Direct Numerical Simulation of bubbles with Adaptive Mesh Refinement with Distributed Algorithms

Arthur Talpaert

▶ To cite this version:

Arthur Talpaert. Direct Numerical Simulation of bubbles with Adaptive Mesh Refinement with Distributed Algorithms. Numerical Analysis [math.NA]. Université Paris Saclay (COmUE), 2017. English. NNT : 2017SACLX016 . tel-01531784

HAL Id: tel-01531784
https://pastel.hal.science/tel-01531784

Submitted on 1 Jun 2017

Thèse de doctorat de l'Université Paris-Saclay préparée à l'École polytechnique

École doctorale n° 573
Interfaces
Spécialité de doctorat : Mathématiques appliquées
par

M. Arthur Talpaert

Simulation numérique directe de bulles sur maillage adaptatif avec algorithmes distribués

Thèse présentée et soutenue à l'INSTN de Saclay, le 24 février 2017.

Composition du Jury :

M.	Nicolas Seguin	Professeur Université de Rennes 1	(Président du jury)
M.	Tony Lelièvre	Professeur École des Ponts ParisTech	(Rapporteur)
M.	Frédéric Lagoutière	Professeur Université Claude Bernard Lyon 1	(Rapporteur)
M.	Grégoire Allaire	Professeur École polytechnique	(Directeur)
M.	Stéphane Dellacherie	Ingénieur Hydro-Québec	(Co-directeur)
M.	Bruno Després	Professeur Université Paris 6	(Examinateur)
M.	Yohan Penel	Chercheur CEREMA	(Examinateur)
M.	Anouar Mekkas	Ingénieur CEA	(Examinateur)

Abstract

This PhD work presents the implementation of the simulation of two-phase flows in conditions of water-cooled nuclear reactors, at the scale of individual bubbles. To achieve that, we study several models for Thermal-Hydraulic flows and we focus on a technique for the capture of the thin interface between liquid and vapour phases. We thus review some possible techniques for Adaptive Mesh Refinement (AMR) and provide algorithmic and computational tools adapted to patch-based AMR, which aim is to locally improve the precision in regions of interest. More precisely, we introduce a patch-covering algorithm designed with balanced parallel computing in mind. This approach lets us finely capture changes located at the interface, as we show for advection test cases as well as for models with hyperbolic-elliptic coupling. The computations we present also include the simulation of the incompressible Navier-Stokes system, which models the shape changes of the interface between two non-miscible fluids.

Keywords Nuclear Thermal-Hydraulics, two-phase flows, numerical simulation of bubbles, low-Mach conditions, Adaptive Mesh Refinement (AMR), patch covering algorithms, multilevel, parallel computing, multiprocessing, numerical schemes, Després-Lagoutière, limited downwind advection scheme, Navier-Stokes equations, hyperbolic-elliptic coupling, lid-driven cavity, surface tension.

Résumé

Ce travail de thèse présente l'implémentation de la simulation d'écoulements diphasiques dans des conditions de réacteurs nucléaires à caloporteur eau, à l'échelle de bulles individuelles. Pour ce faire, nous étudions plusieurs modèles d'écoulements thermohydrauliques et nous focalisons sur une technique de capture d'interface mince entre phases liquide et vapeur. Nous passons ainsi en revue quelques techniques possibles de maillage adaptatif (AMR) et nous fournissons des outils algorithmiques et informatiques adaptés à l'AMR par patchs dont l'objectif est d'améliorer localement la précision dans des régions d'intérêt. Plus précisément, nous introduisons un algorithme de génération de patchs conçu dans l'optique du calcul parallèle équilibré. Cette approche nous permet de capturer finement des changements situés à l'interface, comme nous le montrons pour des cas tests d'advection ainsi que pour des modèles avec couplage hyperbolique-elliptique. Les calculs que nous présentons incluent également la simulation du système de Navier-Stokes incompressible qui modélise la déformation de l'interface entre deux fluides non-miscibles.

Mots clefs Thermohydraulique nucléaire, écoulements diphasiques, simulation numérique de bulles, conditions bas-Mach, maillage adaptatif (AMR), algorithmes de recouvrement par patchs, multi-niveau, calcul parallèle, parallélisme informatique, schémas numériques, Després-Lagoutière, schéma d'advection à limitateur de flux aval, équations de Navier-Stokes, couplage hyperbolique-elliptique, cavité entraînée, tension de surface.

Acknowlegdements

More often than not, I would stare at my cooking pan before pouring my dinner ingredients in it. The liquid water would slowly warm up. Had I put salt, I would soon start to see convective rings. The water would later simmer imperceptibly and suddenly boil in large bubbles. They would rise and quickly merge with the surface. Like a child hunting for familiar shapes in the clouds, I would distinguish balloons, slugs, jellyfish... nuclear reactor hazards. I would think: “Why is my job following me up until right inside my kitchen?”

The P for PhD should stand for Personal. I define it as the strong combination of a very individual effort done of one’s own, regularly enlightened by human relationships. I have the impression this is because this type of research is the dual embodiment of furthering one’s academic education and performing a laboratory job. Therefore, I experienced a very personal – even sometimes solitary – work together with fruitful exchanges. I would like to take the opportunity of this manuscript to acknowledge these said exchanges.

First, I would like to thank my adviser **Stéphane**. I met him personally well before the start of my PhD. Soon he explained to me how he wanted to work with me and how much he was hoping for this work. In three years, he showed me multiple aspects of his personality: leadership, understanding, demand, pedagogy, humanity. I am thankful for everything he is and everything he did, no exception.

Second, I am grateful to **Grégoire**. I have the feeling he played an increasing role during the three years. With time, we took the time for solving problems together and for knowledge transmission, based on mutual trust. This is a work experience I greatly enjoyed. As for Stéphane, his co-workers – and in my case his PhD student – undoubtedly sense humble but actual talent.

Anouar is someone I also met long before the beginning of my thesis.

Countless time, I saw him with a lot of responsibilities and still, he often accepted to spend long hours with me on my code. He taught me a lot about practical Computer Science and suffered with me the long hunts for bugs and other memory leaks. Thank you.

I also acknowledge the leadership and expertise of **Samuel**. I appreciated how original his thinking is. He is quick to imagine new perspectives which helped me come to crucial solutions.

I would like to thank the very eclectic set of people who convinced me to apply for a PhD Candidate position. In chronological order, my former professors (Bernard Bonin, Franck Carré, Tomasz Kozlowski), **my kind and loving parents**, my German co-workers of AREVA (Stefan Nießen and others) all talked me into this challenge. The position was made possible due to the double financing of the CEA and the DGA. The CEA in particular was a welcoming structure for research in full freedom. I would like to thank my hierarchical superiors Didier Jamet and Edwige Richebois, as well as Jacques Segré and Danielle Gallo-Lepage.

I was very lucky to have a wonderful work environment with a very friendly team. I sincerely thank my nice colleagues Olivier, Mathieu, Sandrine, Yannick, Antoine, Sébastien, Marie-Claude, Marc E., Marc T., Erwan, Estelle, Adrien, Pascal; all of which were sometimes very personal. I have been very happy with the relationships I had with my colleagues from **the STMF service**. A special thanks goes to Michaël Ndjinga: he was a real CDMATH "tech-evangelist" and I enjoyed his forward-thinking talks.

I shared so much with the other PhD students, they have a special place in my acknowledgements: **Thi-Phuong Kieu, Thibaud, Alexandre**. Each in their own way, they showed me how the PhD looked like. By relying on the predecessors and their literature testimony, you would have to learn how to pose and solve a problem all by yourself, hopefully for the first time in history if you are talented enough. They warned me about challenges I had not foresawn: the thesis may be very solitary, often with no immediate rewarding, occasionally with failure. But the reward would be to be able to set up and develop your own project, seeing it bloom month after month.

I want to say I was very glad to be a part of **the applied maths community**. I am not afraid to say that computational scientists are likely to be some of the friendliest. I had the opportunity to participate in congresses and in the CEMRACS. I made friends with other students as Déna from Paris VI or Anya from CEA Cadarache for instance. I was also delighted to be familiar with other professors and researchers as Frédérique Charles and

Frédéric Lagoutière.

Last but defiantly foremost, I would like to express my deep gratitude to my dearest **Camille**. She brings me so much joy every day and I am so happy she carried our sweet son **Gabriel** to this world. I am really looking forward to continuing pursuing a lifelong vision with her.

Contents

List of Figures

List of Tables

Nomenclature

$\star_0$	As a subscript: relative to the initial conditions
$\mathbb{1}$	Indicator function
A	Matrix for predicted velocity
α	Void fraction
$\mathbb{A}$	Operator for the computation of the AMR patch covering
b	Right hand side for predicted velocity
$\star_{BC}$	As a subscript: relative to boundary conditions
$\star^c$	As a superscript: relative to the composite grid
C	Matrix for potential
c	Speed of sound
d	Right hand side for potential
Δt	Time step
Δx	Elementary grid space
$\digamma$	Dilation factor
E	Specific energy
e	Internal energy
$\mathcal{EOS}$	Characteristic function of an equation of state: $\mathcal{EOS}(p, e, \rho) = 0$

η	Patch efficiency
$\widehat{\mathbf{e}}_{\mathbf{x}}$	Elementary unit vector along direction x
$\widehat{\mathbf{e}}_{\mathbf{y}}$	Elementary unit vector along direction y
$\widehat{\mathbf{e}}_{\mathbf{z}}$	Result of the cross-product $\widehat{\mathbf{e}}_{\mathbf{x}} \times \widehat{\mathbf{e}}_{\mathbf{y}}$
$\mathbf{F}$	Volumetric force
$\mathbf{f}$	Force acceleration
$\mathbb{F}$	Operator for initial conditions from an analytic formula
$\mathbf{g}$	Gravitational acceleration
$\star_g$	As a subscript: relative to the gas phase
γ	Average squareness
$\underline{\gamma}$	Ideal gas factor
GC	Set of ghost cells
$\star^H$	As a superscript: relative to the coarse grid
$\star^h$	As a superscript: relative to the fine grid
i	Space index along x
I	Space index along x, specifically for the coarse grid
$\mathbb{I}^h$	Interpolation from coarse to fine
ij	Matrix index
$iter$	Iteration
$iter_{LDC}$	Iteration of LDC algorithm
$iter_{Sch}$	Iteration of Schwarz algorithm
$iter_t$	Iteration of time

j	Space index along y
k	CFL stability coefficient
κ	Curvature
$\mathcal{L}$	Linear elliptic second-order differential operator
$\star_l$	As a subscript: relative to the liquid phase
L	Length
λ	Thermal transfer coefficient
$\mathcal{M}$	Mach number
$\mathbb{M}$	Set of matrices
M	Averaging operator
μ	Volumetric viscosity
n	Time index
n_{max}	Maximum number of cells in any direction of a patch
n_{min}	Minimum number of cells in any direction of a patch
N	Total number of cells of a patch
$\mathbf{n}$	Normal vector
ν	Kinematic viscosity
Ω	Computational domain
$\partial\Omega$	Border of the computational domain
$\mathbb{O}$	Time resolution to get the state $iter_t + 1$ from the state $iter_t$
$\mathbb{O}^r$	The $\mathbb{O}$ operator composed r times with a time step $\frac{\Delta t}{r}$
p	Pressure

P	Thermodynamic pressure
Π	Dynamic pressure
φ	Level-set function
ϕ	Potential
ψ	Pulsation
r	Refinement coefficient
R	Radius
$\mathcal{R}$	Reynolds number
$\mathbb{R}^H$	Restriction from fine to coarse
ρ	Volumetric mass
ϱ	Distance to the origin
s	Source
S	Surface
$\mathbb{S}$	Smoothing operator
Σ	Interface
σ	Normalised standard deviation
ς	Signature in a patch
su	Computation speed-up
t	Time
T	Temperature
$\mathcal{T}$	Duration
$\mathbf{t}$	Tangential vector

τ	Stress tensor
τ_{surf}	Surface tension coefficient
θ	Coefficient of thermal dilation
$\mathbf{T}_{surf}$	Surface tension
$\mathbf{u}$	Velocity
$\mathbf{U}$	Velocity parallel to faces
$\tilde{\mathbf{u}}$	Predicted velocity
V	Volume
$\mathbf{x}$	Spatial position
ξ	Spatial position in another referential
Y	Colour function
$\star_{\square}$	As a subscript: relative to the local refined area
$\star_{\bigcirc}$	As a subscript: relative to the non-refined area

Chapter 1

Introduction

1.1 Purpose of this research: two-phase flows in nuclear reactors

To a large extent, producing electricity with a nuclear power plant resembles a lot how one produces energy with other types of power plants, in particular fossil fuel plants like the ones using coal or gas. The base principle could be summarised as follows: using a thermal source, water is heated up, circulates and transmits its heat to a vapour phase, which then makes a turbine turn. This is a transmission of thermal energy into kinetic energy; the turbine creates electricity with a dynamo-effect and the electricity is transported to the customers. In the case of nuclear power plants, the thermal source takes its origin in the core of the power plant where the nuclear chain reaction takes place. In the end, what makes the Nuclear Engineering field a complex one is the combination of multiple physics fields: nuclear physics, thermal-dynamics, hydrodynamics, material science, solid mechanics, chemistry (e.g. for the fission products), electrical engineering and so forth. The engineering challenge of the scale of the phenomenons comes on top of that: extreme heat, flow rates and turbine rotation speeds for instance.

The Thermal-Hydraulics study of two-phase flows is a major topic for Nuclear Engineering because it contributes to improving the safety and the efficiency of reactors. It is the study of mixtures of liquid water and steam in nominal regime as well as in accidental regime. The first most common nuclear power plant design is the Pressurised Water Reactor (PWR). Figure 1.1 represents a commercial PWR nuclear power plant and Figure 1.2 shows

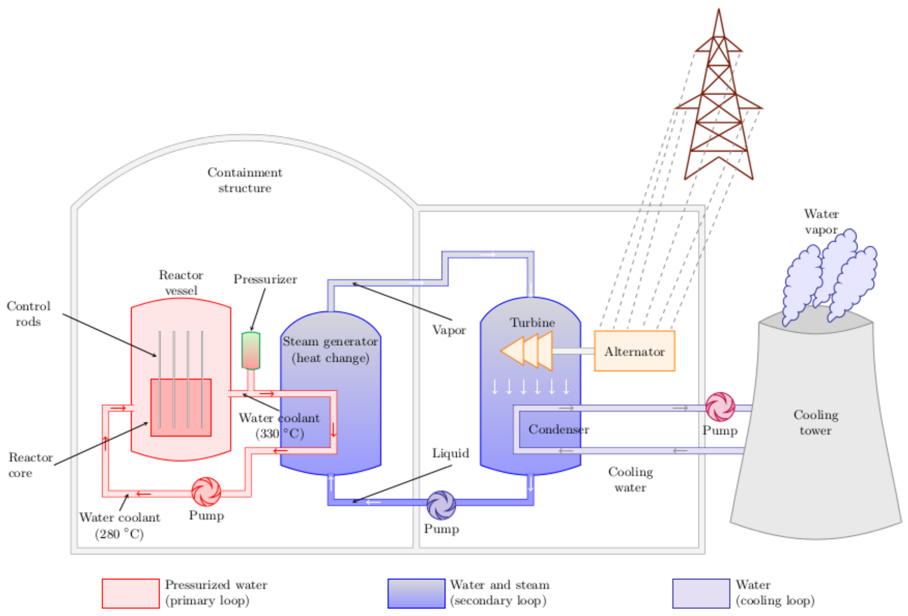


Figure 1.1 – Representation of a commercial nuclear plant [62]

the components for a second-generation submarine nuclear propulsion system. In red, we have the primary loop; the nuclear reaction takes place in the core and transmits the generated heat to the water. In blue, we have the secondary loop; including the steam generators where most if not all the boiling takes place. The two loops are separated with a physical barrier to avoid any eventual contamination by radioactive material detaching from the core. In a normal regime, the water flow inside the primary flow of a PWR is kept liquid thanks to the immense pressure. An eventual vapour phase may nonetheless appear in some conditions, like accidental ones or in the case of meta-stable bubbles from a thermal-dynamics point of view. On the opposite, steam is supposed to appear in the secondary loop, by design. So for safety reasons, it is important to study when and where the steam may appear and to analyse its behaviour.

The other most common nuclear power plant design is the Boiling Water Reactor (BWR). As its name indicates, the flow inside the core of a BWR is by design made of a mixture of liquid water together with steam. This is why it is essential to study two-phase flows in the core of PWRs and BWRs. Similarly, one of the components of PWRs in addition to the nuclear core is the steam generator. This is the location of the boiling, producing the steam which will make the turbines turn and hence produce electricity. For this component the purpose is to understand and predict precisely where the

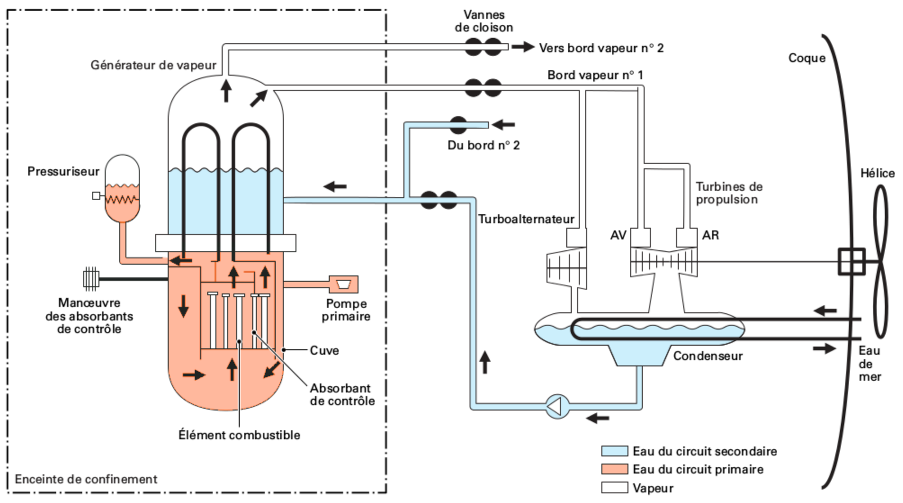


Figure 1.2 – Representation of a submarine propulsion system [68]

boiling takes place. In particular, the area called the Departure of Nucleate Boiling (DNB), where the first bubbles are born, is of foremost importance [27]. An accurate knowledge of its location would permit to increase efficiency. A finer comprehension of the mechanisms of a production unit and a better efficiency will lead to an optimisation of the output and thus a more consequent turnover and profitability, in addition to safety.

Because of the aforementioned stakes, many efforts have been put on the physical aspect of two-phase flows. This means that research laboratories and companies lead many physical experiments to gather knowledge. Examples of those are MISTRA at CEA in Saclay, France [43, 134], and PKL at AREVA in Erlangen, Germany [11]. The experiments permit to get invaluable data and know-how for the dimensioning work for future designs. In addition to experiments, numerical modelling also helps for knowing precisely when and where phenomenons appear in the different parts of a power generation unit. It is crucial for the design of new components [144], for the day-to-day operation of reactors, as well as for the comprehension of past operational regimes, incidents and accidents [92]. For instance, instead of some ten physical very expensive and difficult to manufacture physical experiments, we can try to replace them with hundreds of numerical experiments and just a few physical ones for calibration and validation.

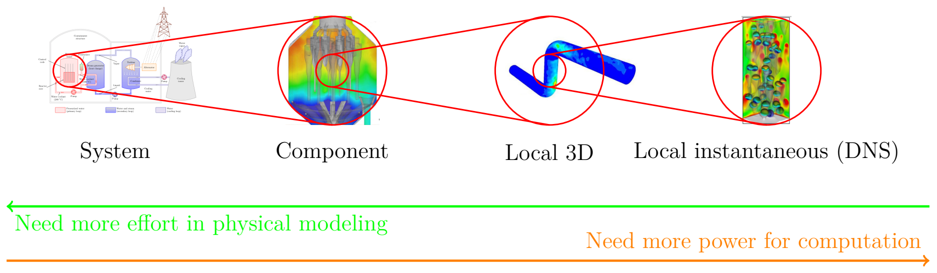

System
Component
Local 3D
Local instantaneous (DNS)
Need more effort in physical modeling
Need more power for computation

Figure 1.3 – Successive scales for modelling [83], [62], [30]

1.2 Different numerical simulation scales to model nuclear Thermal-Hydraulics

The numerical simulation of Thermal-Hydraulics is of primary importance for Nuclear Engineering for both security and efficiency of nuclear facilities at all scales [135]. The thermal-hydraulicists who try to represent flows in a nuclear context work on different modelling scales, represented on Figure 1.3. We can order them from largest to finest [83]:

1. system scale
2. component scale
3. local 3D scale
4. local instantaneous scale (DNS)

The system scale aims at representing the whole energy production unit, including in particular the reactor, the steam generators and all other pipes. It is the macro-scale used in well-known nuclear codes, like CATHARE [28], RELAP [25] or ATHLET [34]. This scale requires a large effort in physical modelling and that is why Thermal-Hydraulics equations are often used in their 0D expression.

The component scale aims at representing only a part of the plant: examples of codes for this scale include GENEPI [109], FLICA [137].

The local 3D scale is much finer and is related to Computational Fluid Dynamics. It is appropriate for the representation of problems with a size ranging from a few meters to a few tens of centimetres. For instance, Bieder et al. used Trio_U for the simulation of flows around nuclear fuel bundles [29].

The final scale is Direct Numerical Simulation. It is appropriate for the representation of problems with a size ranging from a few tens of centimetres to a few millimetres. At this scale, we can represent the interface between liquid and vapour; it is very applicable to bubble problems [40].

Given the sizes of a nuclear core, most if not all nuclear codes represent two-phase flows in an averaged manner. This means that the liquid-vapour interface is not explicitly represented in the models and the codes. Such averaged models thus require closure laws modelling mass, momentum and heat transfer between phases. In order to get such laws, physical experiments are of course necessary. Nonetheless, thanks to the improvement of computation capabilities, we can now hope to take advantage of fine numerical simulation like DNS because they explicitly represent the deformations of the liquid-vapour interfaces.

1.3 Outline and summary of the thesis

Note: Appendix C contains the translation into French of this section about the outline and the summary of the thesis.

1.3.1 Models of flows with two separated phases

In this thesis we start in Chapter 2 with explaining a series of existing thermal-hydraulics models, with a focus on two separated phases. We start with the common compressible models: Euler and Navier-Stokes systems. We explain the first one with only one phase in Section 2.1.1, the second one with two phases in Section 2.1.2. In Section 2.1.3, we then make the hypothesis of incompressibility, to get the incompressible Navier-Stokes model:

$$\begin{cases} \partial_t \mathbf{u} + \mathbf{u} \cdot \nabla \mathbf{u} - \nu \nabla^2 \mathbf{u} = \mathbf{f} = -\dfrac{1}{\rho} \nabla p + \mathbf{g} + \mathbf{f}_{others}, \\ \nabla \cdot \mathbf{u} = 0, \\ \mathbf{u} = \mathbf{u}_{BC} \text{ on } \partial\Omega \text{ (boundary conditions)}, \\ \mathbf{u}(t=0) = \mathbf{u}_0 \text{ on } \Omega \text{ (initial conditions)}. \end{cases} \tag{1.1}$$

Equation (1.1) gives the expression for one phase, with notations thoroughly defined further in the thesis. The expression for two phases is given by

Equation (1.2):

$$
\begin{cases}
\partial_t \mathbf{u} + \mathbf{u} \cdot \nabla \mathbf{u} - \nu \nabla^2 \mathbf{u} = \mathbf{f} = -\frac{1}{\rho} \nabla p + \mathbf{g} + \mathbf{f}_{others}, \\
\nabla \cdot \mathbf{u} = 0, \\
\partial_t Y + \mathbf{u} \cdot \nabla Y = 0, \\
\rho = \rho_g Y + \rho_l (1 - Y), \\
\mathbf{u} = \mathbf{u}_{BC} \text{ on } \partial\Omega \text{ (boundary conditions)}, \\
\mathbf{u}(t = 0) = \mathbf{u}_0 \text{ on } \Omega \text{ (initial conditions)}, \\
Y(t = 0) = Y_0 \text{ on } \Omega \text{ (initial conditions)}.
\end{cases}
\tag{1.2}
$$

We go on in Section 2.1.4 with the models used in nuclear codes, the 6- and 7-equation models, also referred to as mixture models. In Section 2.1.5, we focus on low-Mach number conditions; when the flows go at low speed compared to the speed of sound. We review the Diphasic Low Mach Number model, proposed for nuclear conditions too. Finally we present in Section 2.1.6 the Abstract Bubble Vibration model, a coupling between an hyperbolic equation and an elliptic equation.

When interested in the DNS of bubbles, we have to get the right tools to numerically model the interface. We present in Section 2.2 two important techniques. Front tracking, as explained in Section 2.2.1, is modelling the flow with a convective (or "Lagrangian") perspective. On the opposite, as explained in Section 2.2.2, we relate front capturing to a Eulerian perspective. We explain why we choose the latter. Our implementation is the transport of a colour function Y which equals either 1 in the gas phase, or 0 in the liquid phase. We will have to find the right discretisation scheme to keep the jump from 1 to 0 as sharp as possible.

1.3.2 Front capturing with AMR: finite differences schemes and parallelisation

We show in Chapter 3 that Direct Numerical Simulation is the most precise scale of computation and therefore also the most expensive one. Adaptive Mesh Refinement is the enrichment of a subset of the computational domain

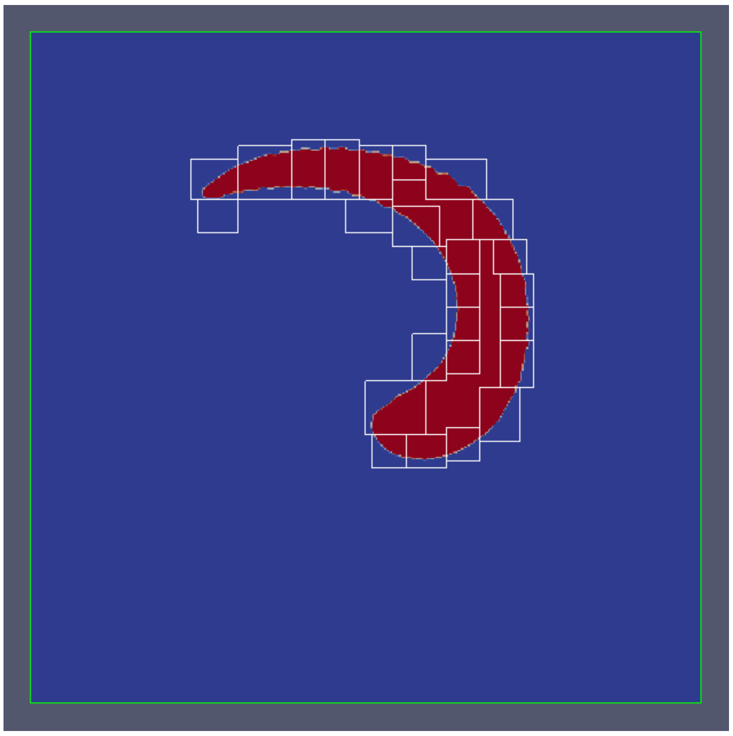


Figure 1.4 – 2D Kothe-Rider test, with AMR patches visible

– the area of interest – with more detail. We explain how AMR is beneficial to DNS by reviewing a part of the mesh refinement literature. In Section 3.1, we differentiate the following techniques: anisotropic meshing, r-adaptation, p-adaptation, h-adaptation, s-adaptation. We decide to focus on patch-based mesh adaptation on Cartesian grids in Section 3.2. This means that we cover the said area of interest with one or several levels of patches, which have a finer space discretisation. Figure 1.4 shows a standard test case for advection equations, the 2D Kothe-Rider test [119]. We used patch-based AMR with one level of refinement. We show the coarse level (in fact, the entire computational domain) inside a green square, it is discretised with a 100×100 grid. We show the fine level inside the many white rectangles, which are patches with a refinement coefficient $r = 4$. They cover the area we determined of interest: the interface between liquid and gas. We can say that the simulation has an equivalent discretisation of 400×400.

We present multiple ways to define the patches; the Berger-Rigoutsos algorithm [22], the Livne algorithm [99]. We also propose our own improvement, the $n_{min} - n_{max}$ algorithm. It constrains the size of the patches with a minimum length n_{min} and a maximum length n_{max}; a third parameter being the minimum efficiency η_{min} of the covering. We introduce three quality functions to compare patch coverings: the average efficiency η, the normalised standard deviation of sizes σ and the average squareness γ. We compare the three algorithms in a multiprocessing perspective in Section 3.3. For this we determine the value of the quality functions on a few hundreds of test coverings using the three algorithms. We settle for $n_{min} - n_{max}$. We decide to test out our choice in Section 3.4 with a 3D Kothe-Rider advection simulation. We present different advection schemes, including the upwind scheme and WENO. We choose the limited downwind scheme introduced by Després and Lagoutière [51], because it captures and transports the colour function Y in a very sharp manner. By locating the refinement patches on the interface between liquid and gas, we get encouraging results as far as computational speed-up is concerned when we use parallel computing.

1.3.3 Elliptic equations and Abstract Bubble Vibration model

As we explain in Chapter 4, elliptic equations are more challenging to represent with patch-based AMR, since they require information from the whole computational domain. This is why, in Section 4.1, we choose to use the Local Defect Correction algorithm [74]. As schematically presented on Figure 1.5, LDC is an iterative process done until we determine that we reached convergence. At each iteration, the computed solutions of the coarse and the fine (patch) grids enrich one another. The computed solution on the coarse grid defines Dirichlet boundary conditions on the borders of the patches. We then solve the elliptic equation with the said boundary conditions on the fine grid. The computed solution of the fine grid permits to calculate the eponymous Local Defect Correction on the refined area: it will replace the source term of the coarse level, eventually leading to a new coarse resolution and a new LDC iteration.

In the following section, we recall the proof of its convergence laid out by Ferket et al. [63] and Anthonissen et al. [10]. We then propose our variant of LDC, differing from the literature in two ways. First, we use cell-centred values and not values on nodes of the meshes. Therefore, for the Dirichlet boundary conditions interpolation step from coarse to fine, we provide the patches with ghost cells around their border. Second, most of the time, we deal with multiple patches, often touching each other. So we decide to consider them as a partition of the fine level: we use the Schwarz iterative algorithm of domain decomposition to determine an acceptable solution of

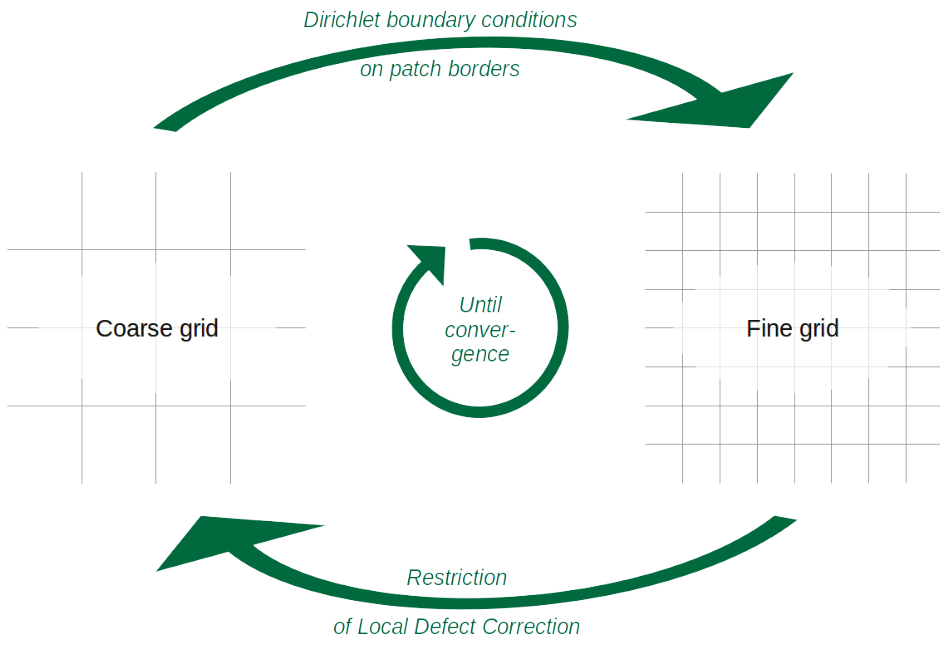


Figure 1.5 – Schematic representation of the LDC algorithm

the fine level, with no discontinuity at the interface between patches. We test our implementation out with the ABV model in Section 4.2, in 2D as well as in 3D. We locate the patches on the interface of the bubble. We use a formula giving the volume of the bubble as a function of time to verify that we get convincing results. We implemented the AMR tools we used in an open-source library named CDMATH and conceived to help other computational scientists and engineers [44, 145]. We present how CDMATH was designed and how AMR is an extension in Section 4.3. With this toolbox, one can implement AMR as easily in 2D as in 3D.

