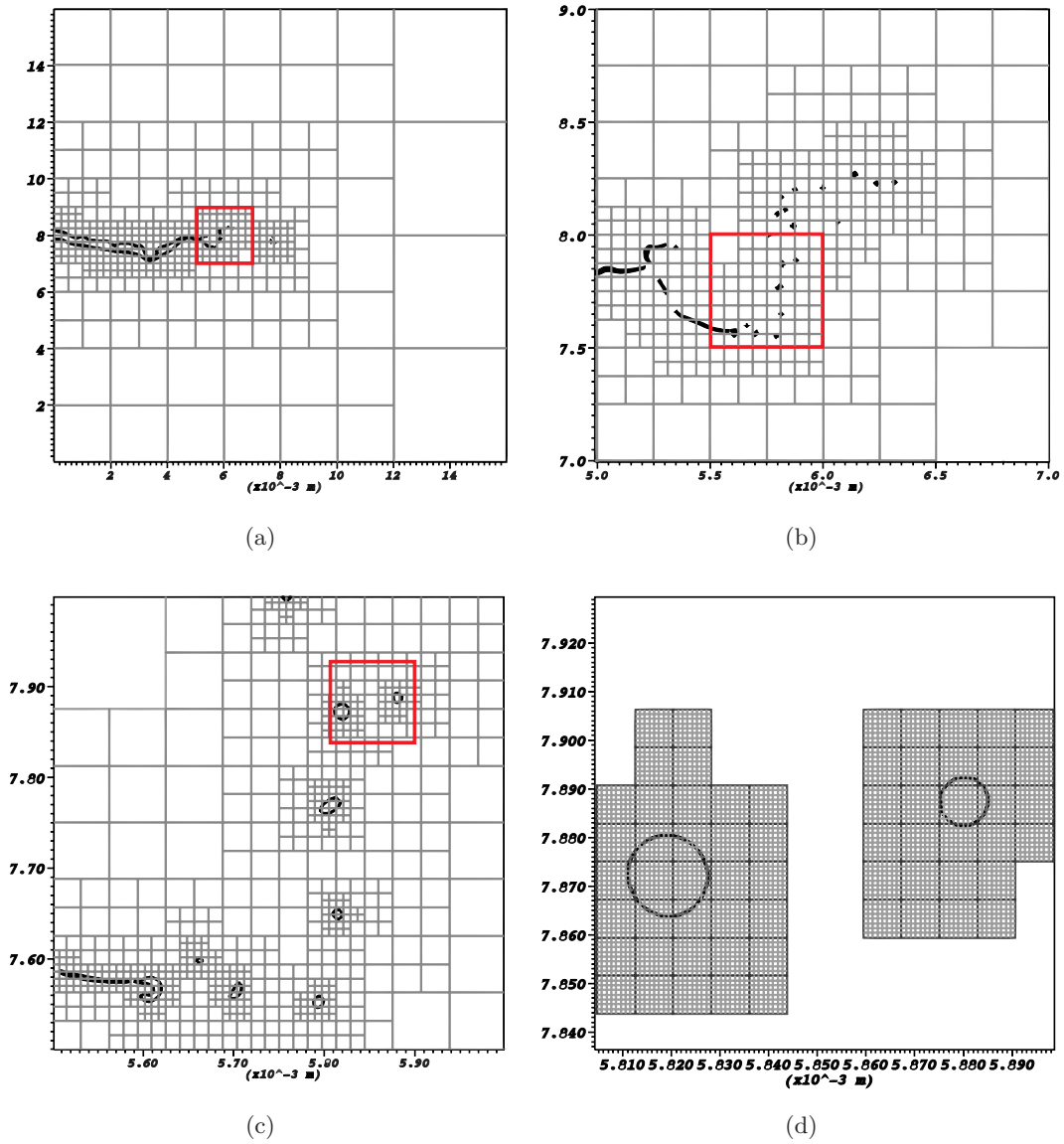
512^2 simulation: for about ten periods ($t \approx 0.005$s of simulated time) around 80 hours of computation on 16 processors. Average block number (oscillation already established) around 2000 (oscillations between 1700 and 2400 blocks) blocks of 8×8 cells for $2000 \times 64 = 128000$ nodes (uniform mesh value $1024 \times 1024 = 1048576$, ratio=8.192).

16384^2 simulation: for about one period ($t \approx 0.0001$s of simulated time) around 240 hours of computation on 128/256 processors, but around 960 hours for the transitory. Estimated block number (oscillation already established) around 50000 blocks blocks of 8×8 cells for $50000 \times 64 = 3200000$ nodes (uniform mesh value $16384 \times 16384 \approx 268000000$, ratio=89).

Obviously the advantages of the AMR increase as resolutions and domain increase (and if the refined region keeps a fraction of the total domain). In particular, the last simulation was realized with 256 processors, while the same on an uniform mesh would have required 4096 processors for the same processor workload. Still it remains unclear whether the accuracy of the solution remains comparable with so many refinement levels.

Conclusion

In this chapter the solver has been applied to the simulation of the primary atomization of a two dimensional sheared liquid sheet. The first part contains low resolution simulations which allow the computation of the global oscillation frequency. A small parametric test is obtained by changing both the injection speeds. Then a more accurate solution investigates the effects of the boundary conditions and two different injection profiles. The last part presents a full two dimensional primary and secondary atomization by a high resolution simulation, obtained with 12 levels of refinement. The phenomena observed include the primary oscillation, the formation and break-up of ligaments (with a good estimation of break-up length) and the formation of a cloud of droplets, of which the diameter is a bit underestimated by the simulation.



(a) (b)

(c) (d)

Figure 7.20: Details of mesh resolution, from whole domain to single droplets. The red square indicates the successive zoom. Each square is a 8×8 block. (a-b-c) Mesh blocks (d) Mesh cells on the finest level.