1.3.4 Application to incompressible Navier-Stokes

Finally, we apply in Chapter 5 the result of our work on Adaptive Mesh Refinement to more realistic simulations. We represent incompressible Navier-Stokes systems on staggered grids (scalar variables located at the centre of cells, vectors located at the faces). We use one level of AMR. In Section 5.1, we fully detail the numerical schemes for a one-phase simulation of Equation (1.1). In the thesis, we detail the following prediction-correction scheme, in the spirit of the work of Chorin [38, 39] and Temam [131, 132]:

$$\begin{cases} \dfrac{\tilde{\mathbf{u}} - \mathbf{u}^n}{\Delta t} + \mathbf{u}^n \cdot \nabla \tilde{\mathbf{u}} - \nu \nabla^2 \tilde{\mathbf{u}} = \mathbf{f}^n = \mathbf{g} - \dfrac{1}{\rho} \nabla p^n, \\ \tilde{\mathbf{u}} = \tilde{\mathbf{u}}_{BC} \text{ on } \partial\Omega, \\ \tilde{\mathbf{u}}_{BC} \cdot \mathbf{n} = \mathbf{u}_{BC} \cdot \mathbf{n}, \\ \tilde{\mathbf{u}}_{BC} \cdot \mathbf{t} = \mathbf{u}_{BC} \cdot \mathbf{t} + \dfrac{\Delta t}{\rho} \nabla \phi^n \cdot \mathbf{t}. \end{cases} \tag{1.3}$$

$$\begin{cases} -\dfrac{1}{\rho} \Delta \phi^{n+1} = -\dfrac{1}{\Delta t} \nabla \cdot \tilde{\mathbf{u}}, \\ \nabla \phi^{n+1} \cdot \mathbf{n} = 0 \text{ on } \partial\Omega. \end{cases} \tag{1.4}$$

$$p^{n+1} = p^n + \phi^{n+1}. \tag{1.5}$$

$$\begin{cases} \dfrac{\mathbf{u}^{n+1} - \tilde{\mathbf{u}}}{\Delta t} = -\dfrac{1}{\rho} \nabla \phi^{n+1}, \\ \mathbf{u}^{n+1} = \mathbf{u}_{BC} \text{ on } \partial\Omega. \end{cases} \tag{1.6}$$

As explained later, we use the LDC algorithm to finely compute the increment of pressure ϕ. We see that it is essential the source term of Equation (1.4) – namely the divergence of the predicted velocity $\nabla \cdot \tilde{\mathbf{u}}$ – be linearly interpolated from the coarse level onto the fine level. We verify our implementation using the classic literature test case of the lid-driven cavity. A permanent regime exists and is well known beforehand: depending on the Reynolds number, several whirlpools are created in the computational domain, so we placed one refinement patch on the main whirlpool. Figure 1.6 represents the permanent regime, with the colouring as a function of the velocity $||\mathbf{u}||$ and the isocontours of $||\mathbf{u}||$ given as white lines. We represent the location of the fine level patch by a white square. A video of the transitional regime can be seen online at `https://youtu.be/esOHN--iW4Y`.

In Section 5.2, we switch to two-phase situations, represented by Equation (1.2). We also use staggered grids, with one exception though: we locate the inverse $\left(\frac{1}{\rho}\right)$ of the volumetric mass at the faces and not at the centre of cells, although it is a scalar. Here too we use a prediction-correction scheme which

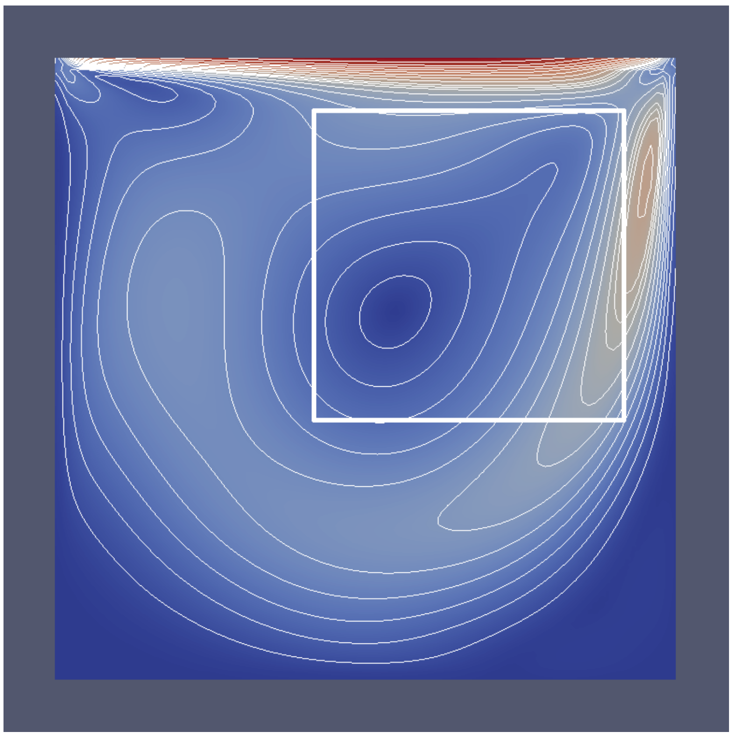


Figure 1.6 – Lid-driven cavity simulation (isocontours of $||\mathbf{u}||$)

equations are fully explained in the thesis:

$$\begin{cases} \dfrac{\tilde{\mathbf{u}} - \mathbf{u}^n}{\Delta t} + \mathbf{u}^n \cdot \nabla \tilde{\mathbf{u}} - \nu \nabla^2 \tilde{\mathbf{u}} = \mathbf{g}^{n+1} - \dfrac{1}{\rho^n} \nabla p, \\ \\ \tilde{\mathbf{u}} = \mathbf{u}_{BC} \text{ on } \partial\Omega. \end{cases} \tag{1.7}$$

$$\begin{cases} -\nabla \cdot \left(\dfrac{1}{\rho^n} \nabla \phi^{n+1} \right) = -\dfrac{1}{\Delta t} \nabla \cdot \tilde{\mathbf{u}}, \\ \\ \nabla \phi^{n+1} \cdot \mathbf{n} = 0 \text{ on } \partial\Omega. \end{cases} \tag{1.8}$$

$$p^{n+1} = p^n + \phi^{n+1}. \tag{1.9}$$

$$\begin{cases} \dfrac{\mathbf{u}^{n+1} - \tilde{\mathbf{u}}}{\Delta t} = -\dfrac{1}{\rho^n} \nabla \phi^{n+1}, \\ \\ \mathbf{u}^{n+1} = \mathbf{u}_{BC} \text{ on } \partial\Omega. \end{cases} \tag{1.10}$$

$$\frac{Y^{n+1} - Y^n}{\Delta t} + \mathbf{u}^{n+1} \cdot \nabla Y^n = 0. \tag{1.11}$$

$$\rho^{n+1} = \rho_g Y^{n+1} + \rho_l (1 - Y^{n+1}). \tag{1.12}$$

As for the one-phase simulation, we use one level of Adaptive Mesh Refinement. We create the patches dynamically at every time step and locate them such that they follow the interface between gas and liquid, as in Chapters 3 and 4. We propose an original AMR approach that benefits from the work done earlier. We first decide to compute the increment of pressure ϕ with the LDC algorithm applied on Equation (1.8), which is elliptic. Then we decide to compute the hyperbolic advection of Equation (1.11) with AMR precision too. We explain why it is essential that we interpolate the source term $\nabla \cdot \tilde{\mathbf{u}}$ of Equation (1.8) from the coarse level to the fine level. As a consequence, we have a precise computation of the variables ϕ, p, Y and ρ at the location of their discontinuities or jumps of derivative. In addition, in spite of the sharp jump of the presence Y, we are able to give a satisfying model for surface tension of bubbles. This lets us obtain realistic evolutions of non-stationary bubbles due to gravity, viscosity, inertia, surface tension, pressure forces. Figure 1.7 is a shot of such a simulation, which video can be seen online at `https://youtu.be/zJEjP6JYEYQ`. We can see the AMR patches as white rectangles on the interfaces and the streamlines of the velocity as white little arrows.

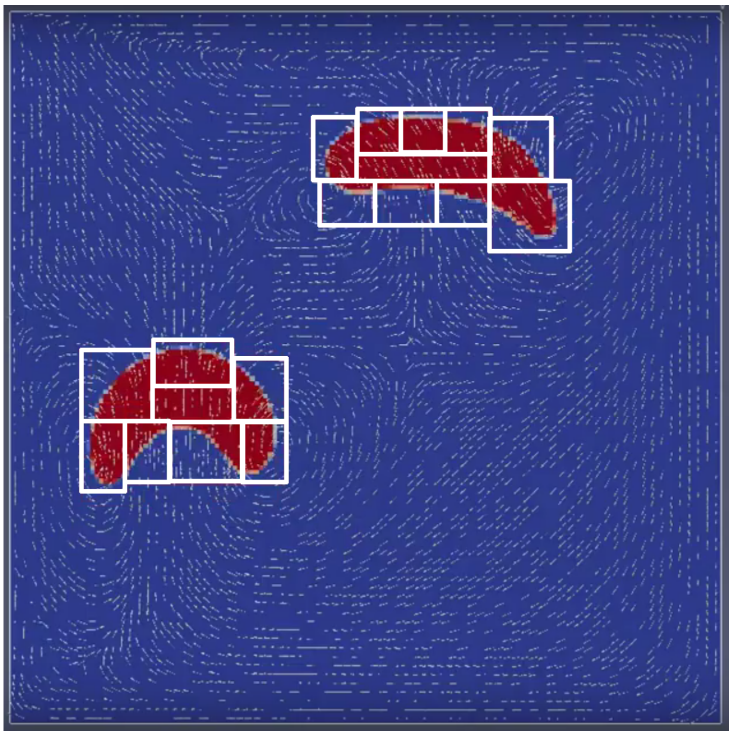


Figure 1.7 – Simulation of two bubbles, with AMR patches visible as white rectangles

1.3.5 Outreach

The last chapter, Chapter 6 is just the listing of our communications realised during the PhD, as well as of supervising and funding institutions.

Chapter 2

Models of flows with two separated phases

2.1 Thermal-Hydraulics models adapted for each scale

In this section we present a few physics models used to represent one-phase and two-phase flows.

For all models, we will use the same denomination of spatial areas of the problem, represented on Figure 2.1. We will also use the same initial conditions.

We name the computational domain Ω and we name its border $\partial\Omega$. The volume of Ω is written $|\Omega|$. We partition the space into two: $\Omega = \Omega_l \cup \Omega_g$. The space Ω_l represents the domain of Fluid l, typically a liquid. The space Ω_g represents the domain of Fluid g, typically a gas.

Let $\mathbf{x}$ be the vector positioning in space and let t be the time. We define the colour function $Y(\mathbf{x}, t)$ as an indicator for the gas phase:

$$\forall(\mathbf{x}, t) \in \Omega \times \mathbb{R}_+, Y(\mathbf{x}, t) = \begin{cases} 1 & \text{if} \quad \mathbf{x} \in \Omega_g \qquad \text{(i.e. gas/vapor)}, \\ 0 & \text{if} \quad \mathbf{x} \in \Omega_l \qquad \text{(i.e. liquid)}. \end{cases} \tag{2.1}$$

So we can express the initial conditions for Y as follows:

$$\forall \mathbf{x} \in \Omega, Y(\mathbf{x}, t = 0) = \begin{cases} 1 & \text{if} \quad \mathbf{x} \in \Omega_g(t = 0), \\ 0 & \text{if} \quad \mathbf{x} \in \Omega_l(t = 0). \end{cases} \tag{2.2}$$

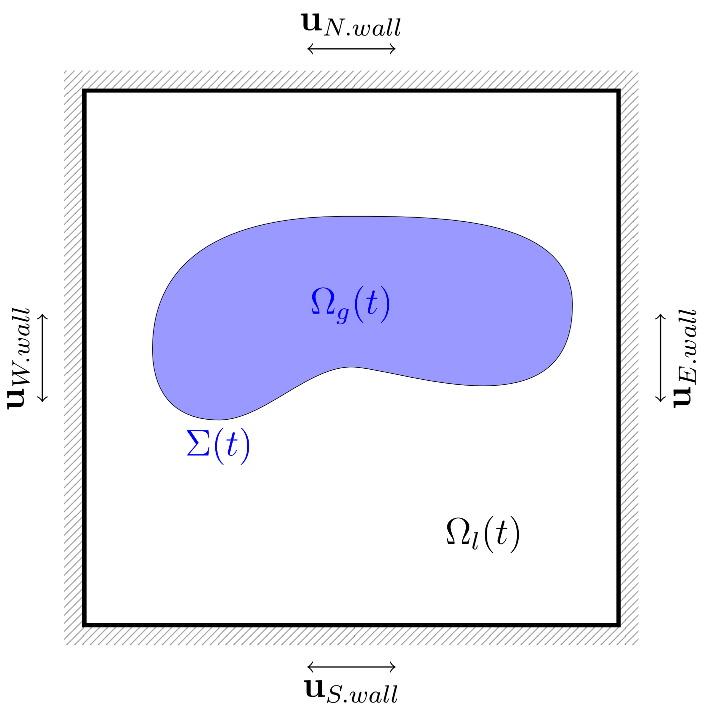


Figure 2.1 – $\Omega(t) = \Omega_l(t) \cup \Omega_g(t)$

We also define $\Sigma(t)$ the interface separating liquid and gas. It may evolve with time, appear in some places or disappear in others.

Let the scalar $T(\mathbf{x}, t)$ be the temperature and $p(\mathbf{x}, t)$ the local pressure of the fluid. We call "normal temperature and pressure conditions" the couple (T, p) where $T = 25\,°\mathrm{C}$ and $p = 1.024\,\mathrm{bar}$. Let the scalar $\rho(\mathbf{x}, t)$ be the volumetric mass. For liquid water at normal temperature and pressure conditions, $\rho_{water} = 1000\,\mathrm{kg\,m^{-3}}$. Let the vector $\mathbf{u}(\mathbf{x}, t)$ be the local velocity of the fluid. Then $\rho\mathbf{u}$ represents the local volumetric momentum.

The vector $\mathbf{F}$ represents physical volumetric forces. In our case, the forces include at least pressure forces, gravity and eventually other sources $\mathbf{F}_{others}$. Let $\mathbf{f}$ be the expression of these forces divided by the volumetric mass ρ: $\mathbf{f}$ is homogeneous to an acceleration. As an example, the vector $\mathbf{g}$ is the gravitational acceleration, with a norm $||\mathbf{g}|| = 9.8\,\mathrm{m\,s^{-2}}$. Let τ be the stress tensor, with Stokes' hypothesis. The scalar μ is the volumetric viscosity and the scalar ν is the kinematic (i.e. specific or massic) viscosity coefficient; $\nu = \frac{\mu}{\rho}$. For instance, for liquid water at $25\,°\mathrm{C}$, $\nu_{water} = 10^{-6}\mathrm{m^2\,s^{-1}}$. Table 2.1 gives other numerical values for the kinematic viscosity of liquid water.

Temperature (°C)	Kinematic viscosity ($\mathrm{m}^2\,\mathrm{s}^{-1}$)
10	1.308×10^{-6}
20	1.002×10^{-6}
30	0.7978×10^{-6}
40	0.6531×10^{-6}
50	0.5471×10^{-6}
60	0.4658×10^{-6}
70	0.4044×10^{-6}
80	0.3550×10^{-6}
90	0.3150×10^{-6}
100	0.2822×10^{-6}

Table 2.1 – Kinematic viscosity of liquid water for different temperatures

Let E be the specific energy, ρE the volumetric energy and e be the specific internal energy. Let λ be the thermal transfer coefficient.

2.1.1 One-phase compressible Euler system

For two given vectors $\mathbf{u}$ and $\mathbf{v}$, let us define $\mathbf{u} \cdot \nabla \mathbf{v}$ as follows:

$$\begin{aligned} \mathbf{u} \cdot \nabla \mathbf{v} &= (\mathbf{u} \cdot \nabla)(\mathbf{v}), \\ &= (u_x \partial_x + u_y \partial_y + u_z \partial_z)(\mathbf{v}). \end{aligned} \tag{2.3}$$

The one-phase compressible Euler system is given by the following set of equations (written using a non-conservative formulation):

$$\begin{cases} \partial_t \rho + \nabla \cdot (\rho \mathbf{u}) = 0, \\ \\ \partial_t \mathbf{u} + \mathbf{u} \cdot \nabla \mathbf{u} = -\dfrac{1}{\rho} \nabla p + \mathbf{g}, \\ \\ \partial_t e + \mathbf{u} \cdot \nabla e + \dfrac{p}{\rho} \nabla \cdot \mathbf{u} = 0. \end{cases} \tag{2.4}$$

The first equation is the conservation of mass, the second one is the transport of momentum, and the last one is the transport of internal energy. To close the system of Equations (2.4), we add the equation of state as well: this is an algebraic equation that establishes the relationship between the pressure p, the internal energy e and the volumetric mass ρ. Using a determined function $\mathcal{EOS}$, the equation of state can take the general formulation:

$$\mathcal{EOS}(p, e, \rho) = 0. \tag{2.5}$$

For instance, let us consider that the fluid is a Laplace ideal gas. Let us call $\underline{\gamma}$ the ideal gas factor, equal to the heat capacity at constant volume divided by the heat capacity at constant pressure. The quantity $\underline{\gamma}$ is equal to $\frac{5}{3}$ if the Laplace ideal gas is a monoatomic gas (like helium He) and $\underline{\gamma} = \frac{7}{5}$ if the gas is diatomic (like nitrogen N_2). In this case we can use the ideal gas law as the equation of state:

$$\mathcal{EOS}(p, e, \rho) = p - \rho e(\underline{\gamma} - 1) = 0. \tag{2.6}$$

As far as the boundary conditions are concerned, let Ω be bounded by immobile walls. The boundary conditions for the Euler system are a little complicated to write down. In particular, they depend on where and when the velocity is directed into the domain.

2.1.2 Two-phase compressible Navier-Stokes model

We can represent the two-phase compressible Navier-Stokes equations with the following set of four equations:

$$\begin{cases} \partial_t Y + \mathbf{u} \cdot \nabla Y = 0, \\ \rho = \rho_g Y + \rho_l (1 - Y), \\ \partial_t \rho + \nabla \cdot (\rho \mathbf{u}) = 0, \\ \partial_t (\rho \mathbf{u}) + \nabla \cdot (\rho \mathbf{u}) = -\nabla p + \nabla \cdot \tau(\mathbf{u}) + \rho \mathbf{g}, \\ \partial_t (\rho E) + \nabla \cdot [(\rho E + p)\mathbf{u}] \\ \qquad = \nabla \cdot (\lambda \nabla T) + \nabla \cdot [\tau(\mathbf{u})\mathbf{u}] + \rho \mathbf{g} \cdot \mathbf{u}. \end{cases} \tag{2.7}$$

The first equation is the transport of the colour function Y. It is expressed in a non-conservative way which satisfies a maximum principle for Y. The second equation gives the volumetric mass ρ as a function of the presence of the two phases. The third equation is the conservation of said ρ. The fourth equation is the conservation of the volumetric momentum $\rho\mathbf{u}$. The fifth and last equation is the conservation of volumetric energy ρE. In addition, we also have the equation of state for the relationship between Y, p, e given by

$e = E - \frac{1}{2}u^2$ and ρ. Let $\mathcal{EOS}_k(p, e, \rho_k) = 0$ be the equation of state of each phase $k = l$ and $k = g$. Multiple choices are possible for the equation of state of the mixture [46, 62]. We may for instance choose the following equation defining $\mathcal{EOS}(Y, p, e, \rho)$:

$$\mathcal{EOS}(Y, p, e, \rho) = (1 - Y)\mathcal{EOS}_l(p, e, \rho_l) + Y\mathcal{EOS}_g(p, e, \rho_g) = 0. \tag{2.8}$$

We choose the value of all the coefficients depending on the phase, such as transport and thermodynamic coefficients, as a function of Y. Let ξ be a coefficient as for instance λ, α, C_p and others:

$$\xi(Y, T, p) = (1 - Y)\xi_l(T, p) + Y\xi_g(T, p). \tag{2.9}$$

For instance, we have the following equality for λ:

$$\lambda(Y, T, p) = \begin{cases} \lambda_l(T, p) \text{ if } \mathbf{x} \in \Omega_l(t) \text{ (i.e. } Y = 0), \\ \lambda_g(T, p) \text{ if } \mathbf{x} \in \Omega_g(t) \text{ (i.e. } Y = 1). \end{cases} \tag{2.10}$$

As far as the boundary conditions are concerned, Ω is bounded by immobile walls. There are many types of possible boundary conditions, as for instance Neumann or Robin ones. We decide to focus on Dirichlet boundary conditions. Let $\mathbf{u}_{BC}$ be a boundary conditions given for the velocity $\mathbf{u}$:

$$\forall \mathbf{x} \in \partial\Omega, \mathbf{u}(\mathbf{x}, t) = \mathbf{u}_{BC}. \tag{2.11}$$

We choose to have walls that can only move in the direction they occupy, as shown on Figure 2.1. In other words, a wall along the x axis can only move in the x direction. Furthermore, we impose no-slip conditions, so $\mathbf{u}_{BC} = \mathbf{u}_{wall}$. However, if the speed on the walls is not zero, then we have to choose entry conditions on the boundaries for the advection contributions.

We also choose to have adiabatic conditions:

$$\forall \mathbf{x} \in \partial\Omega, \nabla T(\mathbf{x}, t) \cdot \mathbf{n}(\mathbf{x}) = 0 \tag{2.12}$$

where $\mathbf{n}$ is the vector normal to the wall $\partial\Omega$.

The compressible Navier-Stokes system is adapted for problems where acoustic shock waves appear and should be represented correctly. Kokh presents this particular two-phase compressible Navier-Stokes system – but also other ways to model the interface – in his PhD thesis [89]. As a final note, the Euler system (2.4) can be seen as a Navier-Stokes system where we make the viscosity ν equal to zero.

2.1.3 Incompressible Navier-Stokes model

The incompressible Navier-Stokes model accounts for the conservation of (volumetric) momentum $\rho\mathbf{u}$ in fluids which density (or volumetric mass) ρ is constant in time and uniform by pieces in space. In particular, ρ only depends on the species of the fluid. Thus it cannot model large thermal dilation nor pressure shock waves.

The one-phase incompressible Navier-Stokes model is hence given by the following set of equations:

$$\begin{cases} \partial_t \mathbf{u} + \mathbf{u} \cdot \nabla \mathbf{u} - \nu \nabla^2 \mathbf{u} = \mathbf{f} = -\dfrac{1}{\rho} \nabla p + \mathbf{g} + \mathbf{f}_{others}, \\ \nabla \cdot \mathbf{u} = 0, \\ \mathbf{u} = \mathbf{u}_{BC} \text{ on } \partial\Omega \text{ (boundary conditions)}, \\ \mathbf{u}(t=0) = \mathbf{u}_0 \text{ on } \Omega \text{ (initial conditions)}. \end{cases} \tag{2.13}$$

The vector $\mathbf{u}_0$ is an initial condition given for the velocity $\mathbf{u}$ and verifying $\nabla \cdot \mathbf{u}_0 = 0$.

We numerically solve an example of this one-phase model in Section 5.1.

For the two-fluid incompressible Navier-Stokes model, we consider two fluids: Fluid l and Fluid g. As a consequence, ρ is constant in time and uniform by pieces in space. The density is attached to the colour function Y. The scalar ρ equals ρ_l in areas where $Y = 0$ and equals ρ_g in the other areas, where $Y = 1$. The colour function Y thus indicates the presence of Fluid g. The scalar Y is transported by the velocity $\mathbf{u}$ (defined on all the computational domain), which brings us to the following set of equations:

$$\begin{cases} \partial_t \mathbf{u} + \mathbf{u} \cdot \nabla \mathbf{u} - \nu \nabla^2 \mathbf{u} = \mathbf{f} = -\frac{1}{\rho} \nabla p + \mathbf{g} + \mathbf{f}_{others}, \\ \nabla \cdot \mathbf{u} = 0, \\ \partial_t Y + \mathbf{u} \cdot \nabla Y = 0, \\ \rho = \rho_g Y + \rho_l (1 - Y), \\ \mathbf{u} = \mathbf{u}_{BC} \text{ on } \partial\Omega \text{ (boundary conditions)}, \\ \mathbf{u}(t=0) = \mathbf{u}_0 \text{ on } \Omega \text{ (initial conditions)}, \\ Y(t=0) = Y_0 \text{ on } \Omega \text{ (initial conditions)}. \end{cases} \tag{2.14}$$

We numerically solve an example of this two-fluid model in Section 5.2.

2.1.4 Mixture models: the 6- and the 7-equation models

An abundant category of two-phase flow models is mixture models. Those consider that for every point in space (and every discretisation volume surrounding it), one can define a void fraction α. For an elementary space of a volume V and containing a volume V_g of gas, the following formula gives the void fraction:

$$\alpha = \frac{V_g}{V}. \tag{2.15}$$

Mixture models are thus "homogenised" or "averaged" models where for each point $\mathbf{x}$ in space, both phases coexist. We understand that mixture models are adapted to larger scale problems.

The 6-equation model, also sometimes named the two-fluid model, is used for instance in the nuclear code CATHARE. It is based on a set of 6 equations of conservation [28].

The conservation of mass of each phase reads as follows:

$$\begin{aligned} \frac{\partial}{\partial t}(\alpha_g \rho_g) + \nabla \cdot (\alpha_g \rho_g \mathbf{u}_g) = 0, \\ \frac{\partial}{\partial t}(\alpha_l \rho_l) + \nabla \cdot (\alpha_l \rho_l \mathbf{u}_l) = 0. \end{aligned} \tag{2.16}$$

The conservation of momentum of each phase reads as follows:

$$\begin{aligned}\frac{\partial}{\partial t}(\alpha_g\rho_g\mathbf{u}_g) + \nabla\cdot(\alpha_g\rho_g\mathbf{u}_g\otimes\mathbf{u}_g) + \alpha_g\nabla p = \mathbf{F}_g,\\ \frac{\partial}{\partial t}(\alpha_l\rho_l\mathbf{u}_l) + \nabla\cdot(\alpha_l\rho_l\mathbf{u}_l\otimes\mathbf{u}_l) + \alpha_l\nabla p = \mathbf{F}_l.\end{aligned} \tag{2.17}$$

Finally, the conservation of momentum of each phase reads as follows:

$$\begin{aligned}&\tfrac{\partial}{\partial t}(\alpha_g E_g) + \nabla\cdot(\alpha_g E_g\mathbf{u}_g) + \alpha_g\nabla\cdot(p\mathbf{u}_g)\\ &\quad= \{\text{Net energy exchange with the environment}\}_g\\ &\qquad+\{\text{Net energy generation}\}_g,\\ \\ &\tfrac{\partial}{\partial t}(\alpha_l E_l) + \nabla\cdot(\alpha_l E_l\mathbf{u}_l) + \alpha_l\nabla\cdot(p\mathbf{u}_l)\\ &\quad= \{\text{Net energy exchange with the environment}\}_l\\ &\qquad+\{\text{Net energy generation}\}_l.\end{aligned} \tag{2.18}$$

The system is completed with closure laws. One of the drawbacks of this model is that it may show non-hyperbolicity problems (see [128] for instance). Without going into too many details, let us consider the 7-equation model which unconditionally solves the non-hyperbolicity problems. This model is called Baer-Nunziato, introduced in [14]. The model is further studied in [67, 45] and schemes specific to the model are presented in [6, 41]. It introduces an interface pressure p_Σ and an interface velocity $\mathbf{u}_\Sigma$. Let k refer to the fluid g or l. Then the conservation laws rewrite as follows:

$$\frac{\partial\alpha_k}{\partial t} + \mathbf{u}_\Sigma\cdot\nabla\alpha_k = \mu(p_k - p_\Sigma), \tag{2.19}$$

$$\frac{\partial\left(\alpha_k\rho_k\right)}{\partial t} + \nabla\cdot(\alpha_k\rho_k\mathbf{u}_k) = 0, \tag{2.20}$$

$$\frac{\partial\left(\alpha_k\rho_k\mathbf{u}_k\right)}{\partial t} + \nabla\cdot(\alpha_k\rho_k\mathbf{u}_k\otimes\mathbf{u}_k) + \nabla(\alpha_k p_k) = p_\Sigma\nabla\alpha_k + \lambda(\mathbf{u}_k - \mathbf{u}_\Sigma), \tag{2.21}$$

$$\begin{aligned}&\frac{\partial\left(\alpha_k\rho_k E_k\right)}{\partial t} + \nabla\cdot\Big(\ (\alpha_k\rho_k E_k + \alpha_k p_k)\mathbf{u}_k\ \Big)\\ &\quad= p_\Sigma\mathbf{u}_k\cdot\nabla\alpha_k + \lambda\mathbf{u}_\Sigma(\mathbf{u}_k - \mathbf{u}_\Sigma) + \mu p_\Sigma(p_k - p_\Sigma).\end{aligned} \tag{2.22}$$

We can notice that Equation (2.19) is a transport of the void fraction plus a relaxation term.

In practical simulations, the relaxation parameters λ and μ are very small (10^{-8}), the interface pressure p_Σ is taken as the largest pressure of the fluids and the interface velocity $\mathbf{u}_\Sigma$ is taken equal to the velocity of the other fluid (i.e. with the lower pressure).

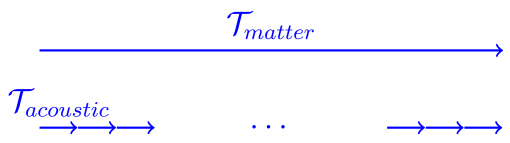


Figure 2.2 – Characteristic durations

2.1.5 The Diphasic Low-Mach Number model

Low-Mach flow conditions

In most situations, there are two very distinct time scales in thermal-hydraulic flows. The first one is the time scale of pressure shocks and other acoustic phenomenons. Typically, the order of magnitude of the phenomenons nears $\mathcal{T}_{acoustic} \simeq 10^{-3}$s. The second time scale is related to how fast the matter – the fluid in itself – flows along pipes etc. It is typically of the order of magnitude of $\mathcal{T}_{matter} \simeq 1\,$s in nuclear cores.

These two time scales are very different one from another, as represented on Figure 2.2. This brings specific difficulties:

1. the precision in schemes (e.g. the Godunov scheme has a poor precision in low-Mach number conditions on Cartesian grids),
2. the robustness of solvers (for instance, inverting matrices with very different eigenvalues is hard with usual iterative methods, because of the bad conditioning; this is indeed the case at low Mach number).

We define the Mach number as follows:

$$\mathcal{M} = \frac{||\mathbf{u}||}{c}. \tag{2.23}$$

The module $||\mathbf{u}||$ is the velocity of the fluid matter (or a magnitude thereof). The quantity c is the speed of sound. In a nuclear reactor in particular, we are in low-Mach number conditions in the vast majority of cases, be it in nominal regime or in many accidental regimes like *Loss Of Flow Accidents* for example: $\mathcal{M} = \frac{||\mathbf{u}||}{c} \simeq 10^{-3}$.

As a consequence, we can say we can neglect shock waves and other acoustic phenomenons.

However thermal phenomenons do remain important: due to thermal dilation, $\nabla \cdot \mathbf{u} \neq 0$ although $\mathcal{M} \ll 1$.

$\mathcal{M} \ll 1$		$\mathcal{M} \ll 1$		$\mathcal{M} = \mathcal{O}(1)$
$\lVert\nabla \cdot \lambda \nabla T\rVert \ll 1$		$\lVert\nabla \cdot \lambda \nabla T\rVert = \mathcal{O}(1)$		
Incomp. Navier-Stokes	$\leq$	Low-Mach asymptotic model	$<$	Comp. Navier-Stokes
×acoustic		×acoustic		✓acoustic
×thermal		✓thermal		✓thermal

Table 2.2 – Classification of approximations

Table 2.2 summarizes a classification of models. On the far right, we have the compressible Navier-Stokes model, which takes into account all phenomenons (acoustic and thermal). The Mach number $\mathcal{M}$ may be large. See Section 2.1.2. On the far left, we have the incompressible Navier-Stokes model, which neglects acoustic as well as thermal phenomenons. See Section 2.1.3. In between, we are searching for a model which neglects acoustic phenomenons but still represents thermal dilation. This would be the low-Mach asymptotic model presented hereafter.

Low-Mach number approximation: DLMN model

The Diphasic Low-Mach Number (DLMN) model [47, 46] takes its inspiration from previous work about combustion, dating back at least since the seventies; see for instance [100]. Low-Mach number models have also been used in many other fields, as for instance in Cosmology [5] or Microfluidics for electronic systems [118]. It was then proposed for Nuclear Reactors Thermal-Hydraulics [48, 76]. Penel studied the model from a theoretical and analytic point of view [113, 114].

We can show that, when the Mach number $\mathcal{M}$ is small, we can approximate the two-phase compressible Navier-Stokes model (as presented in Section 2.1.2) with the DLMN model. Let θ be the coefficient of thermal expansion:

$$\theta(Y, T, p) = -\frac{1}{\rho}\frac{\partial \rho}{\partial T}(Y, T, p). \tag{2.24}$$

In this model, we keep the transport equation for Y. We replace the energy equation with the transport equation for internal enthalpy. We have a conservation equation for momentum. At last, we have a coupling equation of constraints with all other equations. This leads us to the following system of equations:

$$
\begin{cases}
\partial_t Y + \mathbf{u} \cdot \nabla Y = 0, \\
\rho C_p(\partial_t T + \mathbf{u} \cdot \nabla T) = \theta T P'(t) + \nabla \cdot (\lambda \nabla T), \\
\rho(\partial_t \mathbf{u} + \mathbf{u} \cdot \nabla \mathbf{u}) = -\nabla \Pi + \nabla \cdot \tau(\mathbf{u}) + \rho \mathbf{g}, \\
\nabla \cdot \mathbf{u} = G(\mathbf{x}, t), \\
G(\mathbf{x}, t) = -\dfrac{P'(t)}{\Gamma P(t)} + \dfrac{\beta}{P(t)} \nabla \cdot (\lambda \nabla T), \\
\beta = \dfrac{\theta P}{\rho C_p}, \\
\Gamma = \dfrac{\rho c^2}{P}, \\
P'(t) = \dfrac{\displaystyle\int_\Omega \beta(Y, T, P) \nabla \cdot \lambda \nabla T dx}{\displaystyle\int_\Omega \dfrac{dx}{\Gamma(Y, T, P)}}.
\end{cases}
\tag{2.25}
$$

The quantity P is a pressure depending only on time and equations of state. In this model, P is the "thermodynamic pressure" and does not depend on the spatial coordinate $\mathbf{x}$. The coefficient of thermal dilation now depends on P: $\theta = \theta(Y, T, P)$. The quantity Π is a perturbation of the pressure, or a "dynamic pressure". It is not driven by thermodynamics but rather the velocity field, gravity, etc.

For the initial conditions, we keep the same as explained before in the introduction of Section 2.1, with in addition $\mathbf{u}^0(\mathbf{x})$ matching the right divergence condition:

$$\nabla \cdot \mathbf{u}^0 = G(\mathbf{x}, t = 0). \tag{2.26}$$

Boundary conditions (external on $\partial\Omega$ and internal on $\Sigma(t)$) are kept the same as for compressible Navier-Stokes. Similarly, transport and thermal coefficients are also kept identical.

2.1.6 Abstract Bubble Vibration model

The Abstract Bubble Vibration (ABV) model is a simplified model which can represent a dilating bubble. This model is based on a coupling between a hyperbolic part and an elliptic part and was introduced in [50] by Dellacherie and Lafitte. Let Ω be a closed computational domain with a boundary $\partial\Omega$. We consider three unknown functions: the colour function Y which represents the presence of void in liquid when it is equal to 1, $\mathbf{u}$ a vector field which represents the velocity of the fluid, and ϕ a potential from which $\mathbf{u}$ derives. We also consider $\psi(t)$ a given function which could represent a pulsation contribution. The ABV model is represented by the following set of equations:

$$\begin{cases} \partial_t Y + \mathbf{u}\cdot\nabla Y = 0, \\ \\ \Delta\phi = \psi(t)\left(Y - \dfrac{1}{|\Omega|}\displaystyle\int_\Omega Y dx\right), \\ \\ \nabla\phi\cdot\mathbf{n} = 0 \text{ on } \partial\Omega, \\ \\ \mathbf{u} = \nabla\phi. \end{cases} \tag{2.27}$$

The first equation is an advection equation of Y with velocity $\mathbf{u}$, as we had in the previous sections. The second one is an elliptic equation. As for the initial conditions, it is sufficient to only define $Y(t=0)$ on the computational domain Ω. We impose $\forall \mathbf{x} \in \Omega, Y(\mathbf{x}, t=0) \in \{0,1\}$.

As Y is either equal to 0 (outside the bubble) or 1 (inside the bubble), we can interpret $\int_\Omega Y dx$ as the volume V of the bubble and $\frac{1}{|\Omega|}\int_\Omega Y dx$ as the void fraction of the computational domain Ω, a real number between 0 and 1. So if we assume the contribution $\psi(t)$ to be positive, then $\Delta\phi$ is positive in the vapor phase and negative in the liquid phase. The third and fourth equations show that $\mathbf{u}$ is the gradient of the potential ϕ and that the outward velocity is null on the borders of the problem; no fluid comes in or out of Ω and crosses $\partial\Omega$.

Penel et al. show in [113, 115] that there exists a unique solution to this problem with any smooth initial condition. More existence results are given in [73] by Gittel et al. One result is an analytic expression for the volume V of the bubble as a function of $\psi(t)$, which is useful for later tests:

Theorem 1. *The volume of the area where $Y = 0$ is the result of the following formula:*

$$V^{-1}(t) = \left(\frac{1}{V(0)} - \frac{1}{|\Omega|}\right)\exp(-\psi(t)) + \frac{1}{|\Omega|}. \tag{2.28}$$

The first article to present a proof is [50]. The formula is very useful for verification purposes of numerical simulations, as we will see later in Section 4.2.

2.2 Numerical modelisation of an interface

Bubbles are one of the multiple forms of the repartition of physical and chemical species in the environment of two- or multi-phase flows. It generally refers to the volume occupied by the minor phase, that-is-to-say the phase which volume fraction is the smallest. The common representation of a bubble has the approximate shape of a convex globe, but it can actually also have concave parts, have holes, be separated into multiple non-connected volumes.

The main point is that there exists a sharp separation – the interface – between the inside of a bubble and the surrounding species. When the interface is extremely thin or even has a zero thickness, then we talk about a sharp interface. On the opposite, when the interface is defined with a non-zero thickness, then we refer to the interface as a diffuse one. The choice of the scale of the physical model of the two-phase flow determines whether the interface is sharp or not. At the smallest scale of molecules, the interface is just the region where the density of molecules is rapidly evolving, so it is diffuse. At a meso-scale, we can represent this area by a zero-width area and as such as a sharp interface. On a larger scale, bubbles are represented through their impact on void fraction so we are more considering diffuse interfaces between all-liquid regions and all-gas regions. In the present PhD thesis, we will focus on sharp interfaces and the way to simulate them.

There are diverse techniques to discretise the bubble and the sharp interface, as well as their evolution in time. The two families of numerical methods we will describe here are the focus of very active current research: front tracking and front capturing.

2.2.1 Front tracking

As explained in John Dolbow's course [55], methods for "front-tracking" (FT) or "interface-tracking" are methods which use a mesh to follow the interface. As the flow evolves, the interface is updated too. It is a convective (i.e. Lagrangian) perspective, initiated by Tryggvason and co-workers [140, 138, 139].

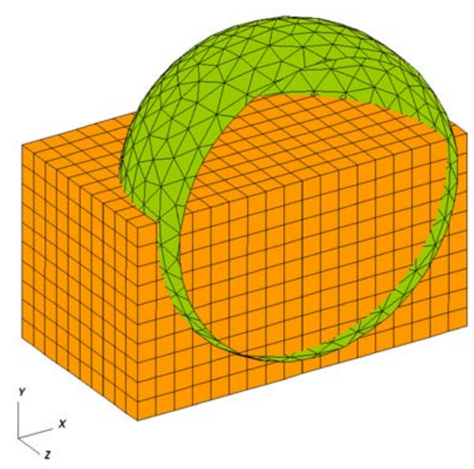


Figure 2.3 – Front tracking used in Trio_U 2.3

In this case, tracking the interface means labelling it with so-called "Lagrangian markers" or "convective markers", as done for instance by Guillaume Bois using nuclear code Trio_U [30] (see Figure 2.3). Those convective markers evolve and are convected by the surrounding velocity field $\mathbf{u}$ determined with conservative methods (also refered to as "Eulerian" methods). In other words, the velocity $\mathbf{u}_i$ of each marker is given by an interpolation based on the field $\mathbf{u}$. The numerical schemes used are of the kind of Lagrange projection.

The front tracking method is very powerful and adapted when it is essential to know the shape of bubbles. In particular, the method permits to determine curvatures and surface tension. However difficulties may arise with higher dimensions and also when dealing with topology breaks, as for instance the separation of a bubble into two or the creation of a hole in the bubble giving it the shape of a tore.

2.2.2 Front capturing

As explained in [55], methods for "front-capturing" or "interface-capturing" are methods which use a function which can be used to locate the interface. In particular, it can be an interface function marking the corresponding location of the interface and which is updated as the flow evolves. This is more of an Eulerian perspective.

Level-set

The level-set method was introduced by Osher and Sethian in 1988 [110]. As explained in John Dolbow's course [56], it is the basic framework for capturing the geometry of moving boundaries and interfaces. It works well for problems where the surface geometry evolves in rather complex ways, including with topology changes. This method is said to be much more successful than any other to represent such a large class of problems.

Let $\Sigma(t)$ be an interface between zones, moving due to a surrounding velocity $\mathbf{u}(\mathbf{x}, t)$. With the level-set method, we represent the interface as the zero-isocontour of a scalar function φ defined over the whole space:

$$\begin{cases} \partial_t \varphi + \mathbf{u} \cdot \nabla \varphi = 0, \\ \Sigma(t) = \{\mathbf{x} \mid \varphi(\mathbf{x}, t) = 0\}. \end{cases} \tag{2.29}$$

The quantity φ is called the level-set function.

We note $u_\mathbf{n}$ the component of $\mathbf{u}$ outwardly normal to the interface. If only $u_\mathbf{n}$ is known, then we can still write the system as follows:

$$\begin{cases} \partial_t \varphi + u_\mathbf{n} ||\nabla \varphi|| = 0, \\ \Sigma(t) = \{\mathbf{x} \mid \varphi(\mathbf{x}, t) = 0\}. \end{cases} \tag{2.30}$$

The variable φ has the property to keep its sign constant in the areas inside and outside the interface: this means that, for instance, $\varphi(\mathbf{x})$ could be positive if and only if $\mathbf{x}$ is located inside gas, as explicited by Equation (2.31):

$$\varphi(\mathbf{x}, t) \begin{cases} < 0 & \text{if } \mathbf{x} \in \Omega_l(t), \\ = 0 & \text{if } \mathbf{x} \in \Sigma(t), \\ > 0 & \text{if } \mathbf{x} \in \Omega_g(t). \end{cases} \tag{2.31}$$

So the zero-isocontour indeed defines the interface. Figure 2.4 shows such a level-set function: its intersection with the $\varphi = 0$ plane is represented with black dashes and has the shape of a circle. So it represents a droplet.

Often, we impose φ to be equal to a signed distance as for initial conditions. In this case, $||\nabla \varphi|| = 1$. Shortly after the initial conditions, we can expect $||\nabla \varphi|| = \mathcal{O}(1)$. Then the level set method is very powerful for the calculation of normal vectors $\mathbf{n}$ and curvatures κ.

$$\mathbf{n} = \frac{\nabla \varphi}{||\nabla \varphi||}. \tag{2.32}$$

$$\kappa = \nabla \cdot \mathbf{n}. \tag{2.33}$$

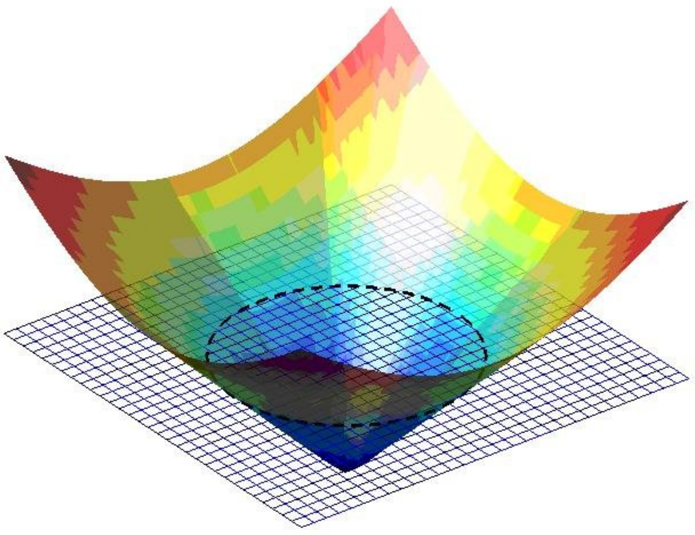


Figure 2.4 – Illustration of the level-set function [61]

Unfortunately, we know from experience that it is hard to maintain the signed distance. That-is-to-say, as shown by Abgrall et al. in [2], that the level-set method is bad at conserving mass – and conserving mass positivity in particular. Depending on the physical model (see Section 2.1), it can also lead to pressure oscillations. The level-set function has to be recalculated or reinitialised frequently, for instance with the Fast Marching Method [37]. But even with the reinitialisations, we still may face issues.

Particle level-set methods

Particle level-set methods [60] are a recent improvement to level-set methods. This is a hybrid approach between level-sets and convective markers particles. The marker particles are placed on one of the two sides of the interface – sometimes on both sides – and help reconstructing the level-set function. It is particularly useful when the interface undergoes stretching and tearing. The advantage compared to pure particle-based methods is that no connectivity between markers is stored.

This method requires a lot of work. Indeed, due to the movement of the interface, particles may sometimes not be uniformly spread any more and get concentrated in some places. In those conditions, we have to regularly "reseed" the particles to get a more uniform distribution. Enright et al. [60]

show how to effectively reseed in the case of a passively advected interface.

Volume Of Fluid

Another family of front capturing methods is the transport of the void fraction. In the case of two-phase flows, let $\alpha \in [0, 1]$ be the void fraction. For a given volume V_{total}, it is equal to the proportion of volume of gas:

$$\alpha = \frac{V_{gas}}{V_{total}}. \tag{2.34}$$

The quantity α is a real number and may virtually take all values between 0 and 1. Notice that it is only defined at large model scales where we can consider elementary volumes to contain (artifical) mixtures of phases: $\alpha = \alpha(\mathbf{x}, t)$.

The *Volume Of Fluid* (VOF) method consists in considering the void fraction α and transporting it with an advection equation. For models where there is only one local velocity of fluids $\mathbf{u}(\mathbf{x}, t)$, it writes as follows:

$$\partial_t \alpha + \mathbf{u} \cdot \nabla \alpha = 0. \tag{2.35}$$

When discretised, this leads to a grid representation where each cell is affected with its α_n value, as shown in early work from [108]. At each time step, we then have to reconstruct the interface from the information we know with methods called SLIC or PLIC [146]. For PLIC we use Young's method: a piecewise linear approximation, where the segments are normal to $\mathbf{n} = -\frac{\nabla \alpha}{||\nabla \alpha||}$. We show such a result on Figure 2.5: the numbers indicate the void fraction in each cell and the interface is reconstructed with red segments. Lesser known variants of VOF include not using an interface reconstruction, but rather an interface sharpening scheme [121], or a so-called compression stage [13]. If we want to represent a small scale problem with a definite interface, then we have to also refine the grid sufficiently. The VOF method is also used in Trio_U [30] (in combination with front tracking).

This family of methods do present some significant drawbacks, as explained in [55]. In order to reconstruct a precise enough interface, we often need a large amount of grid cells. Furthermore, the geometry is fragmented [94] and it is much more challenging to calculate curvatures, contrary to with front tracking methods. Abgrall also identifies another issue with pressure oscillations when shock waves cross the interface in [1].

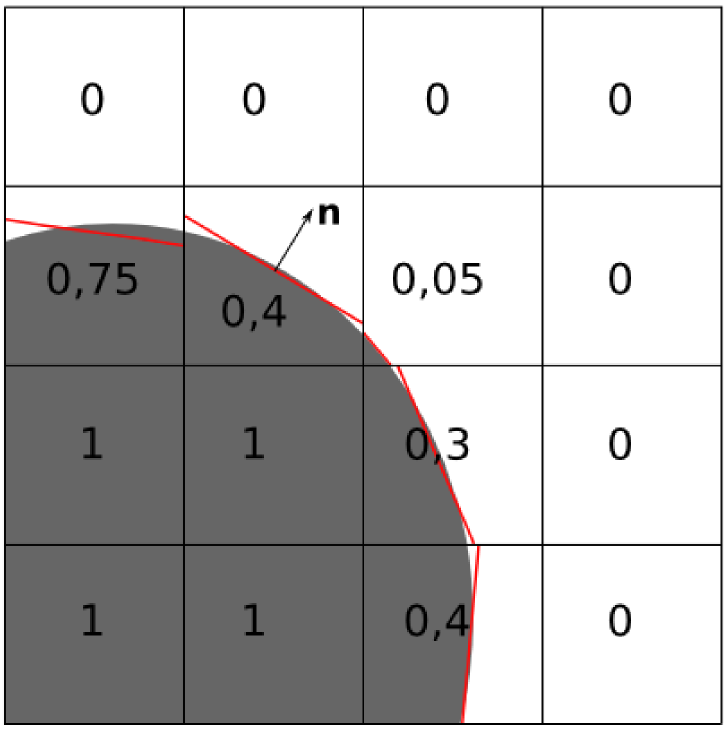


Figure 2.5 – VOF scheme interface recontruction with PLIC [82]

Colour function

Finally, we present here the front capturing method which we will use extensively: it is the transport of what we could call a "colour function", indicating the presence of a fluid. At the smaller scales and in particular at the scale of bubble-liquid interfaces, it is more appropriate to consider a colour function $Y \in \{0, 1\}$.

$$Y(\mathbf{x}, t) = \begin{cases} 0 & \text{if the liquid phase covers } \mathbf{x}, \\ 1 & \text{if the gas phase covers } \mathbf{x}. \end{cases} \tag{2.36}$$

The quantity Y is sometimes called a step-function, because of its shape when we represent it in 1D. As a side note, we can notice that for problems where we have a defined interface and thus where Y is well defined, it is also possible to define $\alpha(\mathbf{x})$ by taking V_{total} to an infinitesimal limit. In this case, $\alpha(\mathbf{x}, t) = Y(\mathbf{x}, t)$. To a larger extent, we can also give a definition of the colour function in the case of more than two phases: $Y_{phase,species}$ then is the space indicator function of the domain where a specific species in a specific phase is.

We also use an advection equation for the colour function:

$$\partial_t Y + \mathbf{u} \cdot \nabla Y = 0. \tag{2.37}$$

For the colour function method, as for the VOF method, we also often need a large amount of grid cells to represent a precise enough interface. Moreover, it is also arduous to compute curvatures and normal vectors as well. But the main challenge here will come from the fact that we want to transport a sharp interface function; any discretisation cell that exhibits a Y with a value different from 0 or 1 would be a numerical approximation. That is why it is necessary to find appropriate numerical schemes to preserve the sharpness. The technique presented in [90] is to (introduce and) sharpen the source terms. We present other applicable numerical schemes in Section 3.4.

Chapter 3

Front capturing with AMR: finite difference schemes and parallelisation

Because it is the discipline of non-averaged models, DNS computation is known to be extremely costly as far as computation time and memory are concerned. For this reason, it is interesting to alleviate the calculation load. In particular, one approach is to use locally adapted meshes for sub-domains of interest. This way, instead of using a mesh that would be fine on all the studied computational domain, we consider a problem with far less degrees of freedom. This is what is referred to as Adaptive Mesh Refinement (AMR) in regions of interest. Typically for bubble simulations, we may choose the region of interest as the vicinity of the interface between gas and liquid, as represented in red on Figure 3.1.

As explained in [18], we can distinguish two large categories of AMR:

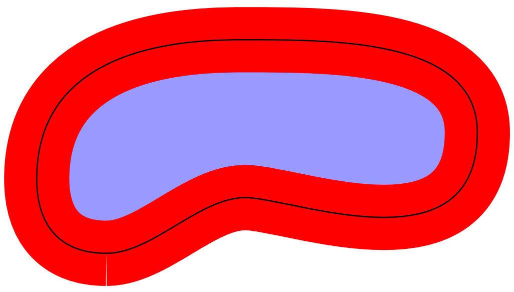


Figure 3.1 – Area of interest for bubbles; the interface

- so-called “adaptive methods”, on a unique but complex grid,
- patch-based AMR, on multiple levels of grids.

3.1 Adaptive methods

The first category of AMR refers to the use of a unique mesh with local enrichment, those are simply called “adaptive techniques”. The enrichment follows the area of interest, so the mesh is time-dependent.

3.1.1 Anisotropic meshing with adaptive metric field

In the case of anisotropic meshes, it is absolutely possible to adapt the mesh at each time step. The anistropic has simply a higher density of cells in the regions of interest. One way to do so is to define a metric field obtained from an error estimation of the solution, as presented in [105]. Figure 3.2 shows the anisotropic mesh used for the computation of a swirling bubble in 2D. We can notice that all the cells are triangle-shaped but of varying surface.

3.1.2 r-adaptive technique

The “r-adaptive methods” are methods using a moving mesh, for dynamic systems. The mesh is deformed to follow the highest variations and/or the regions of interest. To quote Cao et al. [35], to discretise an equation with moving meshes, “*we may view the time-dependent mesh over the computational domain Ω as the image of a fixed mesh over an auxiliary (fixed) computational domain $\Omega_h \in \mathbb{R}^n$*”, where n is the dimension of the problem. So we consider a function $\mathbf{x}$ defined as follows:

$$\begin{array}{cccc} \mathbf{x}: & \Omega & \rightarrow & \Omega_h \\ & (\xi, t) & \mapsto & \mathbf{x}(\xi, t). \end{array} \tag{3.1}$$

Then we can find a partial differential equation on $\mathbf{x}$:

$$\frac{\partial \mathbf{x}}{\partial t} = f\left(\frac{\partial \mathbf{x}}{\partial \xi}, \frac{\partial^2 \mathbf{x}}{\partial \xi^2}, \dots\right). \tag{3.2}$$

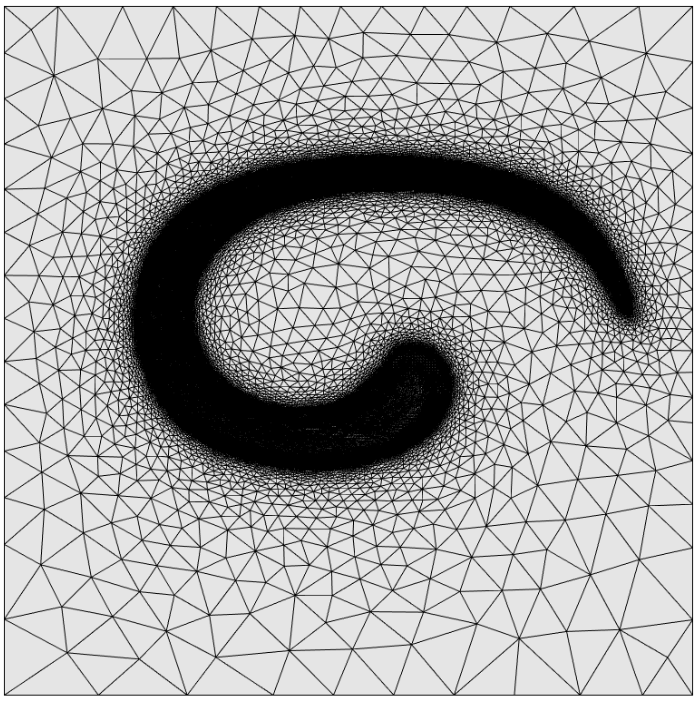


Figure 3.2 – Adapted anisotropic mesh for a swirling bubble problem [105]

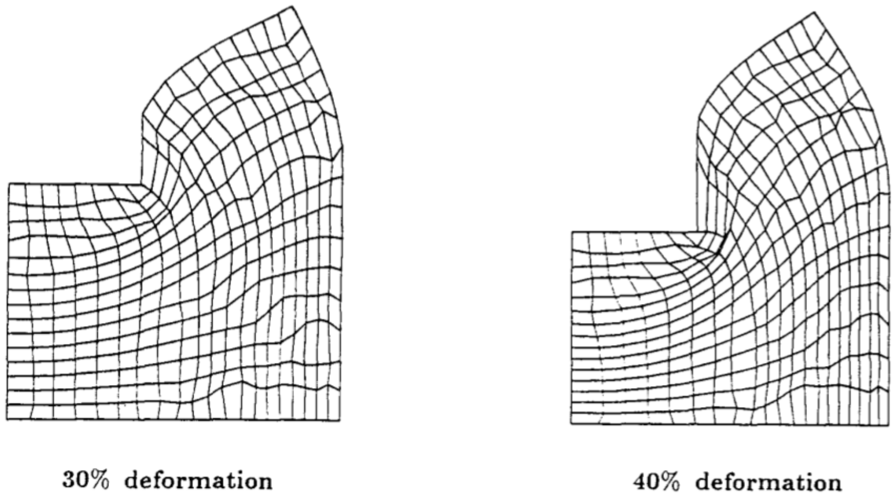


Figure 3.3 – r-adaptive mesh after deformation [72]

Figure 3.3 is extracted from [72] which studies the computation of a metal-forming simulation. It is the representation of a mesh adapted and moved after a certain deformation and thanks to r-adaptation.

A commonly used technique for r-adaptation is the Arbitrary Lagrangian Eulerian method (ALE) (see for instance [70]). For this technique, the mesh moves differently from the Lagrangian referential and the Eulerian referential. The equivalence with r-adaptive techniques is explained in [58] for example.

3.1.3 p-adaptive technique

The “p-adaptive methods” are mostly applied as an extension of the Finite Elements Method or one of its derivatives. Instead of using simple basis functions, we represent the studied functions with polynomials of higher degree, locally in the regions of interest. In other words, we keep the same number of cells and nodes, and enrich some with potentially more information.

As an example, Barros et al. use this technique to study efforts on a notched concrete plate represented on Figure 3.4 [19]. The area around and the area in front of the notch undergo the most mechanical efforts, so they would benefit from an adapted discretisation. Figure 3.5 shows the mesh used for the discretisation of the problem, as well as the degree of the basis polynomials depending on the node: the degree is free to range from $p = 1$ to $p = 6$.

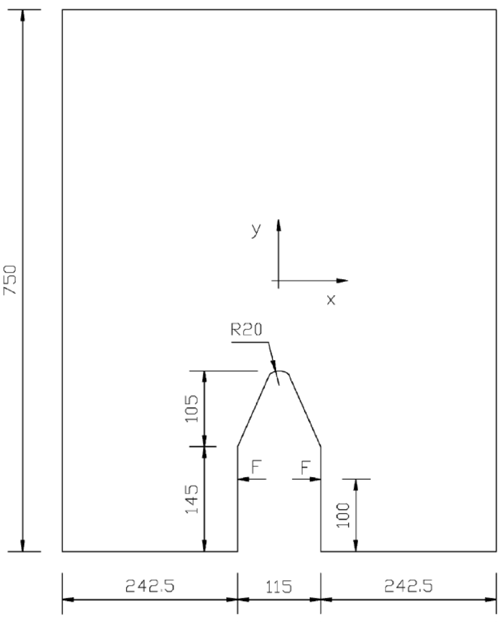


Figure 3.4 – Geometry and applied loading of study by Barros et al. [19]

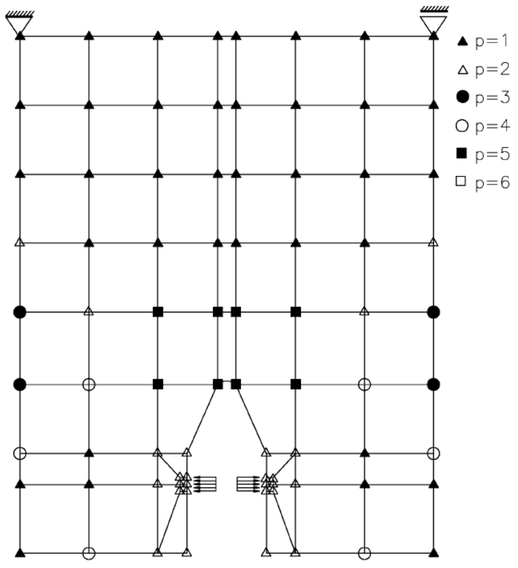


Figure 3.5 – Discretisation used at the beginning of the computation [19]

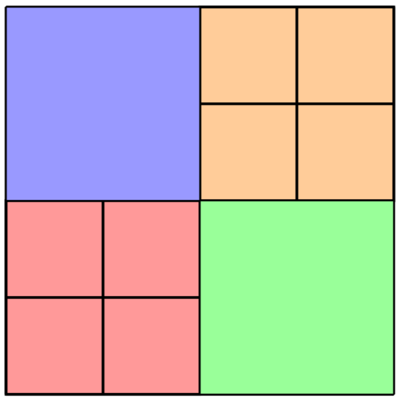


Figure 3.6 – Illustration of tree-based AMR [65]

3.1.4 h-adaptive technique

For “h-adaptive methods”, we have re-meshing or subdivision: we keep all the elements at the same order and subdivide them. We can say it is a cell-based method. As a consequence this increases the degrees of freedom of the global problem.

One of the versions which is easiest to represent is the h-adaptive method for Cartesian meshes. This case is sometimes called “tree-based”; for instance we name the resulting structure a “quadtree” when 2D cells are divided in 2 in every direction, or a “octree” when 3D cells are divided in 2 in every direction. The term “tree” refers to a sense of hierarchy between coarse cells and refined cells. Figure 3.6 is extracted from [65] and represents a 2D grid of four coarse cells, out of which two are subdivided into four finer cells. This can be done recursively, as shown on Figure 3.7 extracted from [106] and representing a bubble rising through an interface, with tree-based adaptation.

In most cases, as explained in [26], we want to keep a property of “1-irregularity”. This applies in cases where we divide cells not once but multiple times: if the original refinement coefficient r is 2, then in 2D it means we end up having cells at least $2^2 = 4$ times smaller than the original coarse cells. And in 3D in means we end up having cells at least $2^3 = 8$ times smaller than the original coarse cells. In the case of 1-irregularity, for any two neighbouring cells on the grid, we want them to differ of at most one level of refinement. We do not want to have a (non-refined) coarse cell neighbouring four little

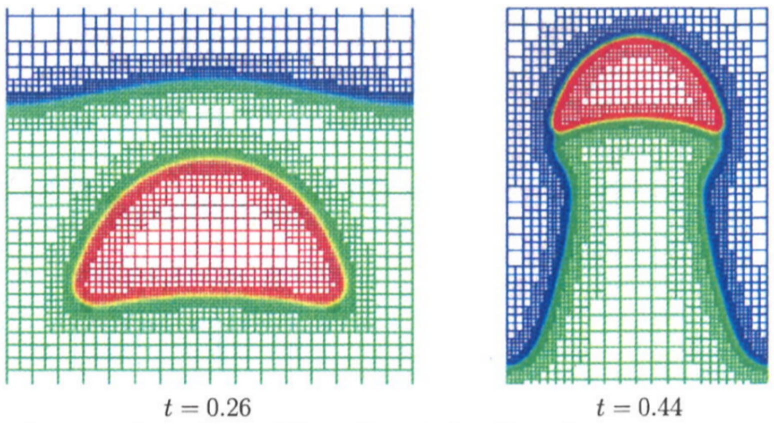


Figure 3.7 – Bubble rising through an interface [106]

cells, product of two consecutive divisions by two. This has the consequence to introduce buffers or margins of refinement.

In fact, h-adaptive methods can easily be combined with other methods for improved results. For instance, see the r-h adaptive method (e.g. in [12]) and the h-p adaptive method (e.g. in [141]).

We notice that the four latter methods (anisotropic, r, p and h) are all based on a single mesh. There is no concept of multigrid.

3.1.5 s-adaptive technique

As explained by Fish in [66] – from which Figure 3.8 is extracted –, the "s-adaptive methods" are methods of superposition of multiple grids. The superposition is then merged into one composite grid, often resolved at once. As a consequence, the number of degrees of freedom quickly becomes problematic. A variant named Arlequin [53] was proposed to treat problems where domains may have different behaviours (physical or dimensional). The s-adaptive techniques are mostly used for solid mechanics.

As we will see in Section 3.2, s-adaptive methods are not so far from patch-based adaptive mesh refinement. The main difference is whether the superposed grids are solved in one system or rather quite independently.

As a conclusion to all these 5 adaptive methods, we can remember that they present the property of resulting in one unique grid. This grid is often complex, because of its number of degrees of freedom, or because of eventual non-conformities, depending on the method.

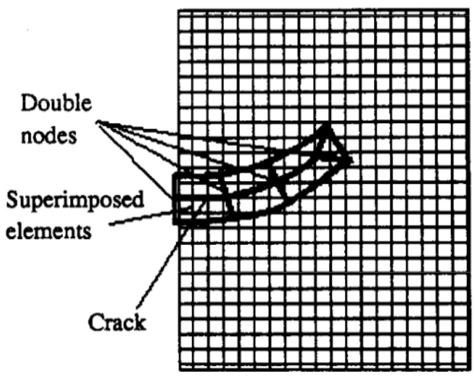


Figure 3.8 – Example of superposition of two grids for a crack problem [66]

3.2 Patch-based mesh adaptation on Cartesian grids

The other large category of AMR, on which we will focus from now on, is the patch-based adaptation. It is sometimes also referred to as block-based AMR or multi-level AMR. It was introduced in 1984 by Berger and Oliver in [21] and later improved for shock problems in [20].

This technique is typically applied on 1D, 2D and 3D Cartesian grids. The first step consists in marking (or "flagging") some cells of the coarse grid as meriting a refinement: typically the cells where variations are larger and/or where an interface between phases is located. This step is similar to what should be done with the tree-based refinements seen before in Section 3.1. Then the flagged cells are bundled together in so-called "patches" which are areas containing one or several flagged cells. As far as Cartesian meshes are concerned, those patches are segments in 1D, rectangles in 2D and parallelepipeds in 3D; in other words, boxes. The union of the patches is sometimes called a "patchwork" [84] and it should contain all flagged cells and thus it is common that they also contain non-flagged cells. Although it is possible to use the patch-based method on non-Cartesian grids, from now on we will assume to be in a Cartesian environment.

Once the region of interest is covered by patches, the patches are the definition of the refinement region: we define for each patch a subgrid which is a regular Cartesian division of the coarse grid contained in the patch. This defines a finer level of refinement in addition to the original coarse grid. We will see in Section 3.2.2 that we may apply this logic recursively and create even finer patches inside the first level patches. This is what is illustrated

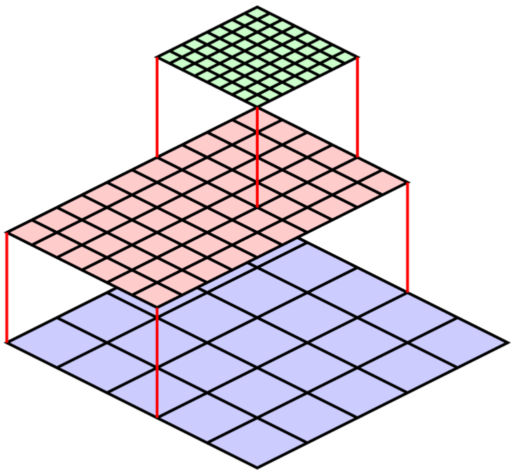


Figure 3.9 – Illustration of patch-based AMR [65]

on Figure 3.9. As each patch is a Cartesian grid too, we can use finite differences schemes very elegantly and we can opt to keep the same scheme for all refinement levels.

Patch-based AMR is very powerful because of the rather simple subproblems it creates. Each patch can indeed be considered as its own specific problem, provided we take care at some point of the communication between this elementary patch problem, its neighbouring patches, its coarser-level patch and eventually its finer-level patches. As a consequence, as we will see in Section 3.3 about the computer implementation, we can distribute the patches quite individually for computation by several threads.

As a side remark, let us notice that multi-grid approach and the tree-based one could be made to cohabit, for instance for preconditioning purposes as shown in Minjeaud [106] or Boyer [32] for example.