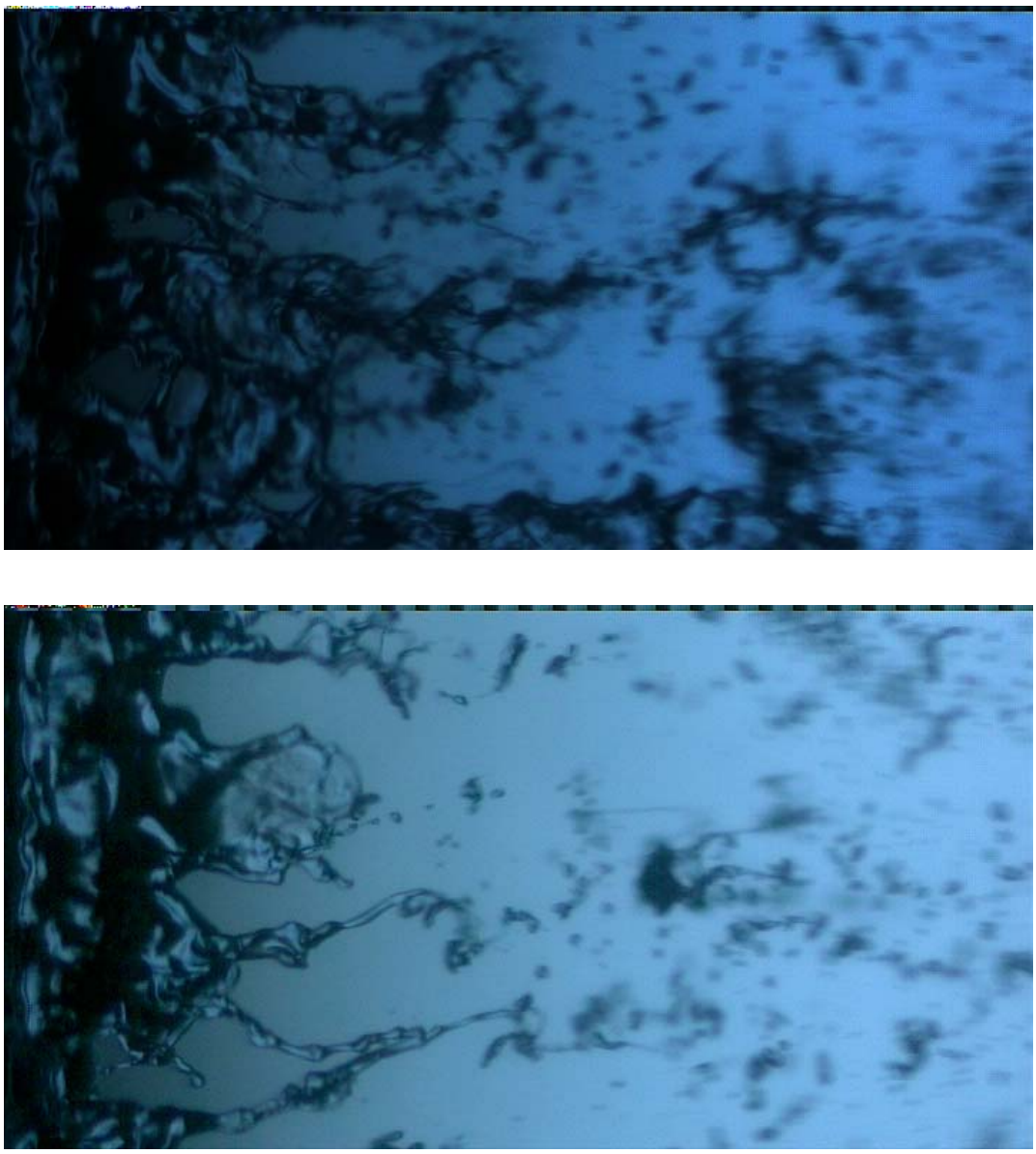


Figure 7.21: Image from a kerosene sheet atomization, ONERA Lacom, $U_l = 1$ m/s, $U_g = 30$ m/s, 11 bar pressure. Two different moments. The flow is directed rightwards.

Conclusions and Perspectives

This PhD thesis has been realized at the ONERA (DMAE/MH) laboratory in cooperation with CERFACS (CFD Team), and it focuses on the direct numerical simulation of two phase interfacial flows with parallel adaptive mesh refinement. The aim is to perform numerical simulation of a gas flow sheared liquid jet, in order make predictions of the final liquid spray configuration, which is a determinant parameter for the combustion quality in turbojets. The interest of the mesh adaptation (AMR) is to be able to perform simulations of multi scale phenomena, as it can be the combined liquid-gas flow entering a combustion chamber. The computational resources at disposal being constantly increasing, still they are not sufficient for a DNS of such large domains, considering that a sufficiently fine mesh is fundamental for a correct representation of the physical mechanism of interfacial flows.

The first part of the thesis has been dedicated to a study of incompressible two phase flows solvers and, in parallel, to the different mesh adaptation techniques. Given this work to be the continuation of the thesis of Couderc [18], the interest is readily focused into the numerical method used in this work, which consist in an exact projection method with an interface tracking via coupled Level Set/Ghost Fluid method. The AMR technique has been chosen to match these characteristics:

- ▷ Cartesian structured mesh;
- ▷ highly parallelizable;
- ▷ easy to integrate to an existing code;
- ▷ patch based (no stencil changes);
- ▷ written in Fortran 90.

Several algorithms are suited for solving this problem. The base idea comes from the Berger's refinement strategy, in which a set of different size mesh are used to dynamically refine a certain region. A variant of this method is the *quad-tree*, where the meshes have fixed size and are generated by bisection of the coarser ones. This has the advantage of being highly suited for parallel computation. The choice has consequently fallen on the PARAMESH libraries, which allow an existing numerical algorithm to be extended to parallel AMR with little effort. PARAMESH AMR is a block based *quad-tree*, which means that the discretizations do not have to be changed; instead, each blocks behaves as a Cartesian mesh and it can be solved into a loop over all blocks, given appropriate boundary conditions from the surrounding blocks or physical boundaries. PARAMESH allows a parallel repartition of the computational workload by dynamically distributing the blocks among the CPUs with an effort to maximise their locality.