3.2.1 Clustering algorithm

Input parameters

In this work, we examine three different methods for the definition of the patches covering the region of interest. Those are three algorithms to construct the cluster of patches according to some criteria: we will call them patch creation algorithms, clustering algorithms or even boxing algorithms.

The algorithms take different input parameters, some of which are shown

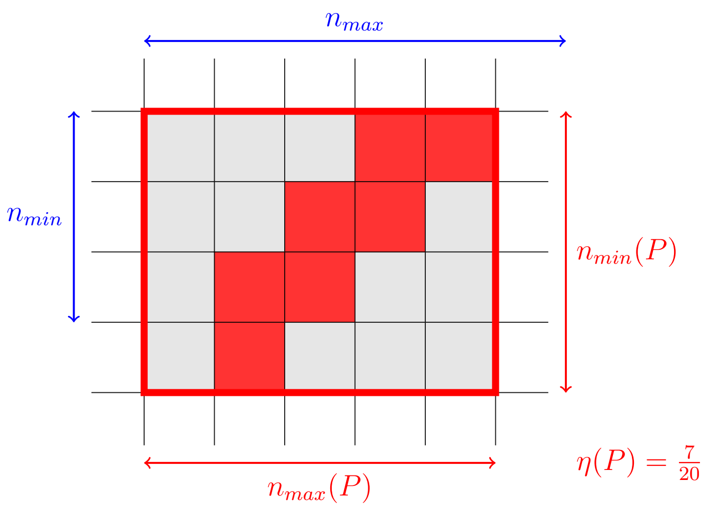


Figure 3.10 – Some input parameters for patch creation algorithms

on Figure 3.10. On that figure, we represent a two-dimensional patch of coarse cells by using a red rectangle. The cells filled in red are the ones for which the refinement criterion detected that they are of interest. Let us call that patch P.

The first input parameter of the algorithms is the efficiency goal and will be noted η_{min}. Let us call $\eta(P)$ the ratio between number of "flagged" cells (i.e. of interest) and the number of cells of the patch:

$$\eta(P) = \frac{\text{number of flagged cells of } P}{\text{number of cells of } P}. \tag{3.3}$$

According to our colour scheme on Figure 3.10, it is the ratio between red cells and the total number of cells of the rectangular patch, so we can compute $\eta(P) = \frac{7}{20}$. Then η_{min} is the minimum ratio between flagged area and patch area that the clustering algorithm may generate, for all patches. In other words, we want $\eta(P) \geq \eta_{min}$. It ensures that the refined grid calculation is concentrated on the areas of interest which are flagged.

The second input parameter is the minimum number of cells a patch has in any direction (x, y or z), noted n_{min}. The quantity n_{min} is an integer and is counted in number of cells. It is a purely geometrical parameter. Let the space be discretised with a mesh of elementary space-step Δx in all directions. Then we can call $L_{min}(P)$ the minimum length a patch has in any direction and we can write $L_{min}(P) = n_{min}(P)\Delta x$. In our 2D example of Figure 3.10, the shortest length is equal to 4 times the elementary length of a cell for this patch. So we have respected the inequality $n_{min}(P) = 4 \geq n_{min} = 3$.

The third input parameter is simply the maximum number of cells a patch has in any direction (x, y or z), noted n_{max}. The quantity n_{max} is an integer and is counted in number of cells. It also is a purely geometrical parameter. Let the mesh have an elementary space-step of Δx in all directions. Then we can call $L_{max}(P)$ the maximum length a patch has in any direction and we can write $L_{max}(P) = n_{max}(P)\Delta x$. In our 2D example of Figure 3.10, the longest length is equal to 5 times the elementary length of a cell for this patch. So we have respected the inequality $n_{max}(P) = 5 \leq n_{max} = 6$. Of course, n_{max} should be greater than or equal to n_{min}.

The fourth input parameter we consider is the maximum number of cells of a patch, noted N_{max}. The quantity N_{max} is an integer and is counted in number of cells. It can be considered as a maximum surface in 2D and a maximum volume in 3D. Let the mesh have an elementary space-step of Δx in all directions. Then in 2D we can call S_{max} the maximum allowed surface and in 3D we can call V_{max} the maximum allowed volume. We can then write respectively $S_{max} = N_{max}(\Delta x)^2$ and $V_{max} = N_{max}(\Delta x)^3$. As our Figure 3.10 is in 2D, we will think in terms of surface: for this particular patch P its number of cells is equal to $N_{max}(P) = 20$ cells. If we suppose that the patch respects the N_{max} parameter, then we can conclude that $N_{max} \geq 20$.

Berger-Rigoutsos

The Berger-Rigoutsos clustering algorithm takes its roots in the introduction of patch-based AMR in [21], followed by [20]. The clustering algorithm was improved in [22] and has been analysed by Livne in [96]. The idea is to set a η_{min} as the sole constricting goal. As we do not set the three other geometrical input parameters, there is no constraint on the geometry.

Let P be a 2D patch of dimensions $n\Delta x \times m\Delta y$. We number the cells of P with $(i,j) \in [\![0, n-1]\!] \times [\![0, m-1]\!]$. Let $f(i,j)$ be the flagging function: it equals 1 if the cell (i,j) is flagged, 0 otherwise. We call ς_x be the "signature" of a (vertical) column and ς_y the signature of a (horizontal) row:

$$\varsigma_x(i) = \sum_{j=0}^{m} f(i,j), \tag{3.4}$$

$$\varsigma_y(j) = \sum_{i=0}^{n} f(i,j). \tag{3.5}$$

Here is how the Berger-Rigoutsos clustering algorithm works, in a recursive fashion explained in [96]. The initialisation step is to consider the whole computational domain as one large step. Then, for every iteration and for every patch, we consider splitting it into two, alternatively in a direction x or y. For example, in the vertical direction y, we do the following five steps.

1. We test whether the patch contains flagged cells. If not, we drop the patch (we do not keep it) and we do step 1 again with another patch candidate. If yes, we continue to step 2.

2. We compute the signature for all i. If there is a coordinate i_{zero} at which $\varsigma_x(i_{zero}) = 0$, we make the cut there and go to step 5. Otherwise, we continue to step 3.

3. We compute a discretised second derivative of ς_x. If there is an inflexion point $i_{inflexion}$, we make the cut here and go to step 5. Otherwise, we continue to step 4.

4. We test whether the patch has an associated efficiency greater than η_{min} or not. If not, we cut the patch in the middle. In any case, we continue to step 5.

5. If we did not cut the patch, we keep it for the final collection of patches and we proceed to step 1 with another patch candidate. If we did cut the patch, we proceed to step 1 again for the two resulting subpatches, but this time in the horizontal direction x.

In the example given in Figure 3.11, we consider a bubble in a liquid medium. We flag the cells (here coloured in red) which are on or close to the interface between the two phases. Then, we cover this area of flagged cells with patches using the Berger-Rigoutsos algorithm. The patches are displayed as white-bordered rectangles.

We set the input minimal efficiency at $\eta_{min} = 0.4$. This is a low coefficient if we compare it to common practices, but it is relevant for the comparisons we will do in Section 3.3. As only covering efficiency is taken into account, it is not surprising to see that the patches follow the interface quite closely.

Livne

Livne introduced an improvement of the Berger-Rigoutsos algorithm in [99]. In addition to η_{min}, two more parameters are given as inputs: n_{min} and N_{max}. In fact, the minimum efficiency becomes a second-priority input parameter:

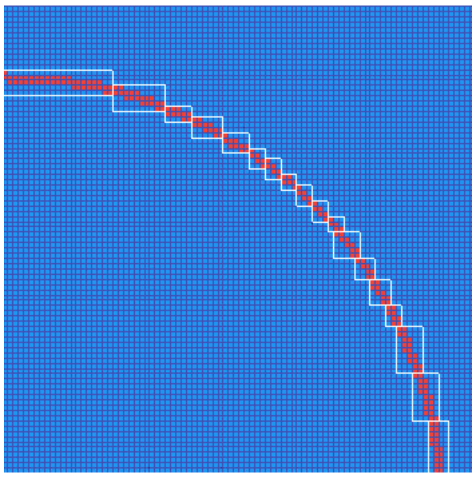


Figure 3.11 – Bubble-liquid interface covered by patches with the Berger-Rigoutsos method

in other words it is essential the algorithm respects the geometric requirements that all patches length be longer than n_{min} cells and that the number of cells in the patches be smaller than N_{max}. If this competes with the η_{min} objective, then the algorithm chooses to respect the geometry input. To take a practical example, we can have patches with all lengths comprised between L_{min} and $2L_{min}$ and still with an efficiency smaller than η_{min}, because cutting this patch into two would result in at least one patch with a minimum length smaller than the goal L_{min}.

The algorithm functions similarly as explained previously for Berger-Rigoutsos in Section 3.2.1. But in addition, we impose to cut all patches which have more than N_{max} cells and we forbid to cut any patch shorter than n_{min} in any direction. This brings comparative advantages discussed later in Section 3.3.4: the patches have more similar sizes when compared to each other.

In the example given in Figure 3.12, we give the input parameters $\eta_{min} = 0.4$, $n_{min} = 5$ and $N_{max} = 10^2$. Notice that the minimum efficiency goal is the same as in the example of Figure 3.11 (Berger-Rigoutsos).

$\boldsymbol{n_{min} - n_{max}}$

The third and last algorithm we will analyse here was introduced in [130]. As Livne's algorithm, it also is a modification of the Berger-Rigoutsos algorithm, based on the same structure. But simply put, instead of using the input parameters η_{min}, n_{min} and N_{max} as with Livne, we suggest to use

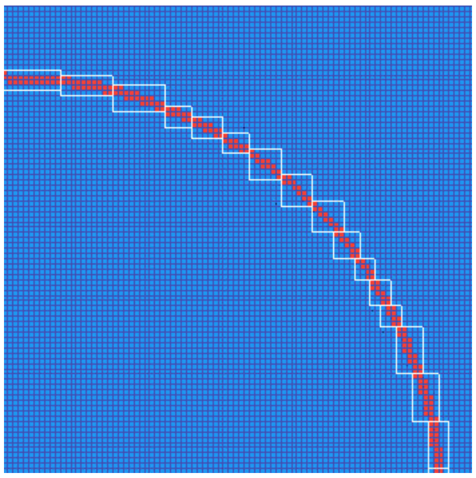


Figure 3.12 – Bubble-liquid interface covered by patches with the Livne method

the minimum efficiency η_{min}, the minimum number of cells in any direction n_{min} and the maximum number of cells in any direction n_{max}. Similarly to Livne, the geometry parameters n_{min} and n_{max} are the priority arguments, in eventual cases when they may compete with η_{min}.

The algorithm functions similarly as explained previously for Berger-Rigoutsos in Section 3.2.1. But in addition, we impose to cut all patches longer than n_{max} in any direction and we forbid to cut any patch shorter than n_{min} in any direction. As for the Livne algorithm, this brings comparative advantages discussed later in Section 3.3.4: the patches are more "square".

In the example given in Figure 3.13, we give the input parameters $\eta_{min} = 0.4$, $n_{min} = 5$ and $n_{max} = 10$. Notice that the minimum efficiency goal is the same as in the example of Figure 3.11 (Berger-Rigoutsos) and 3.12 (Livne) and that the minimum number of cells in any direction n_{min} is the same as in the example of Figure 3.12 (Livne). Also, we choose n_{max} such that $(n_{max})^d = N_{max}$ where d is the dimension of our problem (here $d = 2$) and N_{max} is the maximum number of cells of Livne's algorithm.

For reference, the implementation of the three clustering algorithms we present is shown in Appendix A.

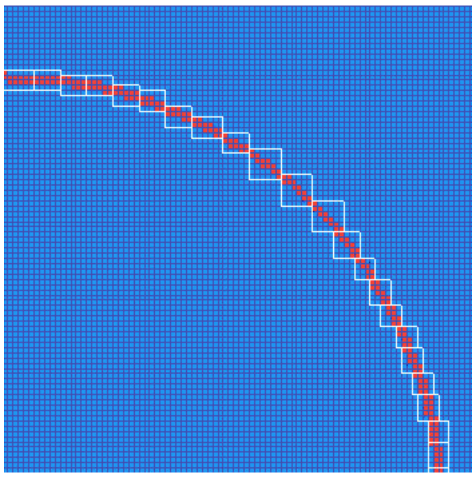


Figure 3.13 – Bubble-liquid interface covered by patches with the $n_{min} - n_{max}$ method

Evolution in time

For time-driven problems, the area of interest may evolve and move. In that case, the flagged area may change and not be covered by patches any more, which would be an issue. Different solutions can be envisioned, as explained in [98]. The question is whether it is necessary to recompute the whole clustering algorithm on each time step, or on the contrary we can use the patches from time t_n for time t_{n+1}.

If the clustering algorithm is very efficient and costs very little in terms of computation, then the most straightforward method is to recompute a patch-coverage for each time step. However, if the computation is very costly, then it may be interesting to save the patches location for a few time steps. One strategy is to flag some extra cells around the cells of interest, and this way to cover a larger area with patches. This way we hope that the region of interest (like the bubble interface for instance), when it moves (because of advection or dilation for instance), still stays in the patch-covered area for a few time steps. This can be linked to a regular test to see whether the patches still sufficiently cover the region of interest.

As far as our computations are concerned, a lot of efficiency progress has been made on the efficiency of the clustering algorithm. So the clustering is near instantaneous. This is why we choose to recompute from scratch the patch distribution for every time step. As a consequence, our clustering evolves with every time step.

3.2.2 Multi-level

Patch-based AMR offers the possibility to consider subpatches of patches, that-is-to-say nested subdomains. This means that if we start with a coarse grid (level 0), we can define patches in which we define an additional and finer grid (level 1). Recursively, those level 1 patches can be the hosts for patches inside them, thus defining a level 2 and so on. Khadra et al. call this "adaptive ZOOM", where ZOOM stands for "ZOom Overlapping Multi-level" [88, 87]. They show this concept can be reused for any multigrid AMR methode including LDC (see Section 4.1), FAC ([102, 103, 101]) or FIC [8]. Figure 3.9 shows a multigrid configuration with two levels of finer refinement.

The biggest advantage of using nested refinement levels is one of economy, both of calculation and of memory. The alternative would indeed be to have a coarse grid and directly a set of patches with an extremely fine grid. Here, as shown in [97], we want to minimise the number of "too small patches" by, at the same time, avoiding an imbalance between the extreme levels of refinement, by going through one or many intermediary levels.

As the levels are supposed to enrich one another, for instance with a defect correction, we have some liberty as far as this exchange is concerned. Let us enumerate the levels with the index l, where $0 \leq l \leq l_{max}$. For AMR with only two levels, we always have a simple exchange pattern similar to what we show on Figure 3.14, that is simply going back and forth between levels $l = 0$ and $l = l_{max} = 1$. However there is more freedom when we have more than two levels ($l_{max} \geq 2$). One first pattern of exchange can be named the "two times V" strategy, shown on Figure 3.15. It means that the data from the coarsest grid ($l = 0$) is transmitted to the finer ($l + 1$) until the finest ($l = l_{max}$) is reached, and then that the data from the finest grid is transmitted back as rapidly as possible to enrich the data of the coarsest grid ($l = 0$), by passing through all the other intermediary grids. The second pattern of exchange can be named the "W" strategy, shown on Figure 3.16. It means that the data of a finer grid on level l will not benefit a grid of level $l - 2$ unless it has fully enriched the data of the grid directly coarser (intermediate, $l - 1$). Some use the terms "V-Cycles" and "W-Cycles", as in [8].

Finally, we can notice that the multigrid approach – and even more so when there are many intermediary grids – can favour the choice of also a multimodel approach. We can imagine that the models used to represent phenomenons could be dependent on the characteristic lengths and thus the grid size, so some levels could get different physical models. This is not the

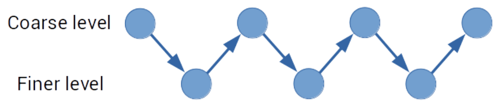


Figure 3.14 – Exchange pattern for just two levels

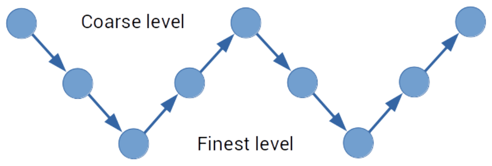


Figure 3.15 – Exchange pattern for more than two levels: “two times V” strategy

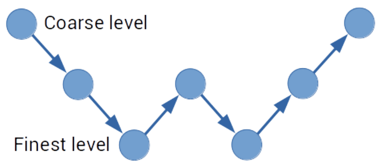


Figure 3.16 – Exchange pattern for more than two levels: “W” strategy

scope of this research. More details about the coupling of fine models with coarse models can be found in [123] or [142].

3.2.3 Efficiency of a patch covering and quality function η

Once we covered the regions of interest of a problem with patches, we may want to quantify how well it is covered. The covering indeed depends on the clustering algorithm for instance. In many if not most cases, the patches do not cover only flagged cells, but also cells not contained in the region marked as interesting. As a consequence, in the case of Adaptive Mesh Refinement, we are going to make refined computations on non-interesting cells which induces a waste of computation power.

Let us index the $n_{patches}$ patches with i and N_i the number of cells of patch i. Furthermore, if $\mathcal{F} = \{f_i\}_i$ is a family of real numbers with n_F members, then let us note $M(\{f_i\}_i)$ the arithmetic average of the family $\mathcal{F}$, such that

$$M(\{f_i\}_i) = \frac{1}{n_F} \sum_i f_i. \tag{3.6}$$

Then, if each patch i has a patch efficiency η_i as defined before in Equation (3.3), then we can define the quality function η as the "average efficiency":

$$\eta = M(\{\eta_i\}_i). \tag{3.7}$$

The quantity η is a real number comprised between 0 and 1. To minimise unnecessary computation, we want η to be as close to 1 as possible.

3.3 Multiprocessing

The computation of rich problems like we have with DNS becomes quickly demanding. So it is interesting to search for ways to speed up the computation time and shrink it from days to hours for instance. It is true that technological advances lead to the production of faster processors; see Moore's law which observes empirically that the density of transistors in microprocessors doubles every 18 to 24 months [107]. But this can be considered as not sufficient enough. One good way to accelerate computation more is to make use of new multi-threading architectures.

There is a wide variety of such machines (see [42] for instance). The material architecture has a large impact on the best way to program in order to take advantage of the specificities, as well as a large impact on the algorithm designs, as we will see in what follows.

The principle of multiprocessing is to find a way such that a large calculation is divided in smaller problems. We then send each of these problems on an independent thread, sometimes referred to as “slave-threads”. After the distributed calculation, results are gathered back to the so-called “master-thread” and interpreted centrally. In addition, sometimes, the process will require specific communications between the master-thread and the slave-threads, or between the slave-threads. This whole masther-slaves structure can be recursively set and slaves can have what could be considered as slaves of slaves.

In our case of the simulation of bubbles using patch-based AMR, our strategy is to distribute the computations of all the patches as independent problems. Ideally, this means that for n patches, we have n threads and we distribute the patches among the pool of threads. This way each thread is responsible for the computation on its own independent patch-limited problem. Then the results are sent back to the master-thread to generate a complete picture of the multi-level problem on the whole representation computational domain.

To be more specific, the strategy that we implement is centred on the first level of refinement. This means that the computation of the coarse grid (level 0) is made by the master-thread. Then we distribute the $n_{patches}$ level-1 patches between the $n_{threads}$ threads that the workstation can generate, as evenly as possible. If we decided to make a multi-level patch-based AMR where the number of levels of refinement is greater than or equal to 2, then we keep the same implementation as explained just before. That-is-to-say, each thread will be responsible for its assigned level-1 patches, as well as all all the finer levels patches which are slaves of those level-1 patches. This is the parallelisation strategy used by Almgren et al. in [4] for a cosmology simulation powered by block-based AMR on some 50000 cores. In the rest of this section, we will focus on AMR computation done with only one level of refinement, with just a coarse grid covered by patches provided a finer grid.

3.3.1 Load balance and quality function σ

When considering a parallelisation strategy, it is important to correctly balance the load between the available threads. The goal is that all the threads have approximately the same quantity of calculations to do, measured in computation time. This way, all the threads will finish the distributed tasks at the same time. The opposite situation would be when some faster threads will have finished their task first and will have to wait for slower threads. This wait is wasted time for other computation, so we want to minimise it.

In our computations, the load is proportional to the number of cells. This is independent from the size of the said cells, so it is not directly related to the space resolution of the finer grid. In order to quantify the disparity of the load carried by the patches, we introduce a quality function σ. The real number σ is the "normalised standard deviation " of the number of cells of all the patches of the fine grid. Before giving a formula for σ, let us give some notations.

Similarly as for the quality function η, let us define $M(\star)$ as the arithmetic average and let us index the patches with i. Similarly, we will note $\max(\{f_i\}_i)$ the maximum element of $\mathcal{F}$ and $\min(\{f_i\}_i)$ the minimum element of $\mathcal{F}$. As before, N_i stands for the number of cells in patch i. Then the formula for the normalised standard deviation of sizes σ reads as follows:

$$\sigma = \sqrt{\frac{M(\{N_i^2\}_i) - M(\{N_i\}_i)^2}{\max(\{N_i^2\}_i) - \min(\{N_i\}_i)^2}}. \tag{3.8}$$

The quantity σ is a real number comprised between 0 and 1. For a good load balance, we want σ as close to 0 as possible.

Limit As a side note, it is possible to find some "pathological" cases where σ would be small and the distribution of patches would still be ill-balanced. For instance, let us take the example where we have n_p patches in total. An amount of $n_p - 2$ patches are of size N, one patch is very small and of size 1 and one is very large and of size $2N - 1$. In other words, two patches are outliers. Then we have the following equalities:

$$M(\{N_i\}_i) = N, \tag{3.9}$$

$$M(\{N_i^2\}_i) = \frac{1}{n_p}\left((n_p + 2)N^2 - 4N + 2\right), \tag{3.10}$$

$$\sigma^2 = \frac{1}{n_p}\frac{2N^2-4N+2}{4N^2-4N} = \frac{1}{n_p}\frac{1}{2N}\frac{(N^2-2N+1)}{N-1} = \frac{1}{n_p}\frac{N-1}{2N}. \tag{3.11}$$

So if n_p is large, we may have a small normalised standard deviation, but the load is very unbalanced.

This pathological condition could be avoided if we compute the deviations of higher orders. But in what follows, we will assume we are not in extreme conditions and we will stick to the deviation of second order σ.

3.3.2 Communication minimisation and quality function γ

For certain technologies of parallelisation, communication between threads is very time-consuming. This concerns distributed memory parallelisation as with MPI for instance, as opposed to shared memory parallelisation as with OpenMP for instance. If we choose to use distributed memory parallelisation, for example with a view to using a computation cluster, then we should minimise the communication between the patches of the fine level. In our case of non-overlapping patches, communication may occur on the borders of the patches.

From a purely geometry point of view, in 2D we are trying to find the shape that minimises its perimeter for a given area. In the case of necessarily rectangular patches on a 2D-Cartesian grid, this shape is not a circle but a square. On a 3D-Cartesian grid, we are trying to find the shape that minimises the surface for a given volume; this would be a cube.

For a given 2D or 3D patch, let the shortest side be the one with smallest length, among respectively the 2 possible sides (x or y) or the 3 possible edges (x, y or z). Let the longest side be the side or the edge with the largest length. We can define the "squareness" γ_i of a patch indexed by i by the ratio between the length of its shortest side and the length of its longest side:

$$\gamma_i = \frac{length\ of\ shortest\ side}{length\ of\ longest\ side}. \tag{3.12}$$

The quantity γ_i is equal to exactly 1 for square patches and cubic patches.

As in last subsection, let us define $M(\star)$ as the arithmetic average. Then we can define the quality function γ as the "average squareness" of the squarenesses of the patches:

$$\gamma = M\left(\left\{\frac{length\ of\ shortest\ side\ of\ patch\ i}{length\ of\ longest\ side\ of\ patch\ i}\right\}_i\right). \tag{3.13}$$

The quantity γ is a real number comprised between 0 and 1. For a minimisation of communication between patches, we want γ as close to 1 as possible.

3.3.3 Calculations speed-up

Let us consider a computation which duration can be measured; we call it the computation in original set. Let us also consider another computation with equivalent results, but made within a shorter time; we call it the computation in modified set. The speed-up of the computation is a quantification of the improvement of the calculation time. It has been introduced with Amdahl's law in [7] and its simplest expression is the following in a very general sense:

$$su = \frac{computation\ time\ in\ original\ set}{computation\ time\ in\ modified\ set} = \frac{\mathcal{T}_{original}}{\mathcal{T}_{modified}}. \tag{3.14}$$

The computation in modified set takes a shorter time, so su is a real number larger than 1.

In our case, we intend to improve the computation time with two main consecutive contributions: first the Adaptive Mesh Refinement instead of a fine grid on all the computed space, and then our parallelisation strategy. As a consequence, we can distinguish several possible speed-ups:

$$su_{AMR} = \frac{computation\ time\ with\ a\ fine\ grid\ everywhere}{computation\ time\ with\ AMR} = \frac{\mathcal{T}_{fine,seq}}{\mathcal{T}_{AMR,seq}}, \tag{3.15}$$

$$su_{\parallel} = \frac{computation\ time\ with\ AMR\ in\ sequential\ set}{computation\ time\ with\ AMR\ and\ parallelisation} = \frac{\mathcal{T}_{AMR,seq}}{\mathcal{T}_{AMR,\parallel}}, \tag{3.16}$$

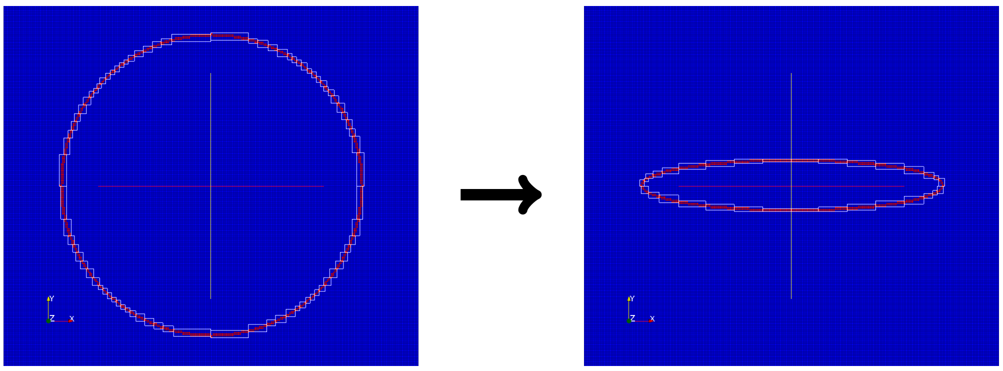
$$su_{AMR,\parallel} = \frac{computation\ time\ with\ a\ fine\ grid\ everywhere}{computation\ time\ with\ AMR\ and\ parallelisation} = \frac{\mathcal{T}_{fine,seq}}{\mathcal{T}_{AMR,\parallel}}. \tag{3.17}$$

In theory, we have the following formula:

$$su_{AMR,\parallel} = su_{AMR} \times su_{\parallel}. \tag{3.18}$$

Others like Delage-Santacreu et al. worked on su_{AMR} specifically [54]. However as for us, we will concentrate our efforts in what follows on the speed-up which is due to the parallelisation strategy, that-is-to-say $su_{\parallel}$.

Intuitively, we want to improve (i.e. diminish) the computation time with a higher number of threads $n_{threads}$. We can say that su is an increasing



(a) Dilation factor = 1

(b) Dilation factor = 6

Figure 3.17 – Ellipsoidal bubbles covering test cases

function related to the number of threads. Optimally, the speed-up would be equal to the number of threads; a computation with twice the number of threads would be twice as fast as the sequential one. However this is rarely the case and the purpose is to be as close to $su(n_{threads}) = n_{threads}$ as possible.

3.3.4 Quantitative comparison of quality functions

In order to compare the quality of the three clustering algorithms, we compare their quality functions η, σ and γ on several test cases. The collection of test cases we choose is the covering of a thin interface by patches, for ellipsoidal bubbles which are more or less dilated.

On a 2D grid of $300\Delta x \times 300\Delta x$, we define an ellipsoidal bubble which has a ratio between its longest radius and shortest radius equal to $\digamma$, the dilation factor, as shown on Figure 3.17. We make $\digamma$ vary from $\digamma = 1$ (Figure 3.17a) to $\digamma = 6$ (Figure 3.17b) with a few hundreds steps. For each step, we redo the computation of each algorithm (Berger-Rigoutsos, Livne and $n_{min} - n_{max}$) in order to cover the interface between the bubble and its surroundings. The numerical input parameters for each algorithm are given in Table 3.1. We choose them identical between algorithms (when it makes sense), such that the comparison holds.

Figure 3.18 shows the resulting efficiency of the patch covering, as a function of the dilation factor. The different colours of the graphs stand for each patch-creation algorithm: red for Berger-Rigoutsos, blue for Livne and green for $n_{min} - n_{max}$. As a reminder, the closer η is to 1, the better the covering is

	η_{min}	n_{min}	n_{max}	N_{max}
Berger-Rigoutsos	0.4			
Livne	0.4	5		10^2
$n_{min} - n_{max}$	0.4	5	10	

Table 3.1 – Input parameters for our ellipsoidal bubble test cases

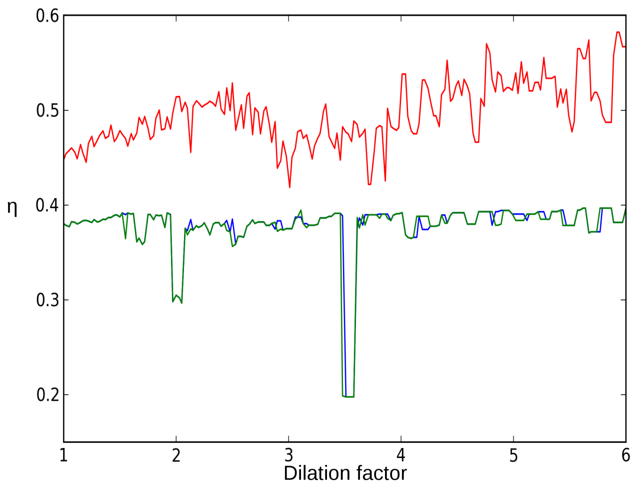


Figure 3.18 – Resulting efficiency η, as a function of the dilation factor and of the patch creation algorithm (higher is better)

because it minimises the unnecessary computation. Clearly, the conclusion for this figure is that the Berger-Rigoutsos algorithm seems to be better for this aspect.

Figure 3.19 shows the normalised standard deviation of the patch covering, as a function of the dilation factor. We chose the same colour code as before. As a reminder, the closer σ is to 0, the better the covering is because it permits a good load balance. The conclusion of this figure is that $n_{min} - n_{max}$ seems to be better for this aspect.

Figure 3.20 shows the average squareness of the patch covering, as a function of the dilation factor. Again, we chose the same colour code as before. As a reminder, the closer γ is to 1, the better the covering is because it theoretically minimises communication. The conclusion of this figure is that $n_{min} - n_{max}$ seems to be better for this aspect.

Table 3.2 summarises our last statements. Although we have studied only

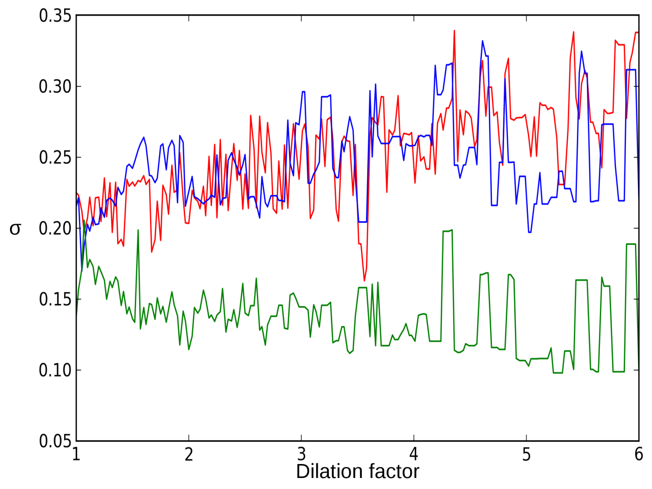


Figure 3.19 – Normalised standard deviation σ, as a function of the dilation factor and of the patch creation algorithm (lower is better)

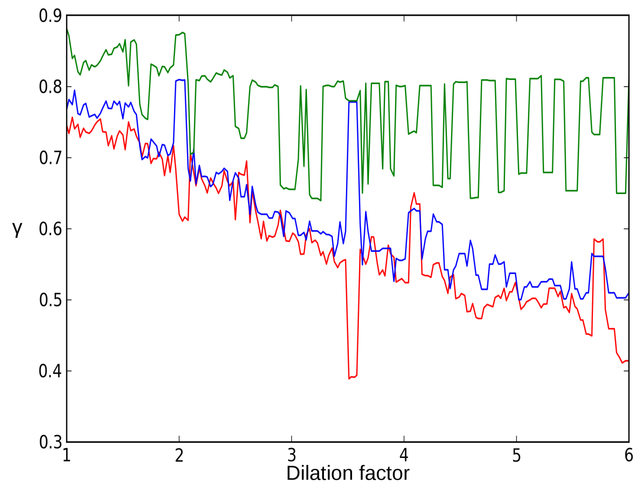


Figure 3.20 – Average squareness γ, as a function of the dilation factor and of the patch creation algorithm (higher is better)

	Berger-Rigoutsos	Livne	$n_{min} - n_{max}$	Note
η	0.503 (0.037)	0.454 (0.042)	0.463 (0.043)	higher is better
σ	0.270 (0.035)	0.234 (0.038)	0.134 (0.031)	lower is better
γ	0.603 (0.093)	0.623 (0.097)	0.778 (0.049)	higher is better

Table 3.2 – Comparison of the average of quality functions of each algorithm on a series of tests (+ standard deviation of quality function)

some particular cases, we can use this study to make a selection of the right algorithm.

- For sequential programming, efficiency is the foremost priority. So the Berger-Rigoutsos algorithm seems more appropriate.
- For multiprocessing, we want to balance CPU load and to minimise CPU communications. So the $n_{min} - n_{max}$ algorithm seems more appropriate.

Therefore, we choose $n_{min} - n_{max}$ as our patch covering algorithm, which we will later call $\mathbb{A}$. Our choice remains to be confirmed with a real application, as we will proceed in Section 3.4.

3.4 Application: pure advection

In order to have some real-life indications of how the different clustering algorithms perform, we used the canonical Kothe-Rider test introduced in [119]. It is the simulation of the deformation of a sphere due to a time-periodical advection velocity field.

Let the space domain be a unit cube $[0, 1] \times [0, 1] \times [0, 1]$. We locate a sphere with the coordinates (x_0, y_0, z_0) for its centre. It represents a bubble with a radius R. In our example, the numerical values are the following:

- $x_0 = 0.35$,
- $y_0 = 0.35$,
- $z_0 = 0.35$,
- $R = 0.15$.

The inside of the sphere represents a gas bubble, which void fraction $\alpha(x, y, z, t)$ or colour index $Y(x, y, z, t)$ is equal to 1. The outside, on the contrary, represents the liquid surrounding. So its void fraction or colour index is equal to 0. As we are reasoning with small DNS scales, we want to think in terms of sharp interface, theoretically without mixtures. So we will keep the notation $Y \in \{0, 1\}$ in what follows.

The field Y is advected by a velocity field $\mathbf{u}$:

$$\partial_t Y + \mathbf{u} \cdot \nabla Y = 0. \tag{3.19}$$

Let $\mathcal{T}$ be a time period. The velocity field is set to be a periodical function of time:

$$\begin{cases} u_x(x, y, z, t) = 2\sin^2(\pi x)\sin(2\pi y)\sin(2\pi z)cos(2\pi\frac{t}{\mathcal{T}}), \\ u_y(x, y, z, t) = -\sin(2\pi x)\sin^2(\pi y)\sin(2\pi z)cos(2\pi\frac{t}{\mathcal{T}}), \\ u_z(x, y, z, t) = -\sin(2\pi x)\sin(2\pi y)\sin^2(\pi z)cos(2\pi\frac{t}{\mathcal{T}}). \end{cases} \tag{3.20}$$

In our example, the numerical value for the time period is the following:

- $\mathcal{T} = 6$.

Therefore, the bubble is transported until time $t = \mathcal{T}/4$. Then the direction of the flow is reversed and the sphere comes back to its initial position at time $t = \mathcal{T}/2$. If the simulation were done well – that is if the numerical scheme were ideal – then the initial and final positions of the sphere should coincide. As a consequence, it is a good way to assess the quality of an advection scheme. A similar test case also exists for 2D, but we choose to focus on the more computationally intensive 3D geometry.

Many use the test case to challenge the quality of their advection scheme, like by Mekkas [104] or Menier [105].

3.4.1 Advection scheme

As a reminder, we use Cartesian meshes because their multi-level implementation is the easiest. As a consequence, a finite differences approach is the most intuitive to discretise the space and time. Because it will simplify the writings greatly, in the following subsubsection, we will write discretisation schemes for the one-dimensional problem

$$\partial_t Y + u\partial_x Y = 0 \tag{3.21}$$

where u is positive, but not necessarily constant in time nor uniform across space. The case of u negative can be obtained by symmetry. We explain our computation of problems with a dimension strictly greater than 1 in Section 3.4.1.

Equation (3.21) is an advection equation and it respects the principle of maximum. In what follows, we will present only three numerical schemes for this equation:

- the upwind scheme, the most basic scheme,
- WENO, as an example of a widely used high-order scheme,
- the limited downwind scheme, which we will implement.

Of course, many more very important schemes exist (see [120] for instance).

In what follows, we will discretise the time with exponent n and a time step Δt, such that $\forall n \in \mathbb{N}, t^n = n\Delta t$. Similarly, we will discretise the space with index j and a grid space Δx, such that $\forall j \in [\![0, n_{nodes} - 1]\!], x_j = j\Delta x$ if $x \in [0, (n_{nodes} - 1)\Delta x]$. We locate the colour function Y as well as the velocity u at the centre of cells. Thus we will note $Y_j^n = Y(x_j, t^n)$.

Upwind

In this subsection we will present the most basic working scheme which is stable and satisfies the maximum principle; the upwind scheme (see for instance [136]). As far as time discretisation is concerned, the discretisation of $\partial_t Y$ as the non-centred expression $\frac{Y^{n+1}-Y^n}{\Delta t}$ is very usual. Depending on what follows for the space discretisation, the time discretisation can be explicit or implicit. In what follows we choose the explicit form:

$$\frac{Y^{n+1} - Y^n}{\Delta t} + (u\partial_x Y)^n = 0. \tag{3.22}$$

As presented for instance in [3], the upwind way to discretise $\partial_x Y$ is as follows:

$$\partial_x Y \approx \frac{Y_j - Y_{j-1}}{\Delta x}. \tag{3.23}$$

since we took $u \geq 0$. The upwind scheme is L^2 and L^∞ stable as long as the CFL condition is respected:

$$|u|\Delta t \leq \Delta x. \tag{3.24}$$

If u is not uniform, then the CFL condition reads as follows:

$$\max(|u|)\Delta t \leq \Delta x \tag{3.25}$$

where $\max(|u|)$ is the maximum of $|u|$ on the computational domain.

The induced error of the upwind scheme is in $\mathcal{O}(\Delta t + \Delta x)$. Unfortunately, the scheme is very diffusive with a diffusivity coefficient of $\frac{|u|}{2}(\Delta x - |u|\Delta t)$. The diffusion accumulates with time, which makes the scheme not fit for the study of the advection of bubbles defined by a sharp interface.

Higher order schemes, WENO

The upwind scheme probably is the most simple scheme existing for the advection. A lot of progress has been made on the topic, in particular to improve the accuracy. One main way to achieve a higher order of accuracy is to use larger stencils. For instance, Vila shows in [143] how to construct a 7-point second-order accurate (TVD) scheme from any 3-point first-order accurate (TVD) scheme. As another example, we present here one of the most advanced high order schemes: the Weighted Essentially Non-Oscillatory Scheme (WENO).

WENO was introduced by Liu, Osher and Chan in [95]. It is an improvement of the ENO (Essentially Non-Oscillatory) schemes [79, 78, 125, 126]. ENO and WENO consist in interpolating the function we want to discretise by a polynomial of degree r. The schemes introduce a numerical dissipation in the discontinuity areas to avoid oscillations and still keep a high order of accuracy in the other areas.

In the case of ENO, the scheme uses the average value of the function on stencils of $r - 1$ cells around each considered cell. One of the specificities of the algorithm is that it picks the “smoothest” stencil – on which the solution is the least sharp –, depending on the region. Because r can be large (for instance equal to 5), this method is said to be a high order scheme. In the case of WENO, it is not one stencil that is picked up, but rather we use a specific convex combination of all the considered stencils. Each candidate stencil is taken into account with a relative “weight” proportional to the smoothness of the discretised function on the stencil. As a consequence, this increases the order of accuracy by one order. One can prove that the resulting order of accuracy should be at least $r + 1$ in shock-free areas and r around the cells where there is a discontinuity. In practice, the order of accuracy is even better and it is measured to be closer to $r + 2$.

As detailed in [93], let us focus on the expression of WENO in a one-dimensional case, with an order of 3:

$$\frac{\partial Y}{\partial t} + \frac{\partial f(Y)}{\partial x} = 0 \tag{3.26}$$

where for instance $f(Y) = uY$. The characteristic flux $f(Y)$ can be decomposed in a positive and a negative part:

$$f(Y) = f^+(Y) + f^-(Y) \tag{3.27}$$

with

$$\frac{\partial f^+}{\partial Y} \geq 0, \ \frac{\partial f^-}{\partial Y} \leq 0. \tag{3.28}$$

By focusing on a third-order WENO scheme, we have for each cell i three possibilities of stencils:

$$\left\{ \begin{array}{lcl} S^{(1)} & = & \{i-2, i-1, i\}, \\ S^{(2)} & = & \{i-1, i, i+1\}, \\ S^{(3)} & = & \{i, i+1, i+2\}. \end{array} \right. \tag{3.29}$$

On these three support stencils, we define three associated fluxes; in the case of the positive part $f^+_{i+\frac{1}{2}}$, we get:

$$\left\{ \begin{array}{lcl} f^{(1)+}_{i+\frac{1}{2}} & = & \frac{11}{6} f^+_i - \frac{7}{6} f^+_{i-1} + \frac{2}{6} f^+_{i-2}, \\ \\ f^{(2)+}_{i+\frac{1}{2}} & = & \frac{2}{6} f^+_{i+1} + \frac{5}{6} f^+_i - \frac{1}{6} f^+_{i-1}, \\ \\ f^{(3)+}_{i+\frac{1}{2}} & = & -\frac{1}{6} f^+_{i+2} + \frac{5}{6} f^+_{i+1} + \frac{2}{6} f^+_i. \end{array} \right. \tag{3.30}$$

Now we recontruct the convex combination of $f^+_{i+\frac{1}{2}}$:

$$f^+_{i+\frac{1}{2}} = \sum_{k=1}^{3} w^{(k)} f^{(k)+}_{i+\frac{1}{2}} \tag{3.31}$$

with

$$w^{(k)} = \frac{\sigma^{(k)}}{\sigma^{(1)} + \sigma^{(2)} + \sigma^{(3)}}, \tag{3.32}$$

$$\sigma^{(k)} = \frac{W^{(k)}}{(\epsilon + IS^{(k)})^p}. \tag{3.33}$$

The $W^{(k)}$ are the correct weights: $W^{(1)} = 1/10$, $W^{(2)} = 6/10$ and $W^{(3)} = 3/10$. The real numbers $IS^{(k)}$ are the "smoothness indicators" to decrease the influence of discontinuities:

$$\left\{\begin{array}{lcl} IS^{(1)} & = & \frac{13}{2}(f^+_{i-2} - 2f^+_{i-1} + f^+_i)^2 + \frac{1}{4}(f^+_{i-2} - 4f^+_{i-1} + 3f^+_i)^2, \\ IS^{(2)} & = & \frac{13}{2}(f^+_{i-1} - 2f^+_i + f^+_{i+1})^2 + \frac{1}{4}(f^+_{i-1} - f^+_{i+1})^2, \\ IS^{(3)} & = & \frac{13}{2}(f^+_i - 2f^+_{i+1} + f^+_{i+2})^2 + \frac{1}{4}(3f^+_i - 4f^+_{i+1} + f^+_{i+2})^2. \end{array}\right. \tag{3.34}$$

Finally, we choose $p = 2$ and $\epsilon = 10^{-6}$ to avoid divisions by 0.

As an other example of use, the WENO scheme was used in its fifth order variant in [36], to model (potentially mixing) multi-phase flows. Irrespective of the order, we have anyway an accumulation of the diffusion error as time passes. As for the upwind scheme, this is in particular true for step-shaped functions as the colour function of a bubble. This is what leads us to the following scheme.

Limited downwind scheme

To quote [51], Després and Lagoutière noted that "*a numerical scheme should ideally respect two points which may be viewed as incompatible: a) a numerical method must have enough dissipation in order to be stable and to capture discontinuous solutions when applied to non-linear hyperbolic problems; b) on the other hand it is important to use a numerical method with as low numerical dissipation as possible, at least one order of magnitude below the real physical dissipation*".

This is how they came to designing a numerical scheme that takes the downwind value as much as possible. Let us consider the 1D problem $\partial_t Y + u\partial_x Y = 0$ where u is uniform on the domain, constant in time and furthermore positive. Let us discretise the space with index j and the time with exponent n as follows in the finite-volume-like discretisation:

$$Y_j^{n+1} = Y_j^n - u\frac{\Delta t}{\Delta x}(Y^n_{j+\frac{1}{2}} - Y^n_{j-\frac{1}{2}}) \tag{3.35}$$

with Δt the time step and Δx the space step (cell size). We use a simplified notation as follows:

$$Y_j^\star = Y_j - u\frac{\Delta t}{\Delta x}(Y_{j+\frac{1}{2}} - Y_{j-\frac{1}{2}}). \tag{3.36}$$

The velocity u being taken positive, let us define m_j and M_j for all j as follows:

$$\left\{\begin{array}{l} m_j = \min(Y_j, Y_{j-1}), \\ M_j = \max(Y_j, Y_{j-1}). \end{array}\right. \tag{3.37}$$

Let us then define b_j and B_j for all j as follows:

$$\begin{cases} b_j = \max(m_{j+1}, M_j + \dfrac{\Delta x}{u\Delta t}(Y_j - M_j), \\ \\ B_j = \min(M_{j+1}, m_j + \dfrac{\Delta x}{u\Delta t}(Y_j - m_j). \end{cases} \tag{3.38}$$

As presented in Lagoutière's thesis [94], the downwind limiting scheme writes as follows:

$$Y_{j+\frac{1}{2}} = \begin{cases} b_j & if \quad Y_{j+1} < b_j, \\ Y_{j+1} & if \quad b_j \leq Y_{j+1} \leq B_j, \\ B_j & if \quad B_j < Y_{j+1}. \end{cases} \tag{3.39}$$

As shown in [51], this limited downwind scheme is equivalent to the so called Ultra-Bee limiter [120, 136], but here proposed in a constructive way. The scheme extends for negative values of the velocity u by symmetry. Moreover, we can also extend the scheme for a non-uniform and non-constant u: in this case, for the computation of $Y_{j+\frac{1}{2}}$, we replace the value of u by the local value u_j. This extension of the scheme has been used by many, like by Bernard Champmartin and De Vuyst for the simulation of free-surface flows for instance [24]. Kokh and Lagoutière show good results when applying it for a 5-equation model [91] and Dellacherie achieves a successful simulation of DLMN in [46].

Després, Lagoutière and coworkers made an extension of the limited downwind scheme called “Vofire” [52]. It keeps the same powerful properties and is extended for unstructured meshes, that is not necessarily Cartesian. It would be interesting to try Vofire out on our Cartesian grids.

Multidimensional scheme

The schemes we presented in the previous sections were expressed for one-dimensional spaces. We choose to do the simulation of the 3D Kothe-Rider test with a directional splitting of the one-dimensional limited downwind scheme. This means that we do not solve $\partial_t Y + \mathbf{u} \cdot \nabla Y = 0$ directly. Instead, we rather discretise the following equations one after another:

$$\partial_t Y + u_x \partial_x Y = 0, \tag{3.40}$$

$$\partial_t Y + u_y \partial_y Y = 0, \tag{3.41}$$

$$\partial_t Y + u_z \partial_z Y = 0. \tag{3.42}$$

We understand that the "composition" of these three equations – which we might loosely write as (3.42) $\circ$ (3.41) $\circ$ (3.40) – is "equivalent" to the 3D equation (3.19) we want to solve.

Let us semi-discretise this set of equations in time. Let Δt be the time step. We compute Δt beforehand – for instance with a CFL condition based on $\max(\mathbf{u})$ – and we keep it the same for the three directions x, y and z. This is independent from the choice of an implicit or explicit scheme. Let $Y^\star$ and $Y^{\star\star}$ be intermediary results:

$$\frac{Y^\star - Y^n}{\Delta t} + u_x \partial_x Y = 0, \tag{3.43}$$

$$\frac{Y^{\star\star} - Y^\star}{\Delta t} + u_y \partial_y Y = 0, \tag{3.44}$$

$$\frac{Y^{n+1} - Y^{\star\star}}{\Delta t} + u_z \partial_z Y = 0. \tag{3.45}$$

The composition (3.45) $\circ$ (3.44) $\circ$ (3.43) brings an error of a magnitude of $\mathcal{O}(\Delta t)$. This is however fortunately not such an issue, since each transport scheme will already bring an error in $\mathcal{O}(\Delta t)$. So the directional splitting only marginally affects the order of convergence and the precision.

3.4.2 Multilevel procedure

We now detail the procedure used to solve equations on two AMR levels. Let Ω be the computational domain. We divide it into two areas which depend on time:

1. $\Omega_\square(t)$, the area of interest at time t, which is covered by patches,
2. $\Omega_\bigcirc(t)$, the complementary of $\Omega_\square(t)$ at time t in Ω.

We get the following decomposition:

$$\forall t \in \mathbb{R}_+, \Omega = \Omega_\square(t) \cup \Omega_\bigcirc(t). \tag{3.46}$$

We discretise the computational domain with a coarse mesh called Ω^H. We refer to this mesh as the "level 0" and it does not evolve with time. The mesh Ω^H is subdivided into two complementary domains; $\Omega^H_\square(t)$ and $\Omega^H_\bigcirc(t)$. Since the area $\Omega_\square(t)$ is the region of interest, it is covered by the "level 1" fine mesh $\Omega^h_\square(t)$.

Let $iter_t$ be an integer noting a time iteration of computation and let $t = t(iter_t)$ be the associated time, strictly increasing with $iter_t$. We note

$Y(\Omega^H, iter_t)$ the vector of values of one or many variables Y on the mesh Ω^H at time t. Similarly, Y_0^H is a given value for this or these variables Y. We will use Y_0^H as initial conditions. Let $\Delta t(iter_t) \in \mathbb{R}$ be a time step depending on $t(iter_t)$. Let us define the following discrete iterative problem:

$$\begin{cases} Y(\Omega^H, iter_t = 0) = Y_0^H, \\ \forall iter_t \in \mathbb{N}, Y(\Omega^H, iter_t + 1) = \mathbb{O}\left(Y(\Omega^H, iter_t), \Delta t(iter_t)\right). \end{cases} \tag{3.47}$$

We want to compute this problem on two levels. That means we will deal with Y discretised on Ω^H, $\Omega^h_\square(t)$, $\Omega^H_\square(t)$ and $\Omega^H_\bigcirc(t)$. For the sake of simplicity, we assume that the refinement coefficient r is the same in all directions x, y and z.

Let us use Figure 3.21 to clarify some notations. The dotted line $\Sigma(0)$ represents an interface between bubble and liquid at time iteration 0. The full line $\Sigma(1)$ represents the interface at time iteration 1. They are included in the larger computational domain Ω, not annotated on the figure. At each time step, around the interface, we define a square area of interest (respectively $\Omega_\square(0)$ and $\Omega_\square(1)$) represented by a brown square. We can differentiate three subareas: we represent $\Omega_\square(0) \setminus \Omega_\square(1)$ in white, $\Omega_\square(0) \cap \Omega_\square(1)$ in pink and $\Omega_\square(1) \setminus \Omega_\square(0)$ in red.

Let us define some operators.

- Operator $\mathbb{F}$ determines the initial conditions for $Y(\mathbf{x}, t = 0)$, for all $\mathbf{x} \in \Omega$, following an analytic formula. In particular it gives the value on the coarse mesh ($\mathbf{x} \in \Omega^H$) and on the fine mesh ($\mathbf{x} \in \Omega^h_\square(0)$). In other words, it gives $Y(\Omega^H, 0)$ and $Y(\Omega^h_\square(0), 0)$.

- Operator $\mathbb{A}$ takes a coarsely discretised $Y(\Omega^H, iter_t)$ as an argument and locates the area of interest. It then computes the localisation of the AMR patches and thus of the fine level $\Omega^h_\square(iter_t)$. If $\mathbb{A}$ is well designed, then the intersection between patch coverings at successive time iterations cover the area of interest along its evolution. This is what we represented on Figure 3.21: $\Sigma(1) \subset \Omega_\square(0) \cap \Omega_\square(1)$.

- Operator $\mathbb{O}$ takes $Y(\Omega^H, iter_t)$ and Δt as arguments and computes $Y^*(\Omega^H, iter_t + 1)$. The star as exponent * indicates that the result is intermediary.

- Operator $\mathbb{O}^r$ takes $Y(\Omega^h_\square, iter_t)$ and Δt as arguments and computes $Y^*(\Omega^h, iter_t + 1)$. In fact, as the scripture suggests, $\mathbb{O}^r$ is a composition

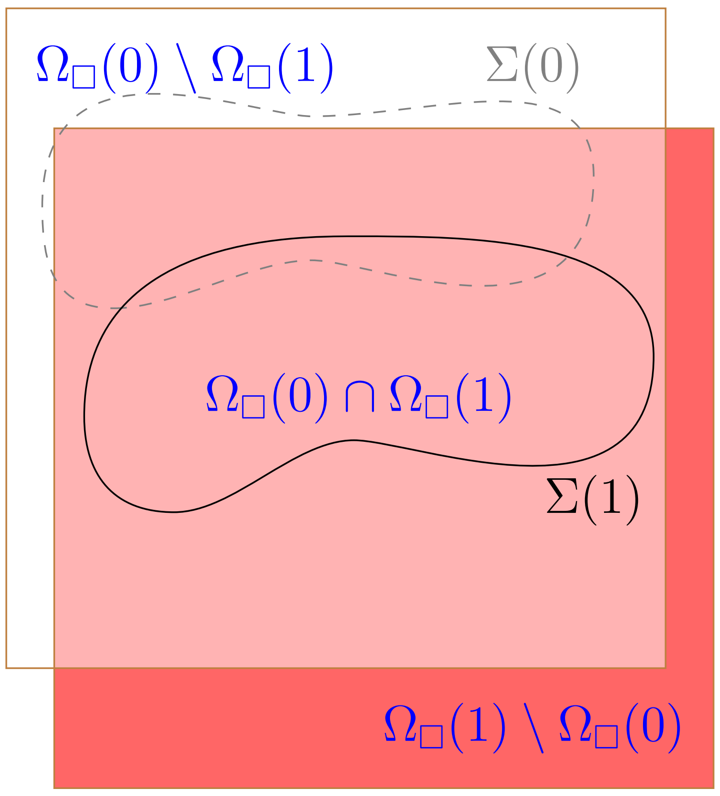

$\Omega_\Box(0) \setminus \Omega_\Box(1)$
$\Sigma(0)$
$\Omega_\Box(0) \cap \Omega_\Box(1)$
$\Sigma(1)$
$\Omega_\Box(1) \setminus \Omega_\Box(0)$

Figure 3.21 – Intersection of $\Omega_\Box(iter_t = 0)$ and $\Omega_\Box(iter_t = 1)$

of r times $\mathbb{O}$ with a time step of $\frac{\Delta t}{r}$:

$$\begin{aligned} &\mathbb{O}^r \left(Y(\Omega^h_\square, iter_t), \Delta t\right) \\ &\qquad = \bigcirc_{i=1\ldots r} \mathbb{O}\left(\star, \frac{\Delta t}{r}\right) \left(Y(\Omega^h_\square, iter_t)\right), \\ &\qquad = \mathbb{O}\left(\star, \frac{\Delta t}{r}\right) \mathbb{O}\left(\star, \frac{\Delta t}{r}\right) \ldots\ \mathbb{O}\left(Y(\Omega^h_\square, iter_t), \frac{\Delta t}{r}\right). \end{aligned} \tag{3.48}$$

Again, the star as exponent * indicates that the result is intermediary.

- Operator $\mathbb{1}$ expresses the copy of data on a subdomain, or in other words the multiplication by an indicator function. For instance, we can write the following definition:

$$\mathbb{1}\left(Y^*(\Omega^H, iter_t + 1), \Omega^H_\bigcirc\right) = \mathbb{1}_{\Omega^H_\bigcirc}\left(Y^*(\Omega^H, iter_t + 1)\right). \tag{3.49}$$

- Operator $\mathbb{R}^H$ is the restriction of a fine discretisation $Y^h_\square$ onto the coarse grid $\Omega^H_\square$.
- Operator $\mathbb{I}^h$ is the interpolation (linear, quadratic, trigonometric...) of a coarse discretisation $Y^H_\square$ onto the fine grid $\Omega^h_\square$.

Figure 3.22 shows a logical flowchart to compute the initial conditions at $iter_t = 0$, $t = 0$, as well as to compute the first step at $iter_t = 1$, $t = \Delta t$. The green boxes indicate the data we save and the blue boxes (often with variables superscripted with an asterisk) indicate intermediary data that we do not save.

It is important to notice that we made an important assumption for the computation of the "coarse time step" Δt and the "fine time step" $\frac{\Delta t}{r}$. We determine the coarse step Δt with a CFL condition which takes a velocity as an argument; in this case the velocity discretised at the coarse level, noted $\mathbf{u}^H$:

$$\max(||\mathbf{u}^H||)\Delta t \leq \Delta x. \tag{3.50}$$

If the variations of $\mathbf{u}$ are large on short distances, especially in the region of interest $\Omega_\square$, then it is possible that we get the following problematic situation:

$$\max(||\mathbf{u}^H||) \leq \max(||\mathbf{u}^h||). \tag{3.51}$$

In this case, we would need a smaller time step on the fine grid $\Omega^h_\square$ for stability purposes: $\frac{\Delta t}{r}$ would already be too large, knowing that the grid size of $\Omega^h_\square$ is

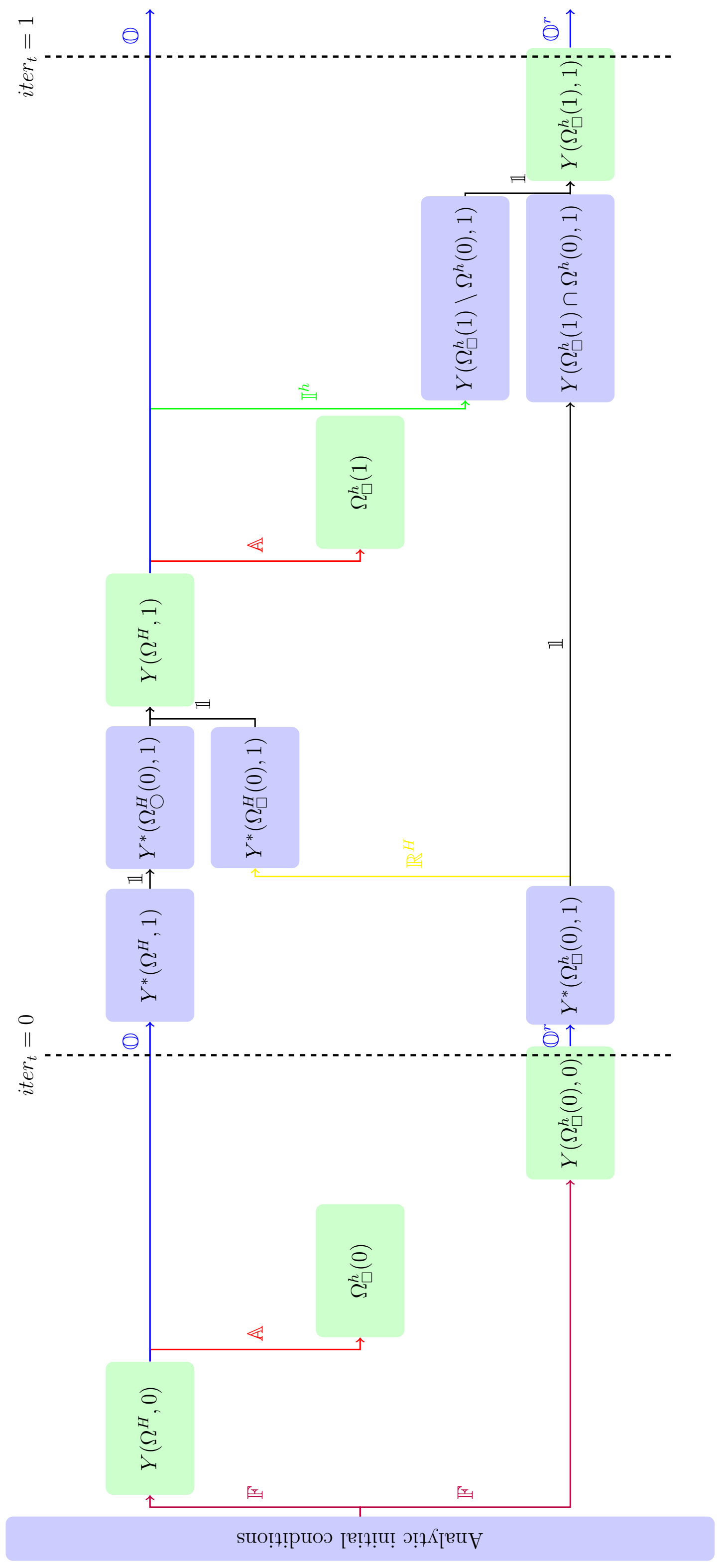


Figure 3.22 – Flowchart for AMR on two levels

equal to $\frac{\Delta x}{r}$. As a consequence, we choose the following value for Δt:

$$\Delta t = k \frac{\Delta x}{\max(||\mathbf{u}^H||, \epsilon_{CFL})} \tag{3.52}$$

where $k = 0.7$ is an extra margin coefficient for CFL stability and ϵ_{CFL} is a small positive real number protecting us from the case $\mathbf{u}^H = 0$. Additionally, we make the assumption that $\frac{\Delta t}{r}$ is now small enough (compared to the spatial variation of $\mathbf{u}^h$) for a good computation on the fine level. In our work, we did not find counter-example situations.

3.4.3 Results in CPU time speed-up

As explained before, we want to estimate $su_{||}$ for the different clustering algorithms. We realise tests on the Berger-Rigoutsos and the $n_{min} - n_{max}$ algorithms definition of speed-up because, as shown in Section 3.3.4, these two algorithms are the most promising ones. Our results ([129] and later work) show that the speed-up is – as expected – a function of the algorithm, the number of threads and the problem we try to represent and cover with patches.

1.7 billion cells The first Kothe-Rider test we make is set on a 3D grid of $200 \times 200 \times 200$ cells, with a refinement coefficient of 6 in every direction, on one level only. As a consequence, were the grid refined everywhere, there would be $(200 \times 6)^3 = 1728000000$ fine cells, so more than 1.7 billion. As we are using the limited downwind scheme on a Y field equal to either 0 or 1, we can conclude that the results we got with local refinement (AMR) are strictly equal to those we would have had with a fine grid everywhere. This is why we refer to this problem as strictly equivalent to more than 1.7 billion cells. The results themselves of the simulation do not depend on the clustering algorithm and are shown on Figure 3.23. An online video is also available at `https://youtu.be/Ixgge4h6eF8`.

The input parameters of each algorithm are given in Table 3.3. At the time step 0, these parameters result in creating 7113 patches for Berger-Rigoutsos, 172 for Livne and 172 as well for $n_{min} - n_{max}$. The reason that Berger-Rigoutsos creates so many patches is probably that the covering followed the interface very closely. This is understandable since the algorithm is allowed to create very small patches, even to the size of $1 \times 1 \times 1$.

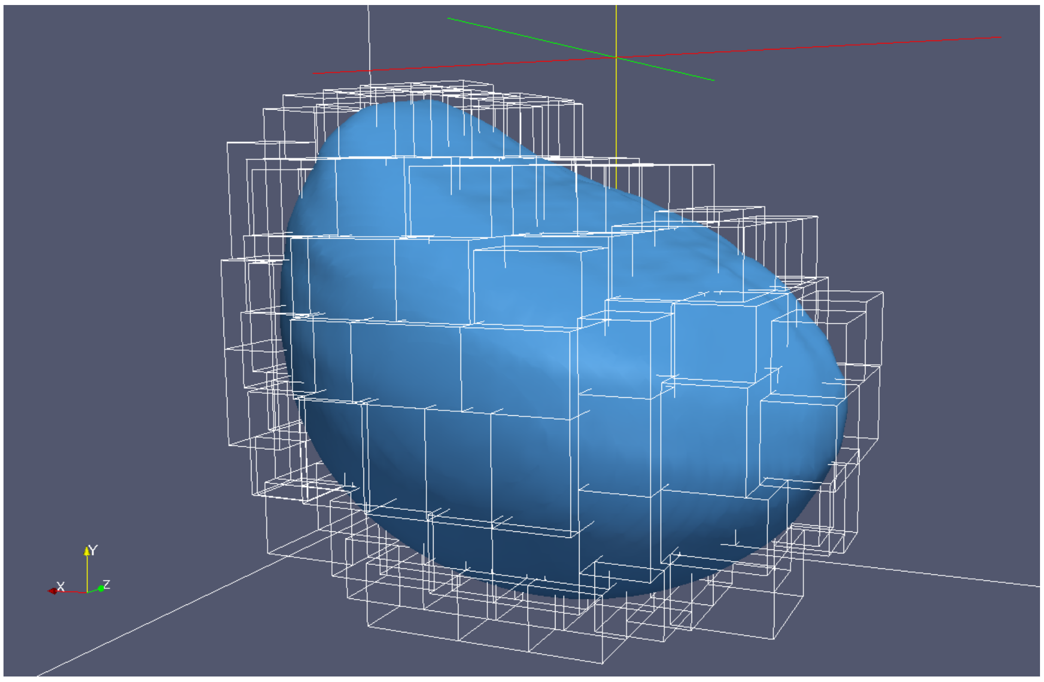
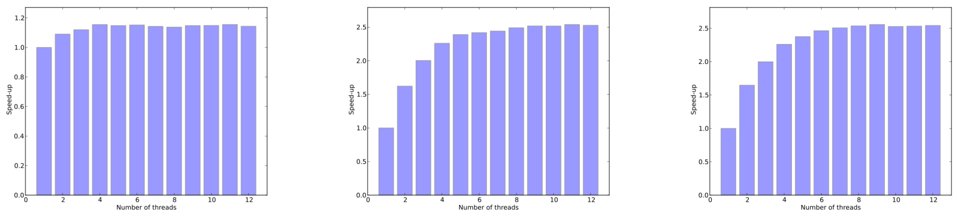


Figure 3.23 – Advected bubble visualised with its 3D patches

	η_{min}	n_{min}	n_{max}	N_{max}
Berger-Rigoutsos	0.8			
Livne	0.8	7		12^2
$n_{min} - n_{max}$	0.8	7	12	

Table 3.3 – Input parameters for our 1.7 billion cells Kothe-Rider test



(a) Berger-Rigoutsos (b) Livne (c) $n_{min} - n_{max}$

Figure 3.24 – Speedup for the 1.7 billion cells test, for all algorithms

Figures 3.24a, 3.24b and 3.24c show the speed-ups for the three algorithms, for a machine that can activate from 1 to 12 simultaneous threads. As Livne and $n_{min} - n_{max}$ approximately give similar results, we choose to focus on the comparison between Berger-Rigoutsos and $n_{min} - n_{max}$.