A certain number of problems relative to the mesh adaptation have been faced. Most of them are bound to the refinement jumps, where the interpolation of boundary values does not assure the consistency of the numerical fluxes: corrections have been introduced to recover the conservation of the incompressibility condition and the convergence and accuracy of the elliptic solver. The most evident obstacle has been the development of an efficient elliptic solver

which could afford the computation of high density ratio flows on the AMR framework. A Fast Adaptive Composite multigrid preconditioned Bi-CGstab solver has been developed. It benefits of both the multigrid convergence speed and the Krilov method robustness, and has proven able to converge for all the test cases. Still some drawbacks persist, like the average scalability of the multigrid (due to the AMR design) and the higher cost of the Bi-CGstab needed to solve a non symmetrical system.

A set of validation tests have been proposed to test the code against the transport equation alone, the Navier-Stokes solver and the full two phase solver. The aim of these test is to demonstrate that the fine grid precision can be maintained by a local refinement on the interface, thus reducing the computational cost of the simulation with negligible reduction of the solution accuracy and the convergence rates. The code has always proven to be able to enhance the solution by the addition a refinement level. Speed-up test have shown its parallel performances and its capability to be faster than the original single grid solver, when a sufficient ratio of points to be reduced is achieved by refinement. If only two dimensional test cases have been proposed, there is no loss of generality, as the extension to the three dimensions is straightforward, where the benefits of the AMR will be even more important.

The final purpose of the code would be to simulate the assisted disintegration of a liquid sheet. The realization of such a simulation meets two main obstacles. The first concerns the numerical method and the limitations of the solver: the high density ratios like in air and water flows still pose a serious challenge to the code temporal stability, and a too high Reynolds number give rise to spatial convergence and boundary conditions issues. The second obstacle is the multi scale nature of the physical process, where the dimensions of an injector are large if compared to the thickness of the liquid sheet and to the small structures generated during the atomization process. In this work the mesh adaptation has been introduced to deal with this latter: a grid refined on the interface allows the representation of larger domains at similar computational times or, conversely, more cheap grids for fixed size domains. In this sense, a first set of low resolution simulations has been performed in order to catch the longitudinal oscillation of a two dimensional sheared liquid sheet, resulting into a very good estimation of the oscillation frequencies in comparison with the original code, at the price of artificially reduced Reynolds numbers. Successively, the real multi scale capability of AMR has been shown in the high resolution simulation. In this latter both the domain scale and the interface resolution have been risen. This latter has allowed a stable simulation with real air/kerosene parameters, and so real Reynolds numbers.

Still the code needs further improvements, in particular in the Level Set interface tracking. The lack of an exact mass conservation may be at the origin of the stability problems, as they mostly appear in case of under resolution of the interface. The coupling with a VOF method may enhance its capabilities, and allow atomization simulations with reduced and more practical resolutions. The AMR contribution is expected to grow when full three dimensional simulations will be performed, even if its main drawbacks persist. These are partially implicitly given by design choices, as for the parallel performances of the multigrid, and partially come from unavoidable mathematical issues, like the difficulties bound to the elliptic equation solve on composite meshes. Still the main goal of the mesh adaptation, the maintaining of the fine grid resolution, can be considered as achieved.

Appendix A

Interpolations

1 Computation of bi-quadratic interpolation coefficients

This interpolation allows the computation of the four children of a cell centered refined cell using a symmetrical 9 points stencil of coarse points. In figure A.1 there is the representation of the nine cells ($c_{i,j}$) which participate to the interpolation of the fine point x.

$$x = \sum_{i,j=1}^{3} w_{ij} c_{ij} \tag{A.1}$$

The evaluation of the weights of each coarse point in the interpolation for the fine ones is an operation which can cost a lot of computational time because of the frequency at which the prolongation routine is called in an AMR computation. As the refinement geometry is fixed, these weights can be computed only once, at the beginning of a simulation. In order to compute the fixed weight of the coarse points for a bi-quadratic prolongation, the fixed coefficient for a quadratic interpolation must be computed at first. The three points x in the following situations are found with the linear combination of the points a, b, c by the weights contained in the three vectors M_1, M_2, M_3 (actually three more vectors are stored, the mirrored stencils respect to the central point M_4, M_5, M_6) so that $x = \{a, b, c\}\{m_1, m_2, m_3\}^T$

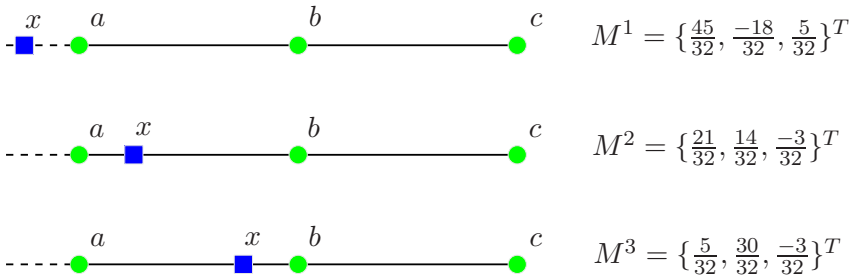


Figure A.1: Interpolation scheme for the general quadratic interpolation. Green: stencil; blue: interpolated point.