Figure 3.25 shows the results of computation time and speed-up of both Berger-Rigoutsos (in red) and $n_{min} - n_{max}$ (in green). The first thing we notice is that the computation in a one-thread set – that-is-to-say in sequential set – is much longer for Berger-Rigoutsos than for $n_{min} - n_{max}$: 10.3 hours compared to 4.5 hours. This is because the workstation used a lot of memory and had to "swap", that is to say it used the ROM memory instead of the faster RAM. It is hard to explain the reason; maybe this could be explained by the fact that the Berger-Rigoutsos algorithm produced many more patches, as we saw earlier. We will see this aspect in more detail in the next section, Section 3.4.4. Second, we see that the speed-up of the Berger-Rigoutsos algorithm stays approximately constant around 1. In other words, it barely benefits from our parallelisation strategy. On the contrary, the $n_{min} - n_{max}$ clearly benefits from our parallelisation strategy, as expected. The speed-up with the 12 threads of our workstation is approximately 2.5, so that brings the workload from 4.5 h to 1.8 h. It would be fair to remark that a speed-up of 2.5 for 12 CPUs is far from ideal. But as a first attempt, it is promising for future developments.

4 billion cells The second Kothe-Rider test we make is set on a 3D grid of $200 \times 200 \times 200$ cells also, but this time with a refinement coefficient of 8 in every direction, still on one level only. As a consequence, were the grid refined everywhere, there would be $(200 \times 8)^3 = 4096000000$ fine cells, so more than 4 billion. Again, as we are using the limited downwind scheme on a Y field equal to either 0 or 1, we can still say that the results we got with local refinement (AMR) are strictly equal to those we would have had

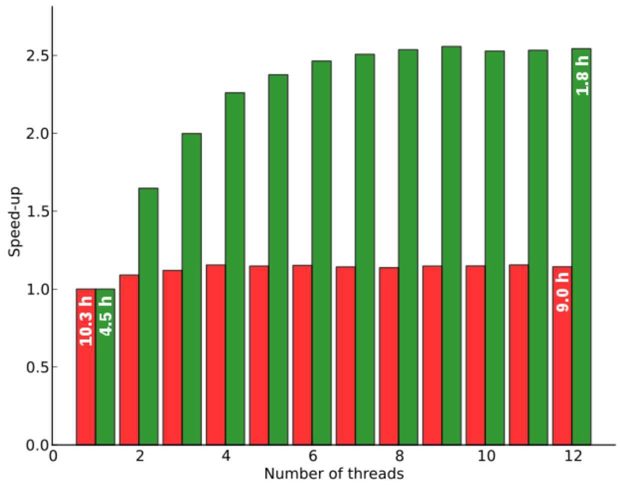


Figure 3.25 – Speed-up for the 1.7 billion cells test

with a fine grid everywhere of 4 billion cells. The results themselves of the simulation are of course similar to what we had before, but more precise.

In other words, when we compare to the 1.7 billion cells case, what we did is dramatically increase the number of cells on the first level of refinement. According to our strategy this is the level we make parallel. So we increase the relative computation subject to parallelisation. We thus expect better results for speed-up.

This time, it is not possible to compute speed-up results for the Berger-Rigoutsos algorithm. Again, it is difficult to assert a convincing reason; perhaps that the calculation is too heavy because of the number of patches, so the computation did not even finish. On the contrary, the computation with patches defined with $n_{min} - n_{max}$ did give results. So we can measure speed-ups, which we show in Figure 3.26.

We can say that the results are even better than with the 1.7 billion test case. Indeed, the 4 billion cells computation takes 28.9 h in a sequential set, but only 6.3 h with 12 threads activated. This leads to a speed-up $su_{\|}$ of approximately 4.6. This is encouraging for future developments.

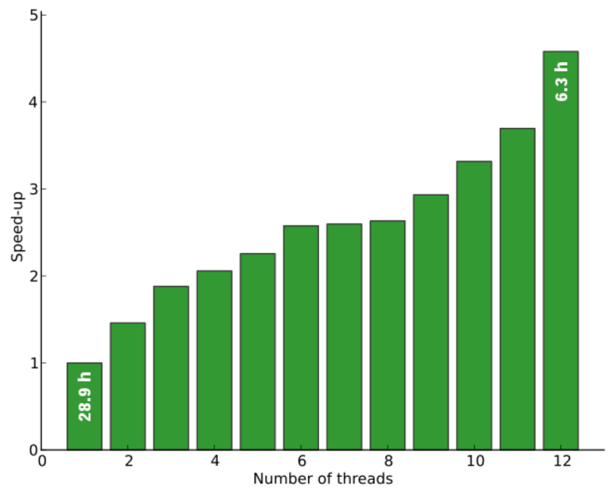


Figure 3.26 – Speedup for the 4 billion cells test

3.4.4 Results in memory consumption

In addition to speed improvements with parallelisation, using AMR also is a way to save on the memory consumption of a computation. The computation is indeed more precise on the regions of interest, so we save the need to store the less relevant data of cells out of the regions of interest. This is because the memory storage is proportional to the number of cells and variables of a problem. The improvements in terms of memory due to AMR are thoroughly explained by Delage-Santacreu et al. in [54].

In our case, we have not measured those improvements quantitatively yet. However we have noticed that we were able to post-process and visualise information-rich simulations like our 4 billion cells Kothe-Rider test, on a good but only standard workstation.

Memory challenges will be all the more important when we try to avoid the swapping if it occurs. It will also be crucial when we will implement distributed memory parallelisation, for instance with a hybrid configuration OpenMP together with MPI.

Chapter 4

Elliptic equations and Abstract Bubble Vibration model

4.1 Resolution of elliptic equations on patch-based AMR with LDC

Now that we have determined with what algorithm we will generate patches, we want to make our model of bubbles more complete with different kinds of equations. In particular, we will not restrict ourselves with just hyperbolic equations like $\partial_t Y + \mathbf{u} \cdot \nabla Y = 0$. We want to also be able to use elliptic equations for instance.

Elliptic partial differential equations – like $-\Delta\phi = s$ for example – present the challenge of making all points in space very linked with all the other points in space. That is, a small variation at one point will have an influence on all other points. As a consequence, our patch-based AMR strategy is not evident any more. In the case of hyperbolic systems, we used to consider each patch as a nearly independent problem. As this is not the case any more, we have to adapt our strategy.

Solving an elliptic problem with a patch-based locally refined AMR grid requires specific techniques. One method could be the Fast Adaptive Composite Grid (FAC) method if we consider the multi-level grids as a single composite grid [102, 103, 101]. Another method is the Flux Interface Correction (FIC) method introduced by Angot et al. in [8] and adapted to conservative equations. Nonetheless, in our work, we use a special case of the Local Defect Correction (LDC) method introduced in [74]. Ferket and Reusken show in

[64] that the two methods give identical iterative results. The explanation of LDC that follows is largely based upon the clear explanations given in [64].

4.1.1 Global coarse grid problem

Dirichlet boundary conditions problem Let Ω be the (closed) space of our problem, with a boundary $\partial\Omega$. Let $\mathcal{L}$ be a scalar linear elliptic second-order differential operator. We want to solve the following Dirichlet boundary conditions problem:

$$\begin{cases} \mathcal{L}\phi = s & \text{in } \Omega, \\ \phi = g & \text{on } \partial\Omega. \end{cases} \tag{4.1}$$

We assume that Equation (4.1) admits a solution ϕ_{exact} which presents large variations in a limited volume $\Omega_\square \subset \Omega$ and that the solution behaves very smoothly in the region outside of $\Omega_\square$, which we note $\Omega_\bigcirc$. We have thus $\Omega = \Omega_\square \cup \Omega_\bigcirc$. The boundary $\partial\Omega_\square$ can be divided in two parts: one part that coincides with $\partial\Omega$ and one part on the inside of Ω. In this section, we will call the latter part the exchange area $\Gamma = \partial\Omega_\square \setminus \partial\Omega$ between $\Omega_\square$ and Ω. We represent these notations on Figure 4.1. In particular, $\Omega_\square$ is drawn with a blue rectangle and Γ is drawn with a full blue line. We name $\Omega_\bigcirc \cup \Gamma$ the area of Ω which is covered by $\Omega_\bigcirc$ or by Γ.

Numerical discretisation We want to use a numerical approximation based upon a finite differences discretisation since we use Cartesian meshes. We use two uniform grids: one global (level 0) with a cell size H which covers Ω, and one local (level 1) with a cell size h which covers only $\Omega_\square$. We will name them respectively Ω^H and $\Omega^h_\square$. As the solution to (4.1) evolves very rapidly (in space) on the region of interest, we want $h < H$. Let the integer $r = \frac{H}{h}$ be the refinement coefficient; r is often taken as smaller than 10. Figure 4.2 shows Ω^H in black (and blue in the $\Omega_\square$ region) and shows $\Omega^h_\square$ in blue, as well as H and h in orange, with $r = 2$ in every direction of space. We can also define a coarse exchange area grid Γ^H: it is the set of coarse faces and nodes located on Γ. Similarly, we define the fine exchange area grid Γ^h. We name $\Omega^H_\bigcirc \cup \Gamma^H$ the area of Ω^H which is either covered by $\Omega_\bigcirc$ or by Γ. We will also consider the composite grid Ω^c, which is the union of the global coarse grid Ω^H and the local fine grid $\Omega^h_\square$. The nodes of Ω^c are represented by green balls on Figure 4.3.

Given these discretisations, in the following demonstrations we choose to place variables at the nodes of the grids. This brings us a discretised coarse

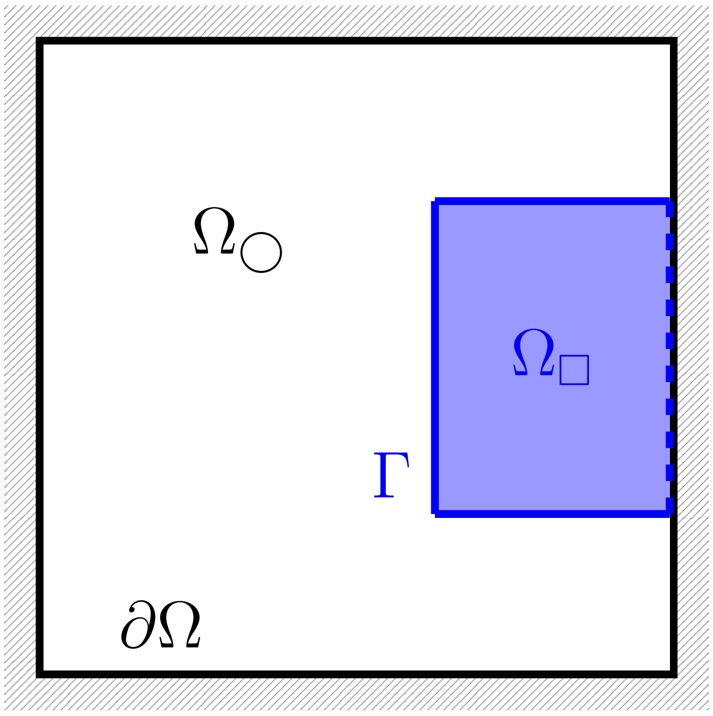


Figure 4.1 – Nomenclature for areas in $\Omega = \Omega_{\square} \cup \Omega_{\bigcirc}$

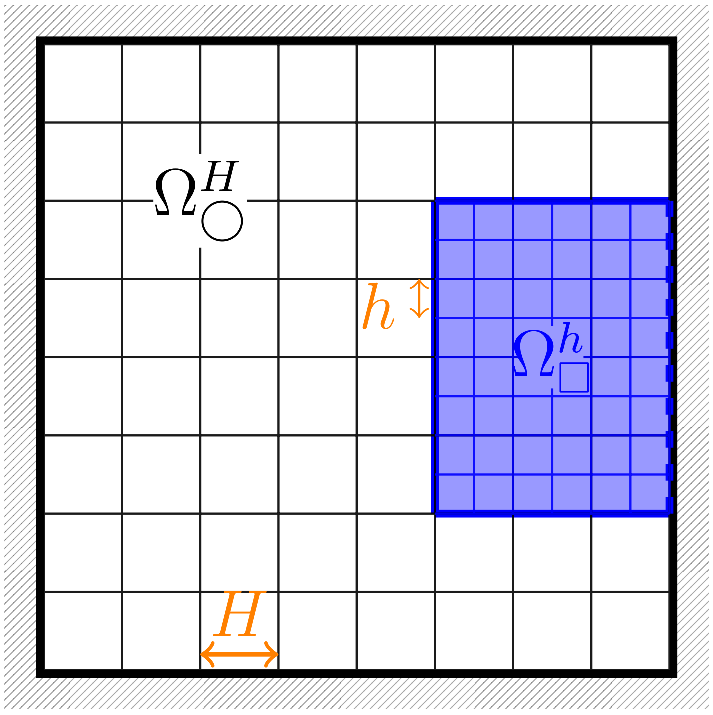


Figure 4.2 – Nomenclature for grids in $\Omega = \Omega_{\square} \cup \Omega_{\bigcirc}$

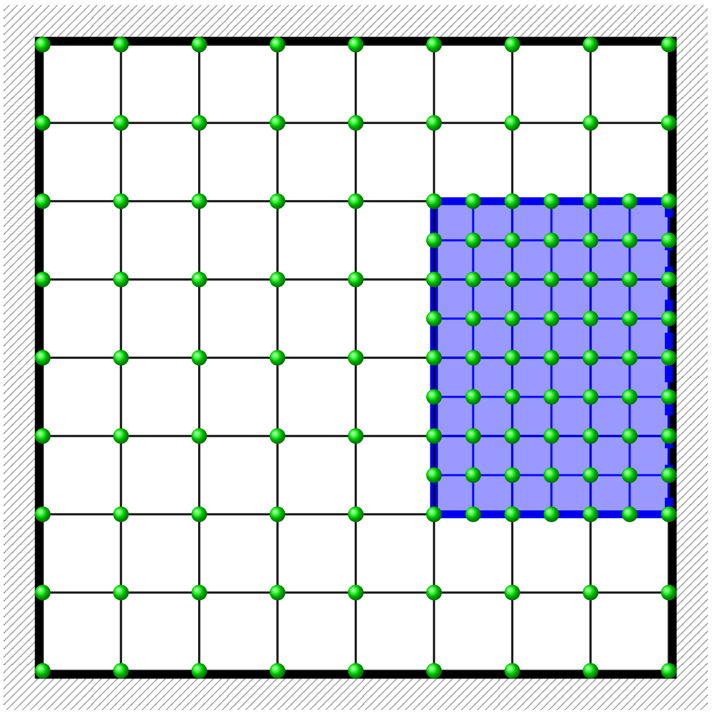


Figure 4.3 – Nodes of Ω^c

grid problem in a matrix form:

$$\mathcal{L}^H \phi^H = s^H \text{ on } \Omega^H. \tag{4.2}$$

The Dirichlet boundary conditions from the physical walls $\partial\Omega$ are incorporated in the source term s^H.

As a remark, we could have chosen $h \ll H$. But for robustness and computational cost reasons, we prefer to use $h = \frac{H}{2}$.

4.1.2 Enrichment from local fine grid

When there is only one patch on which the grid H will be refined, the LDC algorithm is easiest to understand. It is an iterative algorithm initialized at $iter_{LDC} = 0$ and iterated for $iter_{LDC} \in \mathbb{N}^*$.

Interpolation and restriction Let v^H be a grid function defined on the nodes of Ω^H. It could be ϕ^H for instance, but not necessarily. The prolongation of v^H on Γ^h is the interpolation of v^H on the fine grid nodes of

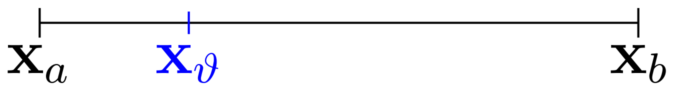


Figure 4.4 – Localisation of grid nodes, $r = 4$ and $\vartheta = \frac{1}{4}$

Γ. In practice this operator $\mathbb{I}^h$ will be linear or quadratic, or in rarer cases trigonometric [10]. For instance, let $\mathbf{x}_a$ and $\mathbf{x}_b$ be two nodes of Γ^H distant of only H. They are associated with the values $v^H(\mathbf{x}_a)$ and $v^H(\mathbf{x}_b)$. Now let ϑ be a real number between 0 and 1. Let us assume $\mathbf{x}_\vartheta = \vartheta \mathbf{x}_a + (1-\vartheta)\mathbf{x}_b \in \Gamma^h$. Figure 4.4 shows an example with $r = 4$ and $\vartheta = \frac{1}{4}$. We can define the value of $v^h(\mathbf{x}_\vartheta)$ with a linear prolongation:

$$v^h(\mathbf{x}_\vartheta) = \mathbb{I}^h(v^H_{|\Gamma^H})(\mathbf{x}_\vartheta) = \vartheta v^H(\mathbf{x}_a) + (1-\vartheta)v^H(\mathbf{x}_b) \tag{4.3}$$

Similarly, we can define the restriction operator $\mathbb{R}^H$ from the finer grid to the coarse one. For instance, let $\mathbf{x}_a$ be a node of Γ^H and $\mathbf{x}_{\vartheta=0}$ be the node of Γ^h superposed to $\mathbf{x}_a$. Then, we can use a simple copy as the restriction operator:

$$v^H(\mathbf{x}_a) = v^h(\mathbf{x}_{\vartheta=0}). \tag{4.4}$$

We could also have used an arithmetic average of nearby points.

Iterative algorithm LDC is an iterative algorithm, for which Figure 4.5 shows a logical flow chart for when there is only one patch. The initialisation consists in solving the linear system $\mathcal{L}^H\phi^H = s^H$ on the nodes of the coarse grid Ω^H, with boundary conditions given by the physical problem data. This gives a solution ϕ^H_0. This solution is interpolated from Γ^H to get $\phi^h_{iter_{LDC}} = \mathbb{I}^h(\phi^H_{iter_{LDC}})$ on the fine exchange grid Γ^h, with $iter_{LDC} = 0$. These values are considered as the Dirichlet boundary conditions for the linear problem $\mathcal{L}^h\phi^h = s^h$. This yields a solution $\phi^h_{iter_{LDC}}$ defined on the whole fine grid $\Omega^h_\square$. We restrict this solution to get $\mathbb{R}^H(\phi^h_{iter_{LDC}})$ on $\Omega^H_\square$. For a given $iter_{LDC}$, we define a Right-Hand Side variable RHS^H on Ω^H as follows:

$$RHS^H = \begin{cases} \mathcal{L}^H\mathbb{R}^H(\phi^h_{iter_{LDC}}) & \text{on } \Omega^H_\square \cup \Gamma^H, \\ s^H & \text{on } (\Omega_\bigcirc \setminus \Gamma)^H. \end{cases} \tag{4.5}$$

The Right-Hand Side variable restricted to $\Omega^H_\square$, namely $RHS^H_\square$, is called the “Local Defect Correction”. Then we solve the coarse linear problem $\mathcal{L}^H\phi^H = RHS^H$ to get $\phi^H_{iter_{LDC}+1}$. Let ϵ_{LDC} be a small positive real number. We consider that the LDC algorithm has converged if the following comparison is true:

$$\frac{||\phi^H_{iter_{LDC}+1} - \phi^H_{iter_{LDC}}||_\infty}{||\phi^H_{iter_{LDC}}||_\infty} \leq \epsilon_{LDC}. \tag{4.6}$$

If it did converge, we keep the latest ϕ^H and ϕ^h results. If not, we loop back at the interpolation step on Γ^h; this adds an iteration to $iter_{LDC}$. In the case of a Poisson problem with only one patch, only a few iterations should suffice (of the order of magnitude of 2 or 3).

4.1.3 Convergence of LDC

According to our bibliography review, the literature authors lay out the proof of convergence of LDC only with specific hypotheses. Multiple authors have shown that it is possible to express the iterative step of the LDC algorithm as a matrix problem. If the LDC algorithm converges, then it means that this matrix problem has a fixed point.

Results on the convergence to the continuous solution (Ferket)

Ferket [64] obtains convergence results for the multi-grid problem by focusing on a composite grid approach on the composite grid Ω^c. Let us call $\bar{\phi}^c$ the fixed point of the matrix problem representing the composite grid problem, if it exists (see Theorem 4 for said matrix form). Let us still call ϕ_{exact} the solution to the original continuous problem (4.1) and let us call ϕ^c_{exact} its projection on the composite grid Ω^c. Then Ferket shows convergence results in [63], pp. 348–358.

Theorem 2. *If LDC has a fixed point and uses linear interpolation for $\mathbb{I}^h$, we have the following global discretisation error:*

$$||\bar{\phi}^c - \phi^c_{exact}||_\infty \leq C(C_{4\bigcirc}H^2 + C_{4\Box}h^2 + C_{2\Gamma}H) \tag{4.7}$$

where the constants $C_\star$ depend on higher derivatives of ϕ but not on h nor H. Additionally, C is a constant which does not depend on ϕ, h and H.

Theorem 3. *If LDC has a fixed point and uses quadratic interpolation for $\mathbb{I}^h$, we have the following global discretisation error:*

$$||\bar{\phi}^c - \phi^c_{exact}||_\infty \leq C(C_qH^2 + C_{4\Box}h^2). \tag{4.8}$$

where the constants $C_\star$ depend on higher derivatives of ϕ but not on h nor H. Additionally, C is a constant which does not depend on ϕ, h and H.

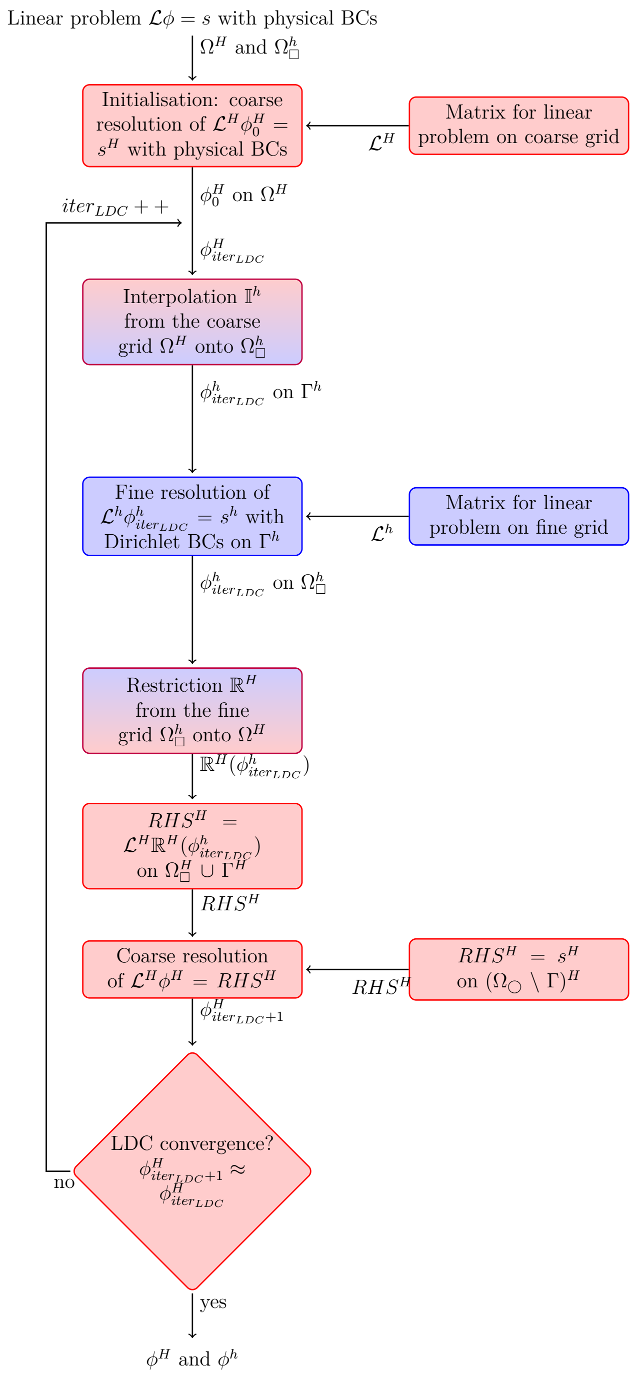

Linear problem $\mathcal{L}\phi = s$ with physical BCs
Ω^H and $\Omega^h_\square$
Initialisation: coarse resolution of $\mathcal{L}^H\phi_0^H = s^H$ with physical BCs
Matrix for linear problem on coarse grid
$\mathcal{L}^H$
ϕ_0^H on Ω^H
$iter_{LDC}++$
$\phi^H_{iter_{LDC}}$
Interpolation $\mathbb{I}^h$ from the coarse grid Ω^H onto $\Omega^h_\square$
$\phi^h_{iter_{LDC}}$ on Γ^h
Fine resolution of $\mathcal{L}^h\phi^h_{iter_{LDC}} = s^h$ with Dirichlet BCs on Γ^h
Matrix for linear problem on fine grid
$\mathcal{L}^h$
$\phi^h_{iter_{LDC}}$ on $\Omega^h_\square$
Restriction $\mathbb{R}^H$ from the fine grid $\Omega^h_\square$ onto Ω^H
$\mathbb{R}^H(\phi^h_{iter_{LDC}})$
$RHS^H = \mathcal{L}^H\mathbb{R}^H(\phi^h_{iter_{LDC}})$ on $\Omega^H_\square \cup \Gamma^H$
RHS^H
Coarse resolution of $\mathcal{L}^H\phi^H = RHS^H$
$RHS^H = s^H$ on $(\Omega_\bigcirc \setminus \Gamma)^H$
RHS^H
$\phi^H_{iter_{LDC}+1}$
LDC convergence? $\phi^H_{iter_{LDC}+1} \approx \phi^H_{iter_{LDC}}$
no
yes
ϕ^H and ϕ^h

Figure 4.5 – LDC algorithm on nodes, with one patch

Let us assume $\phi_{exact} \in \mathcal{C}^4(\Omega)$ and for $k \in \mathbb{N}$, let us note $\phi_{exact}^{(k)}$ the *family* of the kth derivatives of ϕ_{exact} in all spatial directions. Ferket gives the following properties:

$$C_{4\bigcirc} \propto \max\{\phi_{exact}^{(4)}(\mathbf{x}) | \mathbf{x} \in \Omega_{\bigcirc}\}, \tag{4.9}$$

$$C_{4\square} \propto \max\{\phi_{exact}^{(4)}(\mathbf{x}) | \mathbf{x} \in \Omega_{\square}\}, \tag{4.10}$$

$$C_{2\Gamma} \propto \max\{\phi_{exact}^{(2)}(\mathbf{x}) | \mathbf{x} \in \Gamma\}, \tag{4.11}$$

$$C_{3\Gamma} \propto \max\{\phi_{exact}^{(3)}(\mathbf{x}) | \mathbf{x} \in \Gamma\}, \tag{4.12}$$

$$C_q = C_{4\bigcirc} + C_{3\Gamma}. \tag{4.13}$$

Let us impose that $\Omega_\square$ is the area where ϕ_{exact} knows its largest variations, hence the name "area of interest". As a consequence, the constant $C_{4\square}$ is much larger than the constants $C_{4\bigcirc}$, $C_{2\Gamma}$ and C_q. Therefore, and keeping in mind that $\frac{H}{h} = r$ is a fixed ratio usually smaller than 10, the global discretisation error is driven by $C_{4\square}h^2$. It means that refining the area of interest $\Omega_\square^h$ indeed increases the precision.

As a conclusion, Ferket proves that if a fixed point exists for the iterative LDC algorithm, then the problem is equivalent to the one with a composite grid. He also proves the convergence with an error of H or H^2 depending on the interpolation.

Results on the existence of a fixed point (Anthonissen)

Anthonissen takes a different approach in his thesis [9] and his later published article [10]. We will here give the outline of the proof.

Matrix expression of the iterative process Let us separate the grids into four groups; respectively the fine grid on the local area $\Omega_\square^h$, the coarse grid on the local area $\Omega_\square^H$, the coarse grid on the interface Γ^H and the coarse grid on non-refined area $\Omega_\bigcirc^H = (\Omega \setminus \Omega_\square)^H$.

That way, we write ϕ^c as follows:

$$\phi^c = \begin{pmatrix} \phi_\square^h \\ \phi_\square^H \\ \phi_\Gamma^H \\ \phi_\bigcirc^H \end{pmatrix} \tag{4.14}$$

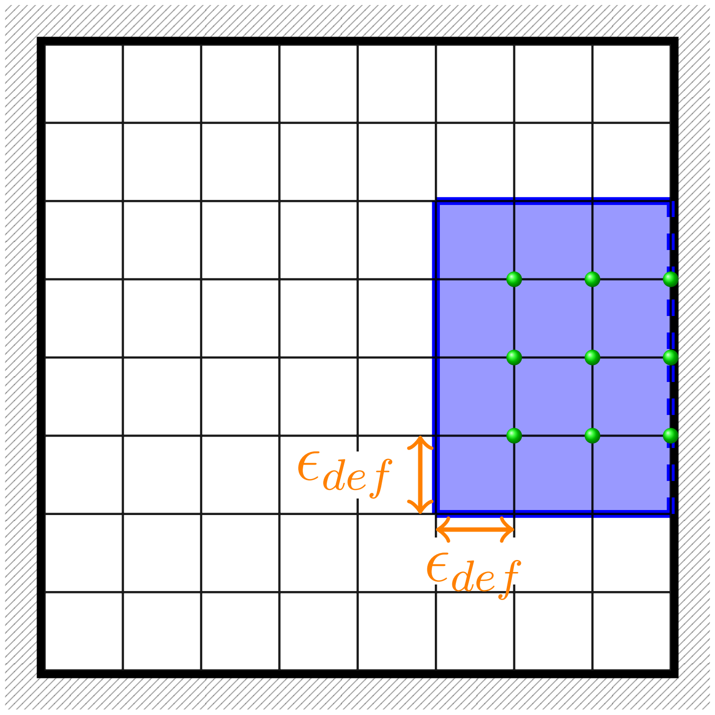


Figure 4.6 – Nodes of Ω^H_{def}

Similarly, we can define s^c for the source term, discretised on the composite grid. Let us also call $\phi^c_{iter_{LDC}}$ the value of ϕ as computed at step $iter_{LDC}$ of the LDC algorithm:

$$\phi^c_{iter_{LDC}} = \begin{pmatrix} \phi^h_{\square,iter_{LDC}} \\ \phi^H_{\square,iter_{LDC}} \\ \phi^H_{\Gamma,iter_{LDC}} \\ \phi^H_{\bigcirc,iter_{LDC}} \end{pmatrix} \tag{4.15}$$

Anthonissen introduces a region of the space called Ω^H_{def}. This grid is a subset of $\Omega^H_\square$; it is the inside of $\Omega^H_\square$, separated from the interface Γ^H by a “safety region” of width ϵ_{def}. Figure 4.6 shows the nodes of Ω^H_{def} when $\epsilon_{def} = H$. The space Ω^H_{def} plays a role in the restriction step of the iterative LDC algorithm: the defect is not calculated in the safety region. For this demonstration we choose to only restrict the values of $\phi^h_{\square,iter_{LDC}-1}$ onto Ω^H_{def} and not onto the safety region $\Omega^H_\square \setminus \Omega^H_{def}$. This is shown to be beneficial in previous literature (for instance in [9]).

Let us define the following operators:

- $\mathcal{L}^H$ is the expression of the elliptic operator discretised on the coarse grid Ω^H. Similarly, we define the operator $\mathcal{L}^h_\square$ as the elliptic operator discretised on the fine grid $\Omega^h_\square$. We also define $\mathcal{L}^H_\square$ and $\mathcal{L}^H_\bigcirc$ for the elliptic operator discretised on the sub-grids of Ω^H: respectively $\Omega^H_\square$ and $\Omega^H_\bigcirc$.

- $\mathbb{1}^H_\square$ is the multiplication by the characteristic function of Ω^H_{def}.

$$(\mathbb{1}^H_\square \phi^H_\square)(x,y) = \begin{cases} \phi^H_\square(x,y) & (x,y) \in \Omega^H_{def}, \\ 0 & (x,y) \in \Omega^H_\square \setminus \Omega^H_{def}. \end{cases} \tag{4.16}$$

- $\mathbb{I}^h$ is the interpolation operator from the coarse grid Ω^H onto the fine grid $\Omega^h_\square$.

- $B^h_{\square,\Gamma}$ is the operator expressing a source term as a function of Dirichlet boundary conditions on Γ, for the area $\Omega_\square$. This means that solving the elliptic equation (4.1) on the refined area $\Omega_\square$ with Dirichlet boundary conditions equal to the interpolation of $\phi^H_{iter_{LDC}}$ on the interface Γ is discretised as follows:

$$\mathcal{L}^h_\square \phi^h_{\square,iter_{LDC}} = s^h_\square - B^h_{\square,\Gamma} \mathbb{I}^h \phi^H_{\Gamma,iter_{LDC}}. \tag{4.17}$$

  Similarly, we define the operator $B^H_{\square,\Gamma}$ for the numerical resolution of the elliptic equation coarsely discretised on only $\Omega^H_\square$:

$$\mathcal{L}^H_\square \phi^H_{\square,iter_{LDC}} = s^H_\square - B^H_{\square,\Gamma} \phi^H_{\Gamma,iter_{LDC}}. \tag{4.18}$$

  We define $B^H_{\Gamma,\square}$ and $B^H_{\Gamma,\bigcirc}$ for the resolution on Ω^H_Γ:

$$\mathcal{L}^H_\Gamma \phi^H_{\Gamma,iter_{LDC}} = s^H_\Gamma - B^H_{\Gamma,\square} \phi^H_{\square,iter_{LDC}} - B^H_{\Gamma,\bigcirc} \phi^H_{\bigcirc,iter_{LDC}}. \tag{4.19}$$

  We define also the operator $B^H_{\bigcirc,\Gamma}$ for the resolution on $\Omega^H_\bigcirc$:

$$\mathcal{L}^H_\bigcirc \phi^H_{\bigcirc,iter_{LDC}} = s^H_\bigcirc - B^H_{\bigcirc,\Gamma} \phi^H_{\Gamma,iter_{LDC}}. \tag{4.20}$$

Theorem 4. *We can write the LDC process at a given iteration $iter_{LDC}$ as a matrix equation for ϕ discretised on the composite grid:*

$$\mathcal{L}^c \phi^c_{iter_{LDC}+1} = \mathcal{S}^c \phi^c_{iter_{LDC}} + s^c. \tag{4.21}$$

The expressions of $\mathcal{L}^c$ and $\mathcal{S}^c$ are given by Equation (4.26) and Equation (4.27) respectively.

Proof. First, we rewrite Equation (4.17) in a matrix form to get the first line of the goal matrix equation:

$$\begin{pmatrix} \mathcal{L}^h_\square & 0 & B^h_{\square,\Gamma}\mathbb{I}^h & 0 \end{pmatrix} \begin{pmatrix} \phi^h_{\square,iter_{LDC}} \\ \phi^H_{\square,iter_{LDC}} \\ \phi^H_{\Gamma,iter_{LDC}} \\ \phi^H_{\bigcirc,iter_{LDC}} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} s^h_\square \\ s^H_\square, \\ s^H_\Gamma \\ s^H_\bigcirc \end{pmatrix}. \quad (4.22)$$

Second, we rewrite Equation (4.18) in a matrix form to get the second line of the goal matrix equation. We replace the source term s^H_i by the local defect correction $RHS^H_\square$ of step $i-1$ when inside $\Omega_\square$.

$$\begin{aligned} &\begin{pmatrix} 0 & \mathcal{L}^H_\square & B^H_{\square,\Gamma} & 0 \end{pmatrix} \begin{pmatrix} \phi^h_{\square,iter_{LDC}} \\ \phi^H_{\square,iter_{LDC}} \\ \phi^H_{\Gamma,iter_{LDC}} \\ \phi^H_{\bigcirc,iter_{LDC}} \end{pmatrix} \\ &\quad - \begin{pmatrix} \mathbb{1}^H_\square \mathcal{L}^H_\square \mathbb{R}^H & 0 & \mathbb{1}^H_\square B^H_{\square,\Gamma} & 0 \end{pmatrix} \begin{pmatrix} \phi^h_{\square,iter_{LDC}-1} \\ \phi^H_{\square,iter_{LDC}-1} \\ \phi^H_{\Gamma,iter_{LDC}-1} \\ \phi^H_{\bigcirc,iter_{LDC}-1} \end{pmatrix} \\ &\quad = \begin{pmatrix} 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} s^h_\square \\ s^H_\square, \\ s^H_\Gamma \\ s^H_\bigcirc \end{pmatrix}. \end{aligned} \quad (4.23)$$

Third, we rewrite Equation (4.19) in a matrix form to get the following line of the goal matrix equation:

$$\begin{pmatrix} 0 & B^H_{\Gamma,\square} & \mathcal{L}^H_\Gamma & B^H_{\Gamma,\bigcirc} \end{pmatrix} \begin{pmatrix} \phi^h_{\square,iter_{LDC}} \\ \phi^H_{\square,iter_{LDC}} \\ \phi^H_{\Gamma,iter_{LDC}} \\ \phi^H_{\bigcirc,iter_{LDC}} \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} s^h_\square \\ s^H_\square, \\ s^H_\Gamma \\ s^H_\bigcirc \end{pmatrix}. \quad (4.24)$$

Fourth, we rewrite Equation (4.20) in a matrix form to get the last line of the goal matrix equation:

$$\begin{pmatrix} 0 & 0 & B^H_{\bigcirc,\Gamma} & \mathcal{L}^H_\bigcirc \end{pmatrix} \begin{pmatrix} \phi^h_{\square,iter_{LDC}} \\ \phi^H_{\square,iter_{LDC}} \\ \phi^H_{\Gamma,iter_{LDC}} \\ \phi^H_{\bigcirc,iter_{LDC}} \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} s^h_\square \\ s^H_\square, \\ s^H_\Gamma \\ s^H_\bigcirc \end{pmatrix}. \quad (4.25)$$

As a conclusion we can define $\mathcal{L}^c$ and $\mathcal{S}^c$ with the following equations:

$$\mathcal{L}^c = \begin{pmatrix} \mathcal{L}^h_\square & 0 & B^h_{\square,\Gamma}\mathbb{I}^h & 0 \\ 0 & \mathcal{L}^H_\square & B^H_{\square,\Gamma} & 0 \\ 0 & B^H_{\Gamma,\square} & \mathcal{L}^H_\Gamma & B^H_{\Gamma,\bigcirc} \\ 0 & 0 & B^H_{\bigcirc,\Gamma} & \mathcal{L}^H_\bigcirc \end{pmatrix}, \tag{4.26}$$

$$\mathcal{S}^c = \begin{pmatrix} 0 & 0 & 0 & 0 \\ \mathbb{1}^H_\square \mathcal{L}^H_\square \mathbb{R}^H & 0 & \mathbb{1}^H_\square B^H_{\square,\Gamma} & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}. \tag{4.27}$$

This ensures the desired equation:

$$\mathcal{L}^c \phi^c_{iter_{LDC}} = \mathcal{S}^c \phi^c_{iter_{LDC}-1} + s^c. \tag{4.28}$$

If we replace $iter_{LDC}$ by $iter_{LDC}+1$, we get Equation (4.21) from Theorem 4. □

Matrix expression of the iteration error If the LDC algorithm converges, let us note $\bar{\phi}^c$ the fixed point:

$$\bar{\phi}^c = \begin{pmatrix} \bar{\phi}^h_\square \\ \bar{\phi}^H_\square \\ \bar{\phi}^H_\Gamma \\ \bar{\phi}^H_\bigcirc \end{pmatrix}. \tag{4.29}$$

Let us call $e^H_{iter_{LDC}} = \phi^H_{iter_{LDC}} - \bar{\phi}^H$ the iteration error of the LDC method on the coarse grid. Let us call $e_{\Gamma,iter_{LDC}} = \phi_{\Gamma,iter_{LDC}} - \bar{\phi}_\Gamma$ the iteration error on the nodes of the interface Γ.

Anthonissen proves the following theorem:

Theorem 5. *There exists a matrix M, defined by (4.38), such that*

$$e^H_{\Gamma,iter_{LDC}+1} = M e^H_{\Gamma,iter_{LDC}}. \tag{4.30}$$

Proof. The definition of $\bar{\phi}^c$ is as follows:

$$\mathcal{L}^c \bar{\phi}^c = \mathcal{S}^c \bar{\phi}^c + s^c. \tag{4.31}$$

We substract (4.21) and (4.31):

$$\mathcal{L}^c e^c_{iter_{LDC}} = \mathcal{S}^c e^c_{iter_{LDC}-1}. \tag{4.32}$$

This rewrites with developed matrices:

$$\begin{pmatrix} \mathcal{L}^h_\square & 0 & B^h_{\square,\Gamma}\mathbb{I}^h & 0 \\ 0 & \mathcal{L}^H_\square & B^H_{\square,\Gamma} & 0 \\ 0 & B^H_{\Gamma,\square} & \mathcal{L}^H_\Gamma & B^H_{\Gamma,\bigcirc} \\ 0 & 0 & B^H_{\bigcirc,\Gamma} & \mathcal{L}^H_\bigcirc \end{pmatrix} \begin{pmatrix} e^h_{\square,iter_{LDC}} \\ e^H_{\square,iter_{LDC}} \\ e^H_{\Gamma,iter_{LDC}} \\ e^H_{\bigcirc,iter_{LDC}} \end{pmatrix} = \begin{pmatrix} 0 \\ \mathbb{1}^H_\square \mathcal{L}^H_\square \mathbb{R}^H e^h_{\square,iter_{LDC}-1} + \mathbb{1}^H_\square B^H_{\square,\Gamma} e^H_{\Gamma,iter_{LDC}-1} \\ 0 \\ 0 \end{pmatrix}. \tag{4.33}$$

The first line gives the following equation:

$$e^h_{\square,iter_{LDC}} = -\left(\mathcal{L}^h_\square\right)^{-1} B^h_{\square,\Gamma}\mathbb{I}^h e^H_{\Gamma,iter_{LDC}} \tag{4.34}$$

which also means, by replacing $iter_{LDC}$ by $iter_{LDC} - 1$:

$$e^h_{\square,iter_{LDC}-1} = -\left(\mathcal{L}^h_\square\right)^{-1} B^h_{\square,\Gamma}\mathbb{I}^h e^H_{\Gamma,iter_{LDC}-1}. \tag{4.35}$$

We thus rewrite the three last lines of (4.33):

$$\begin{pmatrix} \mathcal{L}^H_\square & B^H_{\square,\Gamma} & 0 \\ B^H_{\Gamma,\square} & \mathcal{L}^H_\Gamma & B^H_{\Gamma,\bigcirc} \\ 0 & B^H_{\bigcirc,\Gamma} & \mathcal{L}^H_\bigcirc \end{pmatrix} \begin{pmatrix} e^H_{\square,iter_{LDC}} \\ e^H_{\Gamma,iter_{LDC}} \\ e^H_{\bigcirc,iter_{LDC}} \end{pmatrix} = \begin{pmatrix} -\mathbb{1}^H_\square \mathcal{L}^H_\square \mathbb{R}^H \left(\mathcal{L}^h_\square\right)^{-1} B^h_{\square,\Gamma}\mathbb{I}^h e^H_{\Gamma,iter_{LDC}-1} + \mathbb{1}^H_\square B^H_{\square,\Gamma} e^H_{\Gamma,iter_{LDC}-1} \\ 0 \\ 0 \end{pmatrix}. \tag{4.36}$$

We simplify it:

$$\mathcal{L}^H e^H_{iter_{LDC}} = \begin{pmatrix} I \\ 0 \\ 0 \end{pmatrix} \mathbb{1}^H_\square \left(B^H_{\square,\Gamma} - \mathcal{L}^H_\square \mathbb{R}^H \left(\mathcal{L}^h_\square\right)^{-1} B^h_{\square,\Gamma}\mathbb{I}^h \right) e^H_{\Gamma,iter_{LDC}-1}. \tag{4.37}$$

We now just may to define M as follows:

$$M = \begin{pmatrix} 0 & I & 0 \end{pmatrix} \left(\mathcal{L}^H\right)^{-1} \begin{pmatrix} I \\ 0 \\ 0 \end{pmatrix} \mathbb{1}^H_\square \left(B^H_{\square,\Gamma} - \mathcal{L}^H_\square \mathbb{R}^H \left(\mathcal{L}^h_\square\right)^{-1} B^h_{\square,\Gamma}\mathbb{I}^h \right) \tag{4.38}$$

such that

$$e^H_{\Gamma,iter_{LDC}} = M e^H_{\Gamma,iter_{LDC}-1}. \tag{4.39}$$

If we replace $iter_{LDC}$ by $iter_{LDC} + 1$, we get Equation (4.30) from Theorem 5. □

Anthonissen also proves the following result:

Theorem 6. *Consider the iterative process, where M is defined by Equation (4.38):*

$$e^H_{\Gamma,iter_{LDC}+1} = M e^H_{\Gamma,iter_{LDC}}. \tag{4.40}$$

Although it is only located on the interface, if it converges, then the LDC algorithm converges on the whole coarse grid.

Proof. Let us assume that the process (4.40) converges. Then we obviously have convergence of LDC on Γ^H. Furthermore, thanks to (4.34), we get the convergence on $\Omega^h_\square$. And thanks to (4.37), we have the convergence of LDC on Ω^H. Hence the convergence on Ω^c. □

Thanks to Theorem 6, it is sufficient to compute the norm $||M||_\infty$ of the matrix M and prove it is strictly smaller than 1. We will express M as the product of two matrices, namely

$$M = M_1 M_2 \tag{4.41}$$

where

$$M_1 = (0\ I\ 0)(\mathcal{L}^H)^{-1}\begin{pmatrix} I \\ 0 \\ 0 \end{pmatrix}, \tag{4.42}$$

$$M_2 = \mathbb{1}^H_\square \left[B^H_{\square,\Gamma} - \mathcal{L}^H_\square \mathbb{R}^H (\mathcal{L}^h_\square)^{-1} B^h_{\square,\Gamma} \mathbb{I}^h \right]. \tag{4.43}$$

We note that $||M||_\infty \leq ||M_1||_\infty ||M_2||_\infty$.

Particular case of Laplacian problem with trigonometric interpolation Anthonissen focuses on the specific case of a Poisson problem ($\mathcal{L} = -\Delta$):

$$\begin{cases} -\Delta\phi = s & \text{in } \Omega, \\ \phi = g & \text{on } \partial\Omega. \end{cases} \tag{4.44}$$

By choosing to use the five-point stencil for the Laplacian operator Δ^H referenced at Equation (B.4) and trigonometric interpolation for the operator $\mathbb{I}^h$, he obtains the property hereunder.

Theorem 7. *The LDC method applied to solve Equation (4.44) converges to a fixed point when using the classical five-point stencil for the Laplacian operator Δ^H and trigonometric interpolation for the operator $\mathbb{I}^h$.*

Proof. Since Δ^H is the five-point stencil Laplacian, as shown for instance in [75], we have the following equality for $||M_1||_\infty$:

$$||M_1||_\infty = \frac{1}{8}. \tag{4.45}$$

Because the problem is a Poisson one and $\mathbb{I}^h$ is trigonometric, Anthonissen uses orthogonality arguments to prove that we can compute the exact value of $M_2 g^H$ for any vector g^H, showing the following majoration for all $(x, y) \in \Omega^H_{def}$:

$$|M_2 g^H(x, y)| \leq (C_1 H^2 + D_1 h^2)||g^H||_\infty. \tag{4.46}$$

Hence

$$||M_2||_\infty \leq (C_1 H^2 + D_1 h^2). \tag{4.47}$$

The result is a majoration of $||M||_\infty$:

$$||M||_\infty \leq CH^2 + Dh^2 \tag{4.48}$$

where C and D are independent of H and h. So if we take H (and h) sufficiently small, then we can impose

$$||M||_\infty < 1. \tag{4.49}$$

Since we have $e^H_{\Gamma,iter_{LDC}+1} = Me^H_{\Gamma,iter_{LDC}}$, we also can write an explicit form:

$$e^H_{\Gamma,iter_{LDC}} = M^{iter_{LDC}} e^H_{\Gamma,0} \tag{4.50}$$

So $e^H_{\Gamma,iter_{LDC}}$ converges to 0 as $iter_{LDC} \to \infty$. As a consequence, LDC converges to a fixed point $\bar{\phi}^c$. □

As a conclusion of Anthonissen's contribution, it is now proven that for the Poisson problem using trigonometric interpolation, if we impose a sufficiently small H (and h) independent of $iter_{LDC}$, then LDC converges as $iter_{LDC} \to \infty$. In other words, a fixed point $\bar{\phi}^c$ does exist for the LDC iterative algorithm. When $H \to 0$, this fixed point converges to ϕ^c_{exact}.

4.1.4 Own implementation of LDC

We too implement the LDC method to solve elliptic equations. We solve equations for which the elliptic operator is a Laplacian Δ (see Section 4.2 and 5.1) or rather derived from a Laplacian (see Section 5.2). Among the possible interpolation methods for $\mathbb{I}^h$, we choose the linear interpolation. Our implementation however differs from the previously detailed works in two ways:

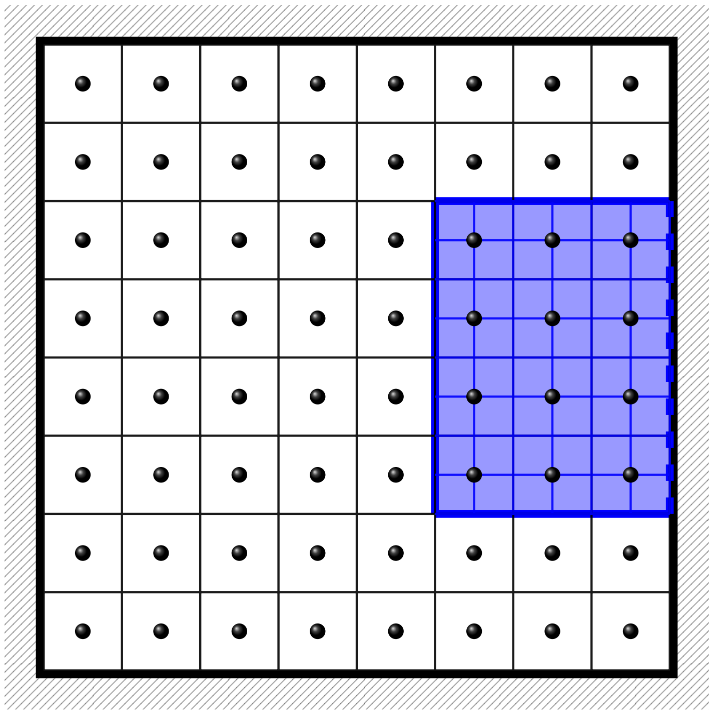


Figure 4.7 – Location of the centre of the cells of the coarse grid Ω^H

1. the localisation of the discretised values on the grid,
2. the number of patches.

Values at the centre of cells

Ferket and Anthonissen localise the discretised values of ϕ at the nodes of the grids Ω^H, $\Omega^h_\square$ and Ω^c. As for us, we choose to locate the discretised values of ϕ at the centre of cells because of data structures reasons: we developed our AMR toolbox with centred data in mind (see Section 4.3.2). Figure 4.7 shows the location of the cell centres for the coarse grid Ω^H and Figure 4.8 shows the location of cell centres for the fine grid $\Omega^h_\square$.

As a consequence, we cannot consider the interface Γ, nor Γ^H or Γ^h. Instead, we use ghost cells. Let n_{GC} be a small integer; we extend the patches by n_{GC} coarse cells in all directions around them, as demonstrated on Figure 4.9 in light blue around the patch. We call "ghost cells" the cells which belong to the extension of the local area $\square$. We note GC the area composed of the ghost cells: the coarse mesh covering GC is noted GC^H and the fine mesh is noted GC^h.

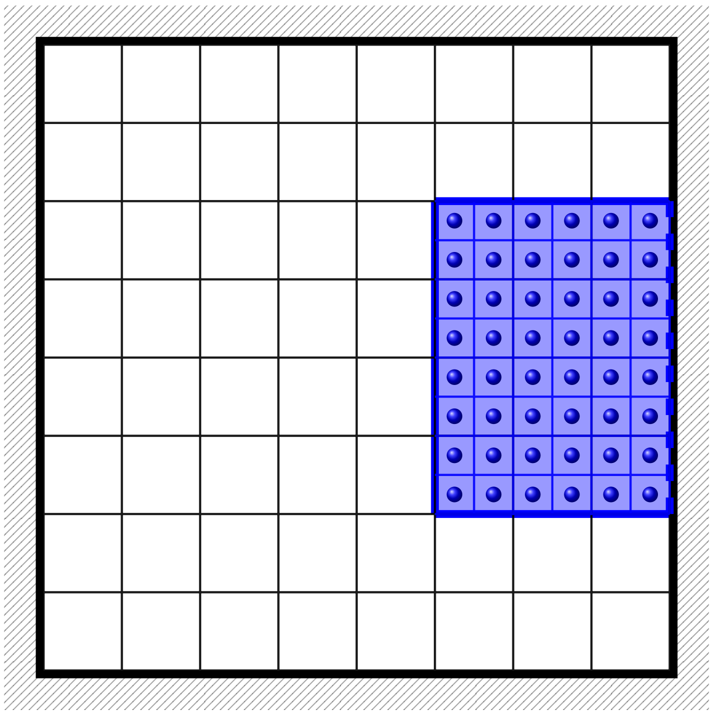


Figure 4.8 – Location of the centre of the cells of the fine grid $\Omega^h_\square$

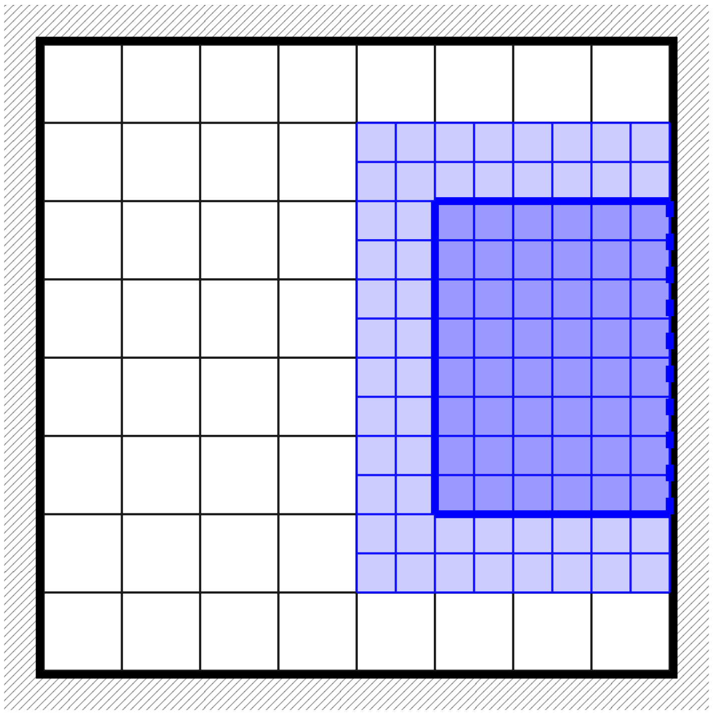


Figure 4.9 – Location of ghost cells, for $n_{GC} = 1$ and $r = 2$

Figure 4.10 shows a logical flow chart explaining the LDC algorithm for when there is only one patch and when the values are located at the centre of cells. The initialisation consists in solving the linear system $\mathcal{L}^H\phi^H = s^H$ on the centre of cells of the coarse grid Ω^H, with boundary conditions given by the physical problem data. This gives a solution ϕ_0^H. This solution is interpolated from the coarse ghost cells GC^H to get $\phi^h_{iter_{LDC}} = \mathbb{I}^h(\phi^H_{iter_{LDC}})$ on the fine ghost cells GC^h, with $iter_{LDC} = 0$. These values are considered as the Dirichlet boundary conditions for the linear problem $\mathcal{L}^h\phi^h = s^h$ in $\Omega^h_\square$. This yields a solution $\phi^h_{iter_{LDC}}$ defined on the whole fine grid $\Omega^h_\square$. We restrict this solution to get $\mathbb{R}^H(\phi^h_{iter_{LDC}})$ on $\Omega^H_\square$. For a given $iter_{LDC}$, we define a Right-Hand Side variable RHS^H on Ω^H as follows:

$$RHS^H = \begin{cases} \mathcal{L}^H\mathbb{R}^H(\phi^h_{iter_{LDC}}) & \text{on } \Omega^H_\square, \\ s^H & \text{on } \Omega^H_\bigcirc. \end{cases} \tag{4.51}$$

The Right-Hand Side variable restricted to $\Omega^H_\square$, namely $RHS^H_\square$, is also called the "Local Defect Correction". Then we solve the coarse linear problem $\mathcal{L}^H\phi^H = RHS^H$ to get $\phi^H_{iter_{LDC}+1}$. Let ϵ_{LDC} be a small positive real number. We consider that the LDC algorithm has converged if the following comparison is true:

$$\frac{||\phi^H_{iter_{LDC}+1} - \phi^H_{iter_{LDC}}||_\infty}{||\phi^H_{iter_{LDC}}||_\infty} \leq \epsilon_{LDC}. \tag{4.52}$$

If it did converge, we keep the latest ϕ^H and ϕ^h results. If not, we loop back at the interpolation step on the ghost cells GC^h; this adds an iteration to $iter_{LDC}$. In the case of a Poisson problem with only one patch, similarly as before for the localisation on nodes, only a few iterations $iter_{LDC}$ should suffice (2 or 3).

As a side note, we can add that we in fact already used ghost cells for hyperbolic equations; to have values for variables on the fine grid outside of $\Omega^h_\square$. Let us consider the example of the advection equation $\partial_t Y + \mathbf{u}\cdot\nabla Y = 0$ where $\mathbf{u}$ is constant, uniform and with a positive value in the x direction:

$$\partial_t Y + u_x \partial_x Y = 0. \tag{4.53}$$

Let us assume we have a 2D rectangular patch covering a desired refined area: we index the fine cells with $(i,j) \in [\![0, n_x - 1]\!] \times [\![0, n_y - 1]\!]$. Let us say we choose the explicit upwind advection scheme, like in Section 3.4.1:

$$\forall (i,j) \in [\![1, n_x - 1]\!] \times [\![0, n_y - 1]\!],$$
$$\frac{Y^h_{iter_t+1}(i,j) - Y^h_{iter_t}(i,j)}{\Delta t} + u_x \frac{Y^h_{iter_t}(i,j) - Y^h_{iter_t}(i-1,j)}{\Delta x} = 0. \tag{4.54}$$

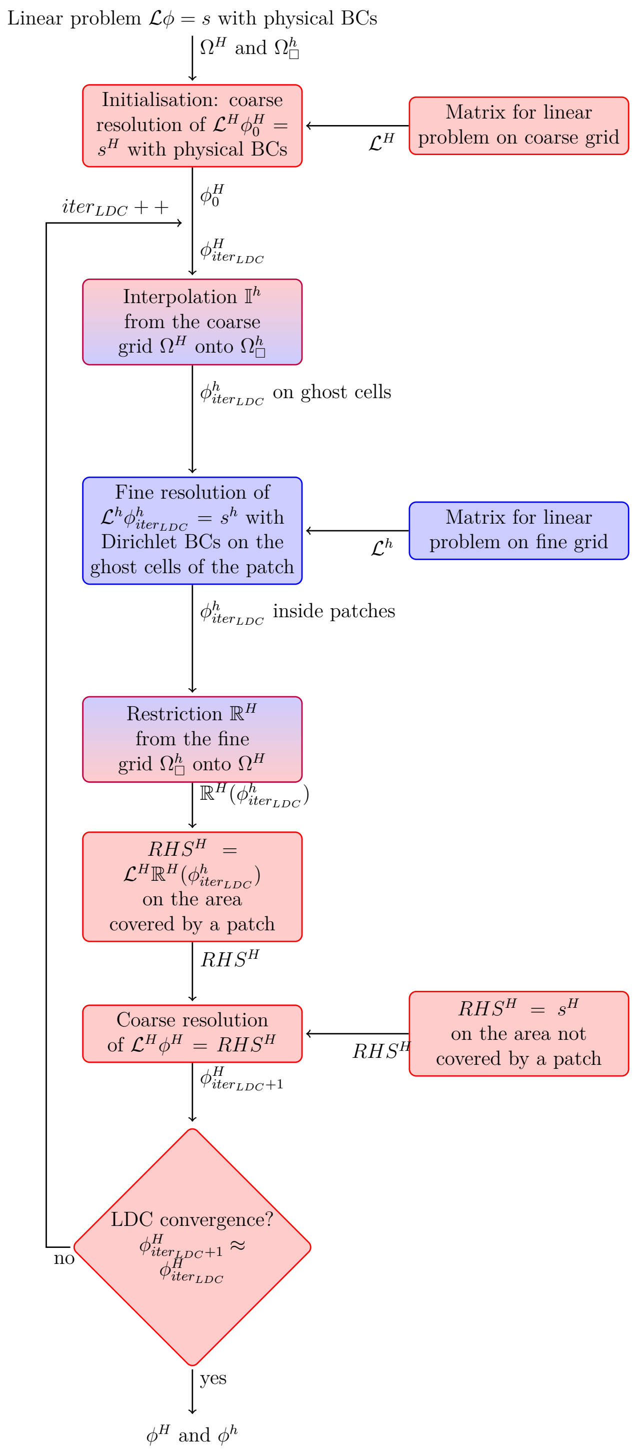

Linear problem $\mathcal{L}\phi = s$ with physical BCs
Ω^H and $\Omega^h_\square$
Initialisation: coarse resolution of $\mathcal{L}^H\phi^H_0 = s^H$ with physical BCs
Matrix for linear problem on coarse grid
$\mathcal{L}^H$
ϕ^H_0
$iter_{LDC}++$
$\phi^H_{iter_{LDC}}$
Interpolation $\mathbb{I}^h$ from the coarse grid Ω^H onto $\Omega^h_\square$
$\phi^h_{iter_{LDC}}$ on ghost cells
Fine resolution of $\mathcal{L}^h\phi^h_{iter_{LDC}} = s^h$ with Dirichlet BCs on the ghost cells of the patch
Matrix for linear problem on fine grid
$\mathcal{L}^h$
$\phi^h_{iter_{LDC}}$ inside patches
Restriction $\mathbb{R}^H$ from the fine grid $\Omega^h_\square$ onto Ω^H
$\mathbb{R}^H(\phi^h_{iter_{LDC}})$
$RHS^H = \mathcal{L}^H\mathbb{R}^H(\phi^h_{iter_{LDC}})$ on the area covered by a patch
RHS^H
Coarse resolution of $\mathcal{L}^H\phi^H = RHS^H$
$RHS^H = s^H$ on the area not covered by a patch
RHS^H
$\phi^H_{iter_{LDC}+1}$
LDC convergence? $\phi^H_{iter_{LDC}+1} \approx \phi^H_{iter_{LDC}}$
no
yes
ϕ^H and ϕ^h

Figure 4.10 – LDC algorithm on cells, with one patch

This computation is problematic when $i = 0$; that is to say on the left border of the patch. So for the value $Y^h_{iter_t}(-1, j)$, we use the value of $Y^h_{iter_t}$ in the fine ghost cell directly to the left of cell $(0, j)$. For schemes with a wider stencil (WENO for intance), it is necessary to have a sufficient number of fine ghost cells around the patches.

Multiple patches

In the majority of our computations, we may have multiple patches and more often than not, the patches touch each other. If this is the case, we follow the common practice mentioned by Anthonissen [10]. We consider that patches which touch each other form the decomposition of a domain formed by the reunion of them.

Figure 4.11 shows an $\Omega^h_\square$ covered by two patches, P_1 in blue and P_2 in red, which touch each other. Therefore the patches have three kinds of ghost cells:

- ghost cells exterior to $\Omega^h_\square$ and owned by one patch only, represented in light blue for P_1 and in light red for P_2,
- ghost cells exterior to $\Omega^h_\square$ and owned by the two patches, represented in light purple,
- ghost cells covered by the neighbouring patch, represented with north-east lines for P_1 and with north-west lines for P_2.