The intermediate points α can be stored into a vector $A = \{\alpha_1, \alpha_2, \alpha_3\}^T$, and they can be expressed as a product of the matrix of coarse points C by the weights M. In the specific case of figure A.1 the stencil for the particular α row is M^3, and it will be called only M, the same procedures applying to the other α rows. Actually, the other two position of x are used only for staggered stencils, in general only the points coming from the refinement of $c_{2,2}$ are used,

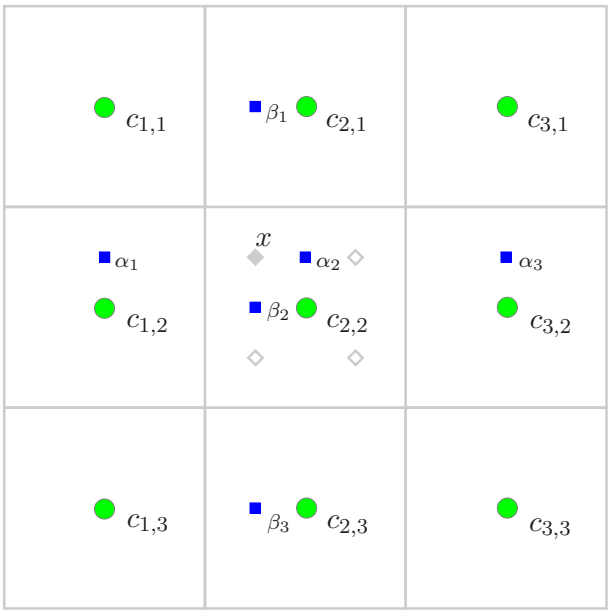


Figure A.2: Interpolation scheme for the cell centered bi-quadratic interpolation. Green: stencil; blue: intermediate interpolation points; grey: fine points.

coming from a combination of M^3 interpolations.

$$\begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{pmatrix} = \begin{pmatrix} c_{1,1} & c_{2,1} & c_{3,1} \\ c_{1,2} & c_{2,2} & c_{3,2} \\ c_{1,3} & c_{2,3} & c_{3,3} \end{pmatrix} \begin{pmatrix} m_1 \\ m_2 \\ m_3 \end{pmatrix} \tag{A.2}$$

or, in a compact form, $A = CM$. In the same way, $B = C^T M$. The coefficient x can be expressed as the quadratic (figure A.1) interpolation coming from the α points or from the β points:

$$\begin{cases} x_\alpha = M^T C M \\ x_\beta = M^T C^T M \end{cases} \tag{A.3}$$

The final value of x can then be approximated by the mean of the x_α and x_β values

$$x = \frac{1}{2}\left[M^T C M + M^T C^T M\right] \tag{A.4}$$

The weights will be contained into a 9×9 matrix W (one element for each coarse point) obtained from a combination of vectors M_3

$$W = \frac{1}{2}\left[M_3^T M_3 + M_3^T M_3\right] \tag{A.5}$$

This is valid for the point $\phi_{i-\frac{1}{4},j+\frac{1}{4}}$ child of $\phi_{i,j}$. Each other fine point, child of any of the nine coarse points, will have a similar formulation with a different combination of the vectors M_1, M_2, M_3. The generic fine point $x_{m,n}$ with $m, n = 1...6$ coordinates of the 36 possible position (four positions around nine coarse points) is given by

$$x_{m,n} = \frac{1}{2}\left[M_m^T C M_n + M_n^T C M_m\right] \tag{A.6}$$

2 Balsara divergence preserving interpolation

A divergence-free, spatially second order accurate reconstruction of the magnetic field in two dimensions is developed in Balsara [7]. This reconstruction strategy is useful for two dimensional AMR codes with zero divergence constraint on face centered vector fields. The three dimensional extension of these ideas follows trivially. The main principle of its reconstruction is to build polynomial interpolation of coarse values, using the $\nabla.\boldsymbol{u} = 0$ condition to find the interpolation coefficients. Following its developments, the left and right coarse u components of the i, j cell are denoted as $U^+ = u_{i+\frac{1}{2},j}$ and $U^- = u_{i-\frac{1}{2},j}$. Likewise, $V^+ = v_{i,j+\frac{1}{2}}$ and $V^- = v_{i,j-\frac{1}{2}}$. In order to make a second order accurate reconstruction of the velocity piecewise limited linear profiles are constructed in each face.

$$\begin{cases} \Delta_y U^{\pm} = \text{minmod}\left(u_{i\pm\frac{1}{2},j+1} - u_{i\pm\frac{1}{2},j}, u_{i\pm\frac{1}{2},j} - u_{i\pm\frac{1}{2},j-1}\right) \\ \Delta_x V^{\pm} = \text{minmod}\left(v_{i+1,j\pm\frac{1}{2}} - v_{i,j\pm\frac{1}{2}}, v_{i,j\pm\frac{1}{2}} - v_{i-1,j\pm\frac{1}{2}}\right) \end{cases} \tag{A.7}$$

Different limiters can be used in place of the minmod limiter. The piecewise-linear WENO interpolation from Jiang [46] provide a form of non-oscillatory reconstruction. The variation of u and v in the upper and lower x and y zone faces is given by (considered the x,y origin in the cell center):

$$\begin{cases} u(x = \pm\Delta x/2, y) = U^{\pm} + \dfrac{\Delta_y U^{\pm}}{\Delta y} y \\ v(x, y = \pm\Delta x/2) = V^{\pm} + \dfrac{\Delta_x V^{\pm}}{\Delta x} x \end{cases} \tag{A.8}$$

The problem of reconstructing the velocity field components in the zone given by $[-\Delta x/2], [\Delta x/2] \times [-\Delta y/2], [\Delta y/2]$ reduces to fitting functions on the interior of the zone so that they match the four linear profiles for Bx and By given above in the x and y zone faces. The two polynomial functions, one for u and one for v, must have the special property that their divergence is exactly zero at all points within the zone. Consider quadratic interpolation the values of u and v become:

$$\begin{cases} u(x, y) = a_0 + a_x x + a_y y + a_{xx} x^2 + a_{xy} xy + a_{yy} y^2 \\ v(x, y) = b_0 + b_x x + b_y y + b_{xx} x^2 + b_{xy} xy + b_{yy} y^2 \end{cases} \tag{A.9}$$

Having to fit linear profiles on the faces, two coefficients are immediately ignored:

$$a_{yy} = b_{xx} = 0 \tag{A.10}$$

Three more constraints are given by the divergence-free condition

$$a_x + b_y = 0 \tag{A.11}$$
$$2a_{xx} + b_{xy} = 0 \tag{A.12}$$
$$a_{xy} + b_{yy} = 0 \tag{A.13}$$

The divergence-free aspect of the velocity field requires an integral constraint to be satisfied in discrete form. That constraint is given by

$$(U^+ - U^-)\Delta y + (V^+ - V^-)\Delta x = 0 \tag{A.14}$$

Further algebraic operations lead to the expressions for the seven lasting coefficients:

$$a_x = -b_y = \frac{U^+ - U^-}{\Delta x} = -\frac{V^+ - V^-}{\Delta y} \tag{A.15}$$

$$a_y = \frac{1}{2}\left(\frac{\Delta_y U^+}{\Delta y} + \frac{\Delta_y U^-}{\Delta y}\right) \tag{A.16}$$

$$b_x = \frac{1}{2}\left(\frac{\Delta_x V^+}{\Delta x} + \frac{\Delta_x V^-}{\Delta x}\right) \tag{A.17}$$

$$a_{xy} = -2b_{yy} = \frac{1}{\Delta x}\left(\frac{\Delta_y U^+}{\Delta y} - \frac{\Delta_y U^-}{\Delta y}\right) \tag{A.18}$$

$$b_{xy} = -2a_{xx} = \frac{1}{\Delta y}\left(\frac{\Delta_x V^+}{\Delta x} - \frac{\Delta_x V^-}{\Delta x}\right) \tag{A.19}$$

$$a_0 = \frac{U^+ + U^-}{2} - a_{xx}\frac{\Delta x^2}{4} \tag{A.20}$$

$$b_0 = \frac{V^+ + V^-}{2} - b_{yy}\frac{\Delta y^2}{4} \tag{A.21}$$

This reconstruction can now be used in an area-weighted sense to prolong the velocity field to the faces of any refined zone that lies within the zone being considered. If the same reconstruction strategy is applied to two zones that share a face then it will produce the same linear profile for the velocity field component on the shared face. As a result, the refined zone to which we want to prolong velocity field components can even have faces that coincide with coarse zone faces, avoiding coarse-fine divergence mismatch.

Bibliography

[1] A C Calder BF, Plewa T, Rosner R, Dursi L, Weirs V, Dupont T, Robey H, Kane J, Remington B, Drake R, Dimonte G, Zingale M, Timmes F, Olson K, Ricker P, MacNeice P, Tufo H (2002) On validating an astrophysical simulation code. Astrophys J Suppl Ser 143:201–229

[2] Almgren AS, Bell JB, Colella P, Howell LH (1993) An adaptive projection method for the incompressible euler equations. In: In 11th AIAA Computational Fluid Dynamics Conference, Orlando, FL

[3] Almgren AS, Bell JB, Colella P, Howell LH, Welcome ML (1998) A conservative adaptive projection method for the variable density incompressible navier-stokes equations. Journal of Computational Physics 142(1):1–46

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

[5] B Lafaurie RSSZ C Nardone, Zanetti G (1994) Modelling merging and fragmentation in multiphase flows with surfer. Journal of Computational Physics 113:134–147

[6] Bai D, Brandt A (1987) Local mesh refinement multilevel techniques. SIAM Journal on Scientific and Statistical Computing 8(2):582–598

[7] Balsara DS (2001) Divergence-free adaptive mesh refinement for magnetohydrodynamics. Journal of Computational Physics 174(2):614–648

[8] Bell J, Colella P, Glaz H (1989) A second order projection method for the incompressible navier-stokes equations. Journal of Computational Physics 85:257–283

[9] Berger MJ (1982) Adaptive mesh refinement for hyperbolic partial differential equations. PhD thesis, Stanford University

[10] Bhaga D, Weber ME (1981) Bubbles in viscous liquid: shapes, wakes and velocities. Journal of Fluid Mechanics 105:61–85

[11] Borthwick A, Leon S, Jozsa J (2002) Adaptive quadtree model of shallow-flow hydrodynamics. Journal of Hydraulic Research 39(4):413–424

[12] Brackbill J, Kothe D, Zemach C (1992) A continuum method for modeling surface tension. Journal of Computational Physics (100)

[13] Brown JD, Lowe LL (2005) Multigrid elliptic equation solver with adaptive mesh refinement. Journal of Computational Physics 209(2):582–598

[14] Cahn JW, Hilliard JE (1958) Free energy of a nonuniform system. i. interfacial energy. Journal of Chemical Physics (28):258

[15] Carvalho I, Heitor M, Santos D (2002) Liquid film disintegration regimes and proposed correlations. International journal of multiphase flows 28:773–789

[16] Chorin A (1968) Numerical solution of the navier-stokes equations. Mathematics of Computation 22(104):745–762

[17] Clift R, Grace JR, Weber ME (1978) Bubbles, Drops and Particles. Academic Press

[18] Couderc F (2007) Développement d'un code de calcul pour la simulation d'écoulements de fluides non miscibles application á la désintégration assistée d'un jet liquide par un courant gazeux. PhD thesis, ENSAE

[19] Couderc F, Estivalezes JL (2003) Numerical simulation of the air-blasted liquid sheet: development of a dns solver based on the level-set method. ILASS Europe Notthigham

[20] Daly B (1967) Numerical study of two fluid rayleigh-taylor instability. Physics of Fluids 10:297–307

[21] Daly B, Pracht A (1967) Numerical study of density current surges. Physics of Fluids 11:15–30

[22] DeBar R (1974) A method in two-d eulerian hydrodynamics. Technical Report UCID-19683, Lawrence Livermore National Laboratory

[23] Delage S, Vincent S, Caltagirone JP, Heliot JP (2006) A hybrid linking approach for solving the conservation equations with an adaptive mesh refinement method. Journal of Computational and Applied Mathematics 191:280–296

[24] Desjardin O, Moureau V, Pitsch H (2008) An accurate conservative level set/ghost fluid method for simulating turbulent atomization. Journal of Computational Physics 227(18):839—8416

[25] DeZeeuw D, Powell KG (1992) An adaptative cartesian mesh for the euler equations. Journal of Computational Physics (245)

[26] Dreher J, Grauer R (2005) A parallel mesh-adaptive framework for hyperbolic conservation laws. Parallel Computing 31:913–932

[27] Duarte F, Gormaz R, Natesan S (2004) Arbitrary lagrangian-eulerian method for navier-stokes equations with moving boundaries. computer methods in applied mechanics and engineering. Computer Methods in Applied Mechanics and Engineering 193:4819–4836

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

[29] Fedkiw R, Xu S (1998) The ghost fluid method for for viscous flows. *In M Hafez, editor,* Progress in Numerical Solutions of Partial Diferential Equations, *Arcachon, France*

[30] Fedkiw R, Aslam T, Xu S (1999) The ghost fluid method for deflagration and detonation discontinuities. Journal of Computational Physics 154:393–427

[31] Fernandez VG (2009) Experimental study of a liquid sheet disintegration in a high pressure environment. PhD thesis, ISAE

[32] Fuster D, Bagu A, Boeck T, LeMoyne L, Leboissetier A, Popinet S, Ray P, Scardovelli R, Zaleski S (2009) Simulation of primary atomization with an octree adaptive mesh refinement and vof method. International Journal of Multiphase Flow 35:550–565

[33] Gorokhovski M, Herrmann M (2008) Modeling primary atomization. Annual Review of Fluid Mechanics 40:343–366

[34] Gottlieb S, Shu C (1998) Total variation diminishing runge-kutta schemes. Mathematics of Computation 67(221):73–85

[35] Guermond J, Minev P, Shen J (2006) An overview of projection methods for incompressible flows. Computer Methods in Applied Mechanics and Engineering 195:6011–6045

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

[37] Hagerty W, Shea J (1959) A study of the stability of plane fluid sheets. ASME Transactions: Journal of Applied Mechanics 22:509–514

[38] Harlow F, Welch J (1965) Numerical calculation of time-dependent viscous incompressible,flow of fluid with free surface. Physics of Fluids 8:2182–2189

[39] Hirt C, Nichols B (1981) Volume of fluids (vof) method for the dynamics of free boundaries. Journal of Computational Physics 39:201–225

[40] Hirt C, Cook JL, , Butler TD (1970) A lagrangian method for calculating the dynamics of an incompressible fluid with free surface. Journal of Computational Physics 5(1):103–124

[41] Hornung RD, Kohn SR (2002) Managing application complexity in the samrai object-oriented framework. Concurrency and Computation: Practice and Experience 14:347–368

[42] Howell LH (1993) A multilevel adaptive projection method for unsteady incompressible flow. In: Proceedings of the Sixth Copper Mountain Conference on Multigrid Methods, pp 243–257

[43] Howell LH, Bell JB (1996) An adaptive-mesh projection method for viscous incompressible flow. SIAM Journal of Scientific Computing 18:996–1013

[44] Hysing S, Turek S, Kuzmin D, Parolini N, Burman E, Ganesan S, Tobiska L (2008) Quantitative benchmark computations of two-dimensional bubble dynamics. MOX-Report 23

[45] Jazayeri S, Li X (2000) Nonlinear instability of plane liquid sheets. Journal of Fluid Mechanics 406:281–308

[46] Jiang G, Peng D (2000) Weighted eno schemes for hamilton-jacobi equations. SIAM Journal on scientific computing (21(6)):2126–2143

[47] Kang M, Fedkiw R, Liu XD (2000) A boundary condition capturing method for multiphase incompressible flow. Journal of Scientific Computing 15:143–173

[48] Kang M, Fedkiw R, Liu XD (2001) A second order accurate symmetrical discretization of the poisson equation in irregular domains. Journal of Computational Physics 176:205–227

[49] Lamb H (1975) Hydrodynamics. Cambridge University Press

[50] Lebas R, Menard T, Beau P, Berlemont A, Demoulin F (2009) Numerical simulation of primary break-up and atomization: Dns and modelling study. International Journal of Multiphase Flow (35):247–260

[51] Lefebvre A (1989) Atomization and sprays lecture notes. Combustion an International Series.

[52] LeVeque R, Li Z (1994) The immersed interface method for elliptic equations with discontinuous coefficients and singular sources. SIAM Journal on Numerical Analysis 31:1019–1044

[53] LeVeque R, Li Z (1997) Immersed interface method for stokes flow with elastic boundaries or surface tension. SIAM Journal of Computational Physics 18:709–735

[54] Liu Y, Vinokur M, Wang Z (2006) Spectral difference method for unstructured grids: Basic formulation. Journal of Computational Physics 216:780–801

[55] Losasso F, Gibou F, Fedkiw R (2004) Simulating water and smoke with an octree data structure. ACM Trans Graph (SIGGRAPH Proc) 23:457–462

[56] Lozano A, Barreras F, Hauke G, Dopazo C (2001) Longitudinal instabilities in a air-blasted liquid sheet. Journal of Fluid Mechanics 437:143–173

[57] M Sussman YY M Ohta (2002) Numerical simulation of bubble motion by a coupled level set/vof method. In: Proceedings of the 35th Annual Meeting of the Society of Chemical Engineers

[58] Mansour A, Chigier N (1990) Disintegration of liquid sheets. Physics of Fluids A: Fluid Dynamics 2:706–719

[59] Mansour A, Chigier N (1991) Dynamic behavior of liquid sheets. Physics of Fluids A: Fluid Dynamics 3:2971–2980

[60] Marmottant P, Villermaux E (2004) On spray formation. Journal of Fluid Mechanics 498:73–111

[61] Martin DF (1991) An adaptive cell-centered projection method for the incompressible navier-stokes equations. PhD thesis, BSME University of Florida

[62] Martin DF, Cartwright KL (1996) Solving poisson equation using adaptive mesh refinement. Journal of Computational Physics

[63] Martin DF, Colella P (2000) A cell-centered adaptive projection method for the incompressible euler equations. Journal of Computational Physics 163:271–312

[64] Martin DF, Colella P, Graves D (2007) A cell-centered adaptive projection method for the incompressible navier-stokes equations in three dimensions. Journal of Computational Physics pp 1863–1886

[65] McCormick SF (1987) Multigrid Methods. Society for Industrial and Applied Mathematics

[66] Menard T (2007) Développement d'une méthode level set pour le suivi d'interface. application de la rupture de jet liquide. PhD thesis, Faculté de sciences de l'université de Rouen

[67] Menard T, Tanguy S, Berlemont A (2007) Coupling level set/vof/ghost fluid methods: Validation and application to 3d simulation of the primary break-up of a liquid jet. International Journal of Multiphase Flow 33(5):510–524

[68] Mihalef V, Sussman M, Metaxas D (2007) The marker level-set method : a new approach to computing accurate interfacial dynamics., not published

[69] Min C, Gibou F (2006) A second order accurate projectionmethod for the incompressible navier-stokes equation on non-graded adaptive. Journal of Computational Physics 219:912–929

[70] Min C, Gibou F, Ceniceros H (2006) A supra-convergent finite difference scheme for the variable coefficient poisson equation on nongraded grids. Journal of Computational Physics 218:123–140

[71] Minion M, Brown D, Cortez R (2001) Accurate projection methods for the incompressible navier-stokes equations. Journal of Computational Physics 168:464–499

[72] Moreau V, Desjardins O (2008) A second-order ghost-fluid method for the primary atomization of liquid fuel in air-blast type injectors. In: Proceedings of the Summer Program 2008, Center for Turbulence Research, Stanford University

[73] Morris JP (2000) Simulating surface tension with smoothed particle hydrodynamics. International Journal for Numerical Methods in Fluids 33(3):333–353

[74] Nguyen V, Peraire J, Khoo BC, Persson P (2010) A discontinuous galerkin front tracking method for two phase flows with surface tension. Computer & Fluids 39:1–14

[75] Noh WF, Woodward P (1976) SLIC /simple line interface calculation/. In A. I. VanDeVooren and P. J. Zandbergen editors, Some Methods of Resolution of Free Surface Problems, volume 59 of Lecture Notes in Physics, Berlin Springer Verlag

[76] Oevermann M, Berger GM (2000) A projection method for two-phase incompressible flow with surface tension and sharp interface resolution. Zuse Institute Berlin - Report 17

[77] Olson K (2006) Paramesh: A parallel adaptive grid tool. in Parallel Computational Fluid Dynamics 2005: Theory and Applications: Proceedings of the Parallel CFD Conference, College Park, MD, USA, eds A Deane, A Ecer, G Brenner, D Emerson, J McDonough, J Periaux, N Satofuka, and D Tromeur-Dervout (Elsevier)

[78] Olson K, MacNeice P (2005) An over of the paramesh amr software and some of its applications. in Adaptive Mesh Refinement-Theory and Applications, Proceedings of the Chicago Workshop on Adaptive Mesh Refinement Methods, Series: Lecture Notes in Computational Science and Engineering, eds T Plewa, T Linde, and G Weirs (Berlin: Springer) 41

[79] Osher S, Fedkiw R (2003) Level Set Method and Dynamic Implicit Surfaces, vol 153. Springer Applied Mathematical Sciences

[80] Osher S, Sethian J (1988) Fronts propagating with curvature dependant speed : Algorithms based on hamilton-jacobi formulations. Journal of Computational Physics 79:12–49

[81] Osher S, Shu CW (1991) High-order essentially non-oscillatory schemes for hamilton-jacobi equations. SIAM Journal of Numerical Analysis 28(4):907–922

[82] Pancheshnyi S, Ségur P, Capeillére J, , Bourdon A (2008) Numerical simulation of filamentary discharges with parallel adaptive mesh refinement. Journal of Computational Physics 227:6574–6590

[83] Parker B, Youngs D (1992) Two and three dimensional eulerian simulation of fluid flowwith material interfaces. Technical report, UK Atomic Weapons Establishment, Aldermaston

[84] Peskin C (1977) Numerical analysis of blood flow in the heart. Journal of Computational Physics 25:220–252

[85] Peter MacNeice CMRd Kevin M Olson, Packer C (2000) Paramesh : A parallel adaptive mesh refinement community toolkit. Computer Physics Communications 126

[86] Pilliod J (1992) An analysis of piecewise linear interface reconstruction algorithms for volumeof- fluid methods. PhD thesis, PhD thesis, University of California, Davis

[87] Popinet S (2009) An accurate adaptive solver for surface tension driven interfacial flows. Journal of Computational Physics 228:5838–5866

[88] Popinet S, Zaleski S (1999) A front tracking algorithm for accurate representation of surface tension. International Journal for Numerical Methods in Fluids 30:775–793

[89] Popinet S, Zaleski S (2003) Gerris: a tree- based adaptive solver for the incompressible euler equations in complex geometries. Journal of Computational Physics 190(2):572–600

[90] Powell KG, Roe P, Linde T, Gombosi TI, DeZeeuw DL (1999) A solution-adaptive upwind scheme for ideal magnetohydrodynamics. Journal of Computational Physics 154:284–309

[91] Prosperetti A (1981) Motion of two superposed viscous fluids. Physics of fluids 24:1217–1223

[92] Puckett E, Almgren A, JB Bell DM, Rider W (1997) A high order projection method for tracking fluid interfaces in variable density incompressible flows. Journal of Computational Physics 100:269–282

[93] Ramaswamy B, Kawahara M (1987) Arbitrary lagrangian-eulerian finite element method for unsteady, convective, incompressible viscous free surface fluid flow. International Journal for Numerical Methods in Fluids 7(10):953–984

[94] Rangel R, Sirignano W (1991) The linear and nonlinear shear instability of a fluid sheet. Physics of Fluids A : Fluid Dynamics 3:2392–2400

[95] Rendleman C, Lijewski V (2000) Parallelization of an adaptive mesh refinement method for incompressible fluid flows. In: Inernational Parallel Distributed Processing Symposium

[96] Ricker P (2007) A direct multigrid poisson solver for oct-tree adaptive meshes. The Astrophysical Journal Suppl 176:293

[97] Rider WJ, Kothe DB (1998) Reconstructing volume tracking. Journal of Computational Physics 141(2):112–152

[98] Saltzman J, Puckett E (1992) A 3-d adaptive mesh refinement algorithm for multimaterial gas dynamics. Physica D 60:84–104

[99] Santacreu SD (2006) Méthode de raffinement de maillage adaptatif hybride pour le suivi de fronts dans des coulements incompresibles. PhD thesis, Université Bordeaux I

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

[101] Schaffer S (1998) A semicoarsening multigrid method for elliptic partial differential equations with highly discontinuous and anisotropic coefficients. SIAM Journal of Scientific Computing 20:228–242

[102] Sethian J (1999) Level-set methods and fast marching methods. Cambridge University Press

[103] Shin S, Juric D (2002) Modeling three-dimensional multiphase flow using a level contour reconstruction method for front tracking without connectivity. Journal of Computational Physics 180:427–470

[104] Shu C (1997) Essentially non-oscillatory and weighted essentially non-oscillatory schemes for hyperbolic conservation laws. Technical report, ICASE - NASA Report

[105] Shu C, Gottlieb S (1998) Total variation diminishing runge-kutta schemes. Mathematics of Computation (67(221)):73–85

[106] Shu C, Jiang G (1996) Efficient implementation of weighted eno schemes. Journal of Computational Physics 126:202–228

[107] Shu C, Osher S (1989) Efficient implementation of essentially non-oscillatory shock capturing schemes. Journal of Computational Physics 83:32–78

[108] Squire H (1953) Investigation of the instability of a moving film. British Journal of Applied Physics (4):167–169

[109] Stapper B, Samuelsen G (1990) An experimental study of the breakup of a two dimensional liquid sheet in the presence of co-flow air shear. In: In AIAA Paper 89-0461

[110] Stapper B, Sowa W, Samuelsen G (1992) An experimental study of the effects of liquid properties on the breakup of a two-dimensional liquid sheet. Journal of Engineering for Gas Turbines and Power (114):32–39

[111] Sussman M (2005) A parallelized, adaptive algorithm for multiphase flows in general geometries. Computers and Structures 83(6-7):435–444

[112] Sussman M, Puckett E (2000) A coupled level set and volume of fluid method for computing 3d and axisymmetric incompressible two-phase flow. Journal of Computational Physics 162:301–337

[113] Sussman M, Smereka P, Osher S (1994) A level set approach for computing solutions to incompressible two-phase flow. Journal of Computational Physics 114:146–159

[114] Sussman M, Almgren AS, Bell JB, Colella P, Howell LH, Welcome ML (1999) An adaptive level set approach for incompressible two-phase flows. Journal of Computational Physics 148:81–124

[115] Sussman M, Smith K, Hussaini M, Ohta M, Zhi-Wei R (2006) A sharp interface method for incompressible two-phase flow. Journal of Computational Physics 221:469–505

[116] Tanguy S (2004) Développement d'une methode de suivi d'interface application aux écoulements diphasiques. PhD thesis, Faculté des sciences de l'université de Rouen

[117] Taylor G (1959) The dynamics of thin sheets of fluid. i. water bells. Proc R Soc Lond (253):280–296

[118] Teigland R, Eliassen IK (2001) A multiblock/multilevel mesh refinement procedure for cfd computations. International journal for numerical methods in fluids 36:519–538

[119] Temam R (1969) Sur l'approximation de la solution des équations de navier-stokes par la méthode des pas fractionnaires. Archive for Rational Mechanics and Analysis (33):377–385

[120] Toth G, Roe PL (2002) Divergence- and curl-preserving prolongation and restriction formulas. Journal of Computational Physics (180):736–750

[121] Trontin P (2009) Développement d'une approche de type les pour la simulation d'écoulements diphasiques avec interface. application á l'átomisation primaire. PhD thesis, Institut Supérieur de lÁéronautique et de l'Espace

[122] Trottenberg U, Oosterlee CW, Schuller A (2001) Multigrid. Academic Press

[123] Unverdi S, Tryggvason G (1992) A front-tracking method for viscous, incompressible, multifluid flows. Journal of Computational Physics (100):25–37

[124] VanDerHolst B, Keppens R (2007) Hybrid block-amr in cartesian and curvilinear coordinates: Mhd applications. Journal of Computational Physics 226:925–946

[125] VanDerPijl SP, Segal A, Vuik C, Wesseling P (2005) A mass-conserving level-set method for modelling of multi-phase flows. International Journal on Numerical Methods in Fluids 47:339–361

[126] VanDerVorst HA (1992) Bi-cstab: a fast and smoothly converging variant of bi-cg for the solution of nonsymmetric linear systems. SIAM Journal of Scientific Computing 12(2):631–644

[127] Vincent S, Caltagirone JP (2000) A one-cell local multigrid method for solving unsteady incompressible multiphase flows. Journal of Computational Physics 163:172–215

[128] Wadell H (1933) Volume, shape and roundness of rock particles. Journal of Geology (40):443–451

[129] Wagner C (1998/99) Introduction to algebraic multigrid. Course notes of an algebraic multigrid course at the University of Heidelberg

[130] Wang Z (2002) Spectral (finite) volume method for conservation laws on unstructured grids, basic formulation. Journal of Computational Physics (178):210–251

[131] York J, Stubbs H, Tek M (1953) The mechanisms of disintegration of liquid sheets. In: Transactions of the ASME, pp 1279–1286

[132] Zalesak S (1979) Fully multidimensional flux-corrected transport for fluids. Journal of Computational Physics (31):335–362

[133] Zeng S, Wesseling P (1994) Multigrid solution of the incompressible navier-stokes equations in general coordinates. SIAM journal on numerical analysis (31):1764–1784