We decide to iterate a Schwarz domain decomposition algorithm to obtain a correct fine solution on the fine grids. It means that for a given LDC iteration $iter_{LDC}$, we iterate multiple times with Schwarz iterations numbered $iter_{Sch}$. After the value in the fine ghost cells is deduced from the coarse grid, we solve the elliptic problem $\mathcal{L}^h \phi^h_{iter_{LDC}} = s^h$ with Dirichlet boundary conditions in the said ghost cells. This gives us $\phi^h_{iter_{LDC}}(iter_{Sch} = 0)$. We then take the value of $\phi^h_{iter_{LDC}}(iter_{Sch} = 0)$ from the ghost cells that are covered by another patch (represented by oblique lines on Figure 4.11). They are the update of the Dirichlet boundary conditions of a new resolution of $\mathcal{L}^h \phi^h_{iter_{LDC}} = s^h$. This gives us $\phi^h_{iter_{LDC}}(iter_{Sch} = 1)$. Let ϵ_{Sch} be a small positive real number. We consider that the Schwarz domain decomposition algorithm has converged if the following comparison is true:

$$\frac{||\phi^H_{iter_{LDC}}(iter_{Sch} + 1) - \phi^H_{iter_{LDC}}(iter_{Sch})||_\infty}{||\phi^H_{iter_{LDC}}(iter_{Sch})||_\infty} \leq \epsilon_{Sch}. \tag{4.55}$$

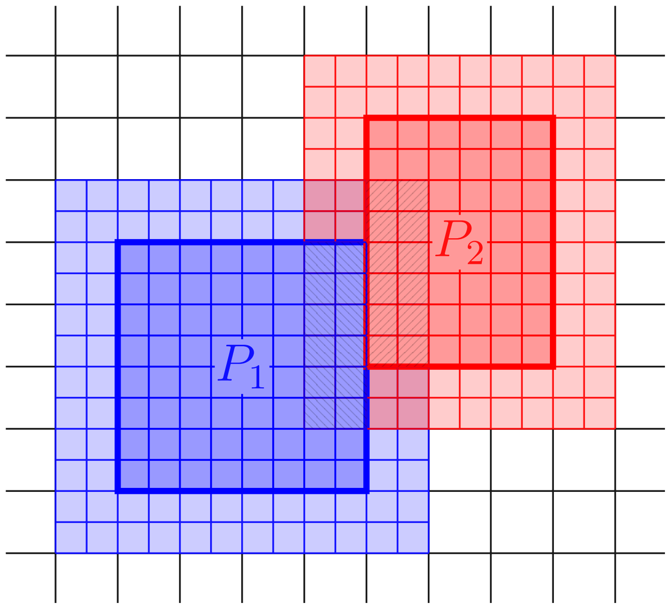


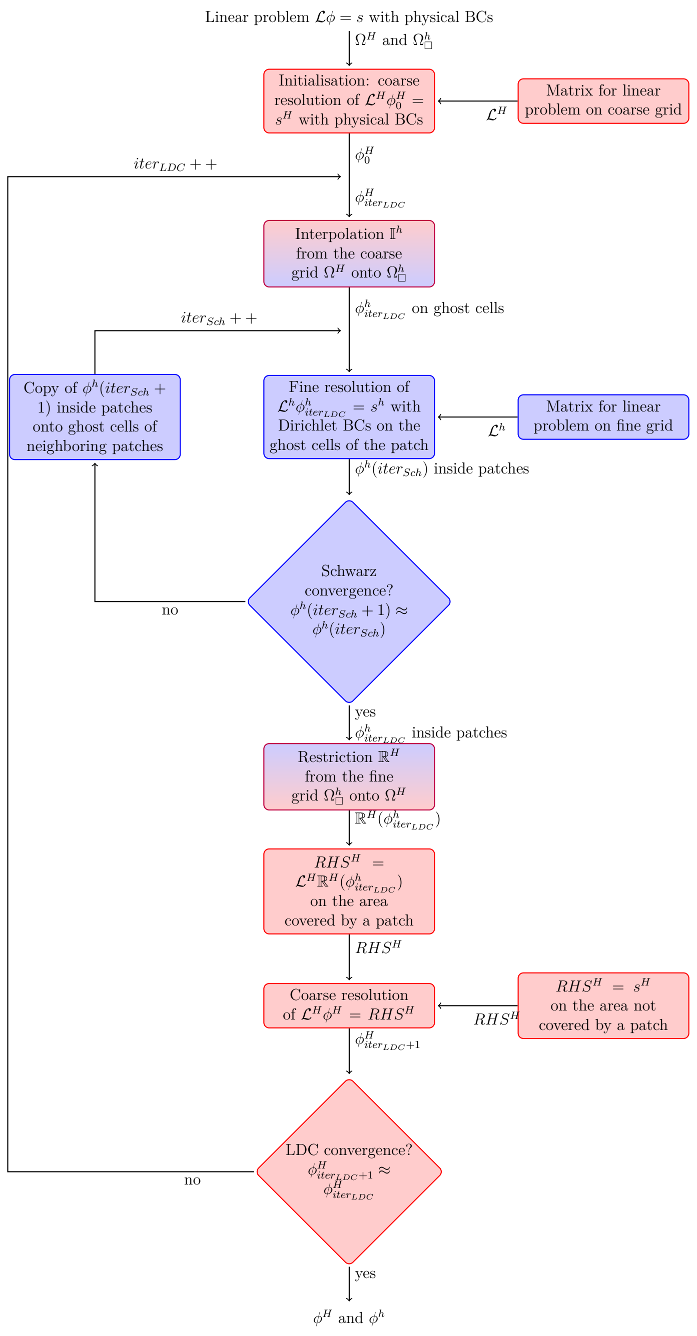
Figure 4.11 – Two patches touching each other

If it did converge, we keep the latest $\phi^h_{iter_{LDC}}$ results. If not, we loop back at the update of the value in the ghost cells covered by neighbouring patches; this adds an iteration to $iter_{Sch}$. Figure 4.12 shows a flowchart of the LDC algorithm with the values at the centre of cells when we make the Schwarz algorithm intervene. According to our tests, not using the domain decomposition technique results in the non-convergence of the LDC algorithm.

4.2 Hyperbolic-elliptic coupling with ABV

4.2.1 ABV model and verification

We wanted to test out a coupling between a hyperbolic part and an elliptic part thanks to the ABV model, which we presented earlier in Section 2.1.6. As a reminder, the ABV model is represented by the following set of equa-



Linear problem $\mathcal{L}\phi = s$ with physical BCs
Ω^H and $\Omega^h_\square$
Initialisation: coarse resolution of $\mathcal{L}^H\phi^H_0 = s^H$ with physical BCs
Matrix for linear problem on coarse grid
$\mathcal{L}^H$
ϕ^H_0
$iter_{LDC}++$
$\phi^H_{iter_{LDC}}$
Interpolation $\mathbb{I}^h$ from the coarse grid Ω^H onto $\Omega^h_\square$
$\phi^h_{iter_{LDC}}$ on ghost cells
$iter_{Sch}++$
Copy of $\phi^h(iter_{Sch}+1)$ inside patches onto ghost cells of neighboring patches
Fine resolution of $\mathcal{L}^h\phi^h_{iter_{LDC}} = s^h$ with Dirichlet BCs on the ghost cells of the patch
Matrix for linear problem on fine grid
$\mathcal{L}^h$
$\phi^h(iter_{Sch})$ inside patches
Schwarz convergence? $\phi^h(iter_{Sch}+1) \approx \phi^h(iter_{Sch})$
no
yes
$\phi^h_{iter_{LDC}}$ inside patches
Restriction $\mathbb{R}^H$ from the fine grid $\Omega^h_\square$ onto Ω^H
$\mathbb{R}^H(\phi^h_{iter_{LDC}})$
$RHS^H = \mathcal{L}^H\mathbb{R}^H(\phi^h_{iter_{LDC}})$ on the area covered by a patch
RHS^H
Coarse resolution of $\mathcal{L}^H\phi^H = RHS^H$
$RHS^H = s^H$ on the area not covered by a patch
RHS^H
$\phi^H_{iter_{LDC}+1}$
LDC convergence? $\phi^H_{iter_{LDC}+1} \approx \phi^H_{iter_{LDC}}$
no
yes
ϕ^H and ϕ^h

Figure 4.12 – LDC algorithm on cells, with Schwarz domain decomposition

tions, where $\psi(t)$ is given:

$$\begin{cases} \partial_t Y + \mathbf{u} \cdot \nabla Y = 0, \\ \Delta \phi = \psi(t) \left(Y - \frac{1}{|\Omega|} \int_\Omega Y dx \right), \\ \nabla \phi \cdot \mathbf{n}|_{\partial\Omega} = 0, \\ \mathbf{u} = \nabla \phi. \end{cases} \tag{4.56}$$

We also have to choose an initial condition for $Y(t = 0)$ on Ω.

We also remind the analytic expression for the volume V of the bubble as a function of $\psi(t)$, which is useful for later tests, as introduced in Theorem 1:

$$V^{-1} = \left(\frac{1}{V(0)} - \frac{1}{|\Omega|} \right) \exp(-\psi(t)) + \frac{1}{|\Omega|}. \tag{4.57}$$

We said this formula was very useful for verification purposes of numerical simulations. For instance, Mekkas in [104] or Penel et al. in [116] implement an AMR strategy to represent the ABV model. This is our aim too in what follows.

4.2.2 Simulation parameters

Our own implementation of LDC permits us to make the simulation of the ABV model. We choose, for each time step of the model ($t = n\Delta t$ to $t = (n+1)\Delta t$), to compute first the hyperbolic part to get Y_{n+1} based on $u_n = \nabla \phi_n$. Then we use the LDC algorithm to find ϕ_{n+1} with multiple convergence iterations. Given (Y_n, ϕ_n):

1. compute Y_{n+1} as a function of ϕ_n,
2. compute ϕ_{n+1} as a function of Y_{n+1}.

We choose a periodic pulsation function:

$$\psi(t) = \frac{1}{2} \cos \left(\frac{2\pi t}{12} \right). \tag{4.58}$$

As for the initial conditions, we choose to draw a complicated case in order to challenge the robustness of our computation.

1. The bubble (i.e. the domain where $Y = 1$) is a Zalesak slotted disk in 2D, a slotted sphere in 3D. This is a disk with a sharp notch cut inside it. This geometrical form is usually used for advection verification tests;

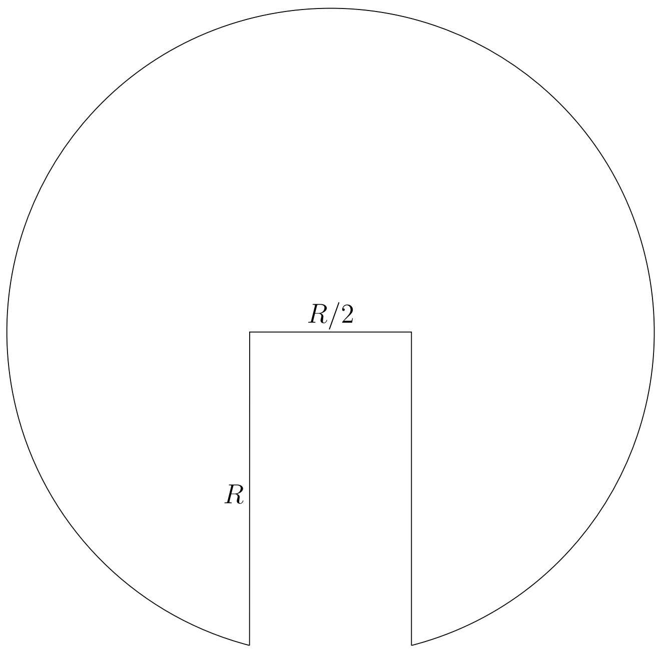


Figure 4.13 – 2D Zalesak disk

we think its sharp edges might be of interest. We give a representation of the 2D Zalesak disk at Figure 4.13.

2. The bubble is near the borders of the Ω domain, to see whether boundary conditions will interfere badly.
3. The bubble is located in a corner of the square domain, in order to avoid any help of an eventual symmetry.

The resulting initial conditions are represented on Figure 4.14 for the 2D case and on Figure 4.15 for the 3D case.

The linear solver we use for solving linear equations is the PETSc library [17, 16, 15].

4.2.3 Numerical results and verification

Figure 4.16 shows 2D results for $t = 3.12$, at $iter_t = 13$. Figure 4.17 shows 3D results for $t = 3.046$, at $iter_t = 13$. Due to the pulsation, the bubble grows and shrinks periodically. Online videos are available at `https://youtu.be/XyxfV3w88AQ` and `https://youtu.be/-V2NmaUWAJM`. We show the borders

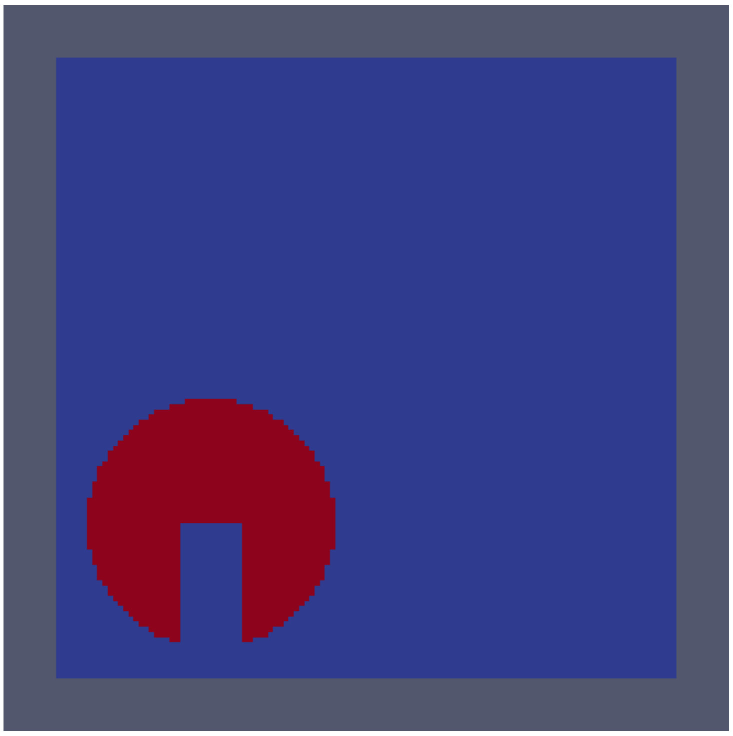


Figure 4.14 – Y field at $t = 0$, 2D case

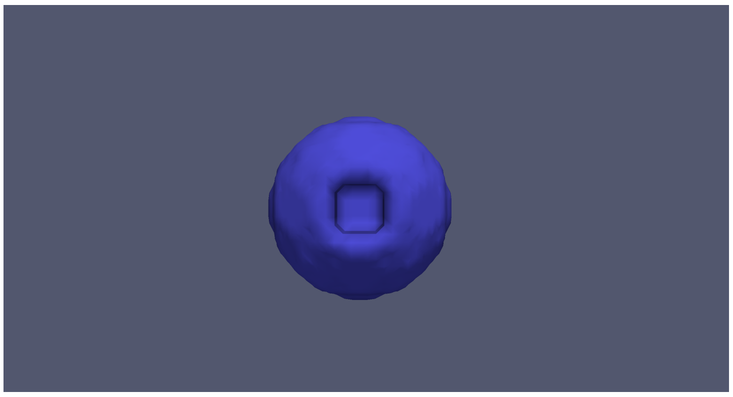


Figure 4.15 – Y field at $t = 0$, 3D case: zoom on a corner of Ω

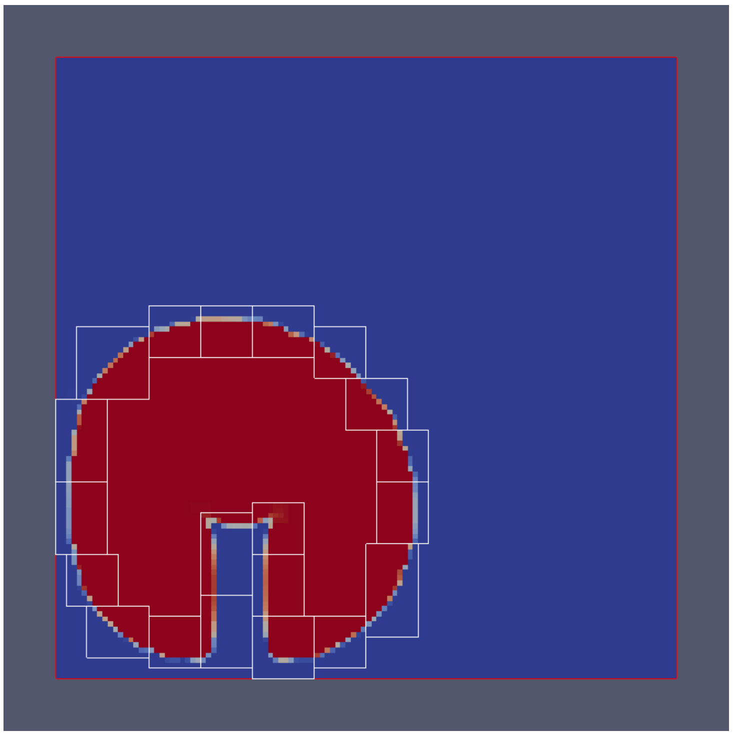


Figure 4.16 – Y field at $t = 3.12$, 2D case

of the patches with white lines.

For verification purposes, we use the formula of Equation (4.57) on the 2D simulations in a Ω space of size 1×1. This yields $|\Omega| = 1$. Let $R = 0.2$ be the radius of the Zalesak sphere, as shown on Figure 4.13. Let arccsc be the inverse function of cosecant, where cosecant is the inverse of sinus:

$$\forall x \in \mathbb{R} \setminus \pi\mathbb{Z}, \csc(x) = \frac{1}{\sin(x)}, \tag{4.59}$$

$$\forall x \in \left[\frac{-\pi}{2}, \frac{\pi}{2}\right], \operatorname{arccsc}(\csc(x)) = x. \tag{4.60}$$

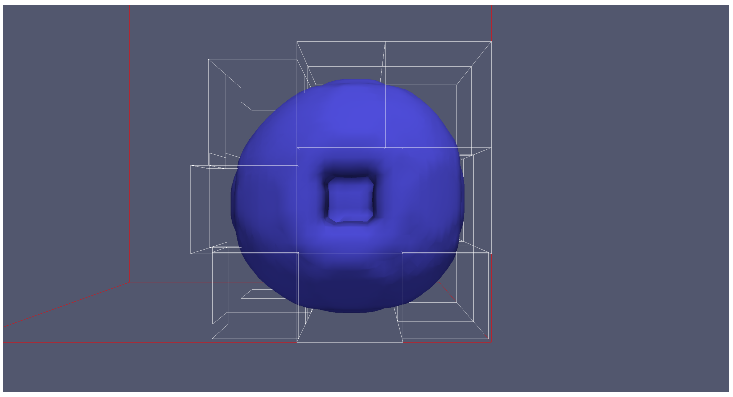


Figure 4.17 – Y field at $t = 3.046$, 3D case: zoom on a corner of Ω

Then we get the initial volume $V(0)$:

$$\begin{aligned}
V(0) &= \pi R^2 - 2\int_0^{\frac{r}{4}} \sqrt{r^2 - x^2}dx, \\
&= R^2\left(\pi - \frac{1}{16}(\sqrt{15} + 16 \text{ arccsc}(4))\right), \\
&\approx 2.64685 R^2, \\
&\approx 2.64685 \times 0.2^2, \\
&\approx 0.105874.
\end{aligned} \tag{4.61}$$

We first run the simulation on a 2D grid with no AMR, divided by 30 $\times$ 30 cells. The measurement of the computed volume is given by Figure 4.18. We then run the simulation on a 2D grid with no AMR either and this time divided by 60 $\times$ 60 cells, that-is-to-say twice as fine in every direction. Figure 4.19 shows that the computation is than – as one could have expected – closer to the theoretical solution. Finally, we run the simulation on a 30 $\times$ 30 2D grid with one level of AMR, which refinement coefficient is 2. The result shown on Figure 4.20 is very close to the one of the corresponding 60 $\times$ 60 grid. This implies that refining is an improvement compared to computation on the coarse grid only.

Finally, we measured the time needed for the computation with and without AMR, without enabling any parallelisation:

- without AMR, on a 60 $\times$ 60 grid: 1 min 2 s,

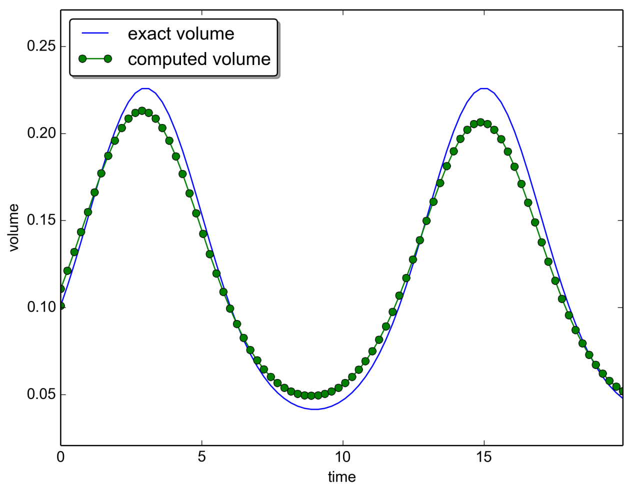


Figure 4.18 – Volume of the bubble as a function of time, for a 30 × 30 grid, without AMR

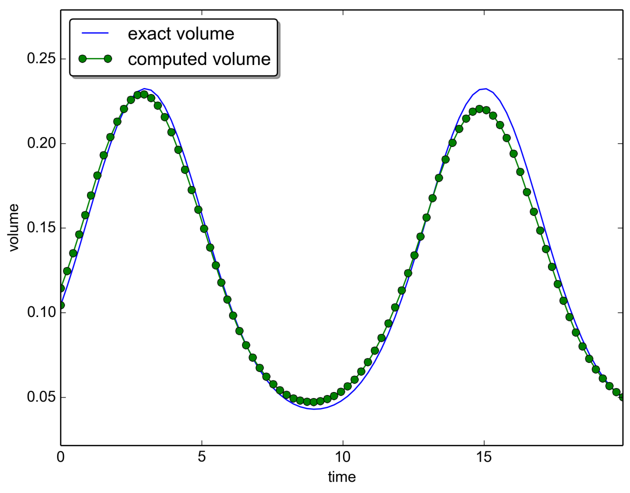


Figure 4.19 – Volume of the bubble as a function of time, for a 60 × 60 grid, without AMR

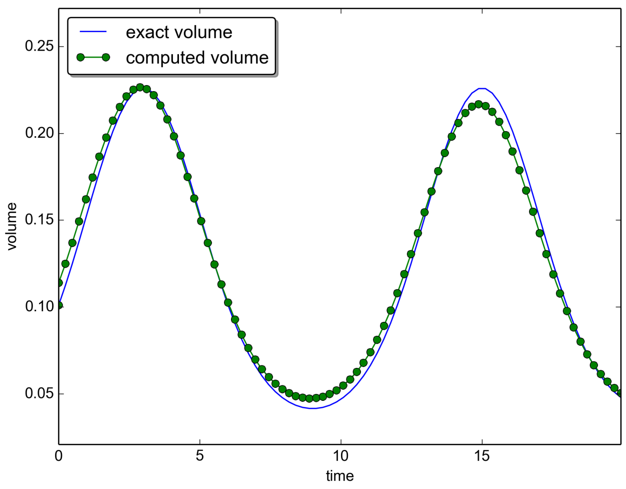


Figure 4.20 – Volume of the bubble as a function of time, for a 30 × 30 grid, with AMR refinement of 2

- with AMR, on a 30 × 30 grid and a refinement coefficient of 2: 41 s.

This means that with adaptive mesh refinement, we get a gain of 30% of computation time, for a similar precision.

4.3 Reusable implementation with CDMATH

We lay down in Chapter 3 how we proceed for AMR and we explain in Chapter 4 how we couple the resolution of elliptic and hyperbolic equations. We want to use our developments for simulations closer to reality, as shown in Chapter 5. But we also want other numericists, researchers and engineers to be able to reproduce our results and implement their own problems with a patch-based AMR. This is why we co-developed and used the CDMATH library [44, 145].

CDMATH is a new library for easy numerical simulation, developed by an eponymous workgroup; CDMATH [48, 23, 76, 49]. It is designed to be used for quick modelling or as a toolbox for students, interns and demonstration projects in numerical simulation. It takes its roots in a dual context. On the industrial side, we need 3D, complex domain (e.g. non-connected space), complex boundary conditions, hybrid meshes, rich physical models, standard data models... The consequence is that it is difficult for researchers to test new numerical methods on complex industrial codes, and those codes are not adapted for teaching. On the academic side, new numerical methods are often tested on "simple" configurations: 1D, 2D, rectangular domains, limited boundary conditions, one type of cell (e.g. either Cartesian or triangular cells)... The consequence is that research code is easy to use but often far from the industrial world applications.

The objective of CDMATH is to reduce the gap between mathematical research and the industry, as well as to provide easy industrial tools for teaching and research [81]. This is the reason why our code is open-source and available for collaboration on `https://github.com/PROJECT-CDMATH/CDMATH`.

4.3.1 Description of the CDMATH library

The prerequesites of CDMATH are the following:

- **HDF5** a set of file formats designed to store and organise large amounts of data,
- **SWIG** (optional) a tool used to connect programs or libraries written in C or C++ with scripting languages like Python,
- **CppUnit** (optional) a unit testing framework module for the C++ programming language,
- **PETSc** (optional) a suite of data structures and routines for the scalable (parallel) solution of scientific applications modeled by partial differential equations [17, 16, 15],
- **SALOME products** MEDFile, InterpKernel, MEDCoupling are bundled with CDMATH and define the data structures exchanged by codes; meshes and fields.

The architecture of the library and its dependencies was designed by Anouar Mekkas and is represented on Figure 4.21. We see that a core component is the SALOME products including MEDFile, InterpKernel, MEDLoader and MEDCoupling. SALOME is a platform for numerical simulation developed by CEA, EDF and OpenCascade [59].

CDMATH comprises a set of toolboxes, including for data, for meshes and fields, and for linear solvers.

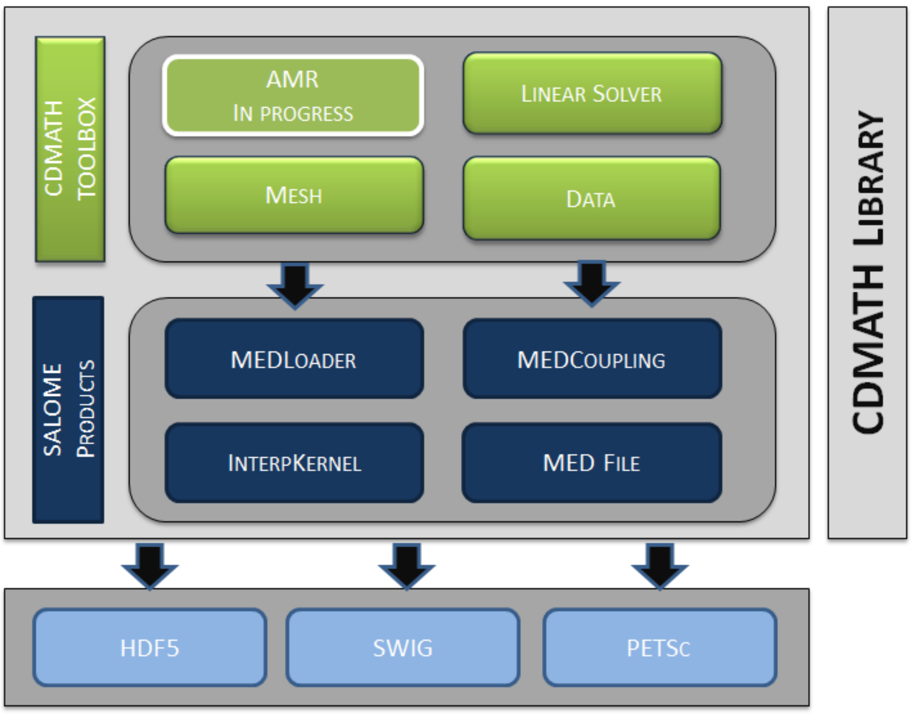


Figure 4.21 – CDMATH architecture

4.3.2 AMR, an extension of CDMATH

As a way to show how powerful and easy-to-use CDMATH is, we created a supplementary toolbox based on it. These additional tools aim at creating the numerical and development environment for Adaptive Mesh Refinement (AMR). We designed the extension such that a new user can easily benefit from AMR, just by providing a solver.

As a consequence, thanks to the help of many including Anouar Mekkas and Anthony Geay, part of our libraries were pushed back "upstream" to the SALOME platform, in particular to MEDCoupling. We show a fraction of the implementation in Appendix A.

Figure 4.22 – Logo of the CDMATH workgroup

Chapter 5

Application to incompressible Navier-Stokes

We decide to use our developments to tackle simulations closer to reality, with the incompressible Navier-Stokes model. We saw in Section 2.1.3 that the volumetric mass ρ is constant in time and uniform by pieces in space. In particular, ρ is a constant which depends only on the species on the fluid.

For the computation of both the one-phase and the two-fluid models, we use a prediction-correction scheme as introduced by Chorin [38, 39] and Temam [131, 132], which we explain further in this chapter.

Staggered grid discretisation For both models, we consider a 2D mesh of N^2 cells numbered using indices $(i, j) \in [\![0, N-1]\!]^2$. We choose to use finite differences on a staggered grid space discretisation, also known as MAC [77]. It means that scalar fields (for example the pressure p or divergences) are located at the centre of cells. The vector fields (for instance the velocity $\mathbf{u}$ or gradients) are located on the faces of the meshes. More specifically, they are located on the faces normal to the direction represented: u_x on vertical faces and u_y on horizontal faces. Staggered grid schemes are known to have good properties to avoid pressure oscillations and chequerboard modes.

Thus we define the values of u_x and u_y only on the faces normal to the represented direction. The vertical faces are numbered $(i+\frac{1}{2}, j) \in \left([\![-1, N-1]\!]+\frac{1}{2}\right) \times [\![0, N-1]\!]$. The horizontal faces are numbered $(i, j+\frac{1}{2}) \in [\![0, N-1]\!] \times \left([\![-1, N-1]\!]+\frac{1}{2}\right)$.

In total, we count $2N(N+1)$ faces. Among those, there are $2N(N-1)$

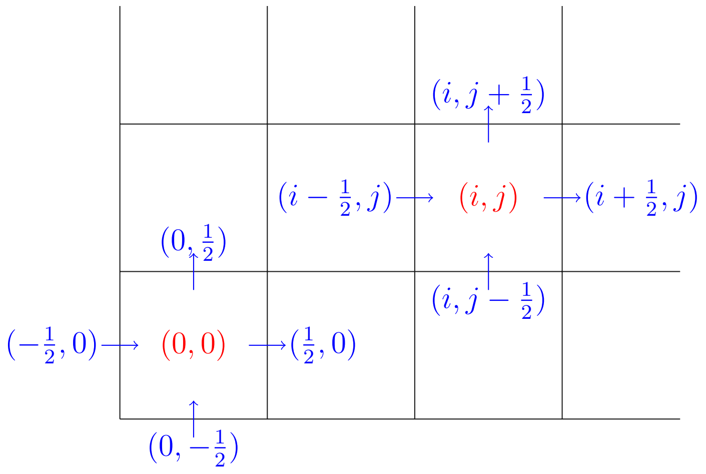


Figure 5.1 – Location of coordinates

faces not superposed with the domain boundaries, and thus for which the value of u_x and u_y is not given by the boundary conditions. The vertical interior faces are numbered $(i+\frac{1}{2}, j) \in \left(\llbracket 0, N-2 \rrbracket + \frac{1}{2}\right) \times \llbracket 0, N-1 \rrbracket$. The horizontal interior faces are numbered $(i, j+\frac{1}{2}) \in \llbracket 0, N-1 \rrbracket \times \left(\llbracket 0, N-2 \rrbracket + \frac{1}{2}\right)$. This is represented on Figure 5.1.

We decide to introduce three *ad hoc* terms. We will call "staggered vector field" a vector field which components are known on the faces orthogonal to the direction of its components. For instance, for the staggered field $\mathbf{u}$, we know u_x on vertical faces and u_y on the horizontal faces. We will call "anti-staggered vector field" a vector field which components are known on the faces parallel to the direction of its components. For instance, for the anti-staggered field $\mathbf{U}$, we know U_x on the horizontal faces and U_y on the vertical faces. Finally, we will call "dense vector field" a vector field which components are known on all faces. For instance, for the dense field $\mathbf{v}$, we know both v_x and v_y on all faces; both horizontal and vertical.

This way, with the knowledge of the staggered field $\mathbf{u}$ and the anti-staggered field $\mathbf{U}$, we can assemble the dense field $\mathbf{v}$. On the vertical faces, we choose the following components:

$$\mathbf{v} = \begin{pmatrix} u_x \\ U_y \end{pmatrix}. \tag{5.1}$$

On the horizontal faces, we choose the following components:

$$\mathbf{v} = \begin{pmatrix} U_x \\ u_y \end{pmatrix}. \tag{5.2}$$

As a last remark, as we said earlier, in the context of staggered grid schemes, for the vector fields we use staggered fields in the vast majority of cases.

5.1 One-phase incompressible Navier-Stokes

As seen before in Section 2.1.3, we solve the following set of equations:

$$\begin{cases} \partial_t \mathbf{u} + \mathbf{u} \cdot \nabla \mathbf{u} - \nu \nabla^2 \mathbf{u} = \mathbf{f} = -\frac{1}{\rho} \nabla p + \mathbf{g} + \mathbf{f}_{others}, \\ \nabla \cdot \mathbf{u} = 0, \\ \mathbf{u} = \mathbf{u}_{BC} \text{ on } \partial\Omega \text{ (boundary conditions)}, \\ \mathbf{u}(t=0) = \mathbf{u}_0 \text{ on } \Omega \text{ (initial conditions)}. \end{cases} \tag{5.3}$$

5.1.1 Prediction-correction scheme

We define a 2D computational domain with the shape of a square. Let $\widehat{\mathbf{e}}_\mathbf{x}$ and $\widehat{\mathbf{e}}_\mathbf{y}$ be the be the elementary unit vectors of the 2D Cartesian coordinate system and let $\widehat{\mathbf{e}}_\mathbf{x} \times \widehat{\mathbf{e}}_\mathbf{y} = \widehat{\mathbf{e}}_\mathbf{z}$ be the cross product in a hypothetical 3D extension. For a given border, let us call $\mathbf{n}$ the unit vector outwardly normal to the surface. For this same border, let $\mathbf{t}$ be the vector tangential to the surface, such that $\mathbf{n} \times \mathbf{t} = \widehat{\mathbf{e}}_\mathbf{z}$.

As mentioned before, we decide to use a prediction-correction scheme as done by Chorin [38, 39] and Temam [131, 132]. The model discretised in time is expressed as follows, for a time step Δt and a n^{th} time $t^n = n\Delta t$.

$$\begin{cases} \frac{\tilde{\mathbf{u}} - \mathbf{u}^n}{\Delta t} + \mathbf{u}^n \cdot \nabla \tilde{\mathbf{u}} - \nu \nabla^2 \tilde{\mathbf{u}} = \mathbf{f}^n = -\frac{1}{\rho} \nabla p^n + \mathbf{g}, \\ \tilde{\mathbf{u}} = \tilde{\mathbf{u}}_{BC} \text{ on } \partial\Omega, \\ \tilde{\mathbf{u}}_{BC} \cdot \mathbf{n} = \mathbf{u}_{BC} \cdot \mathbf{n}, \\ \tilde{\mathbf{u}}_{BC} \cdot \mathbf{t} = \mathbf{u}_{BC} \cdot \mathbf{t} + \frac{\Delta t}{\rho} \nabla \phi^n \cdot \mathbf{t}. \end{cases} \tag{5.4}$$

$$\begin{cases} -\frac{1}{\rho} \Delta \phi^{n+1} = -\frac{1}{\Delta t} \nabla \cdot \tilde{\mathbf{u}}, \\ \nabla \phi^{n+1} \cdot \mathbf{n} = 0 \text{ on } \partial\Omega. \end{cases} \tag{5.5}$$

$$p^{n+1} = p^n + \phi^{n+1}. \tag{5.6}$$

$$\begin{cases} \dfrac{\mathbf{u}^{n+1} - \tilde{\mathbf{u}}}{\Delta t} = -\dfrac{1}{\rho}\nabla \phi^{n+1}, \\ \\ \mathbf{u}^{n+1} = \mathbf{u}_{BC} \text{ on } \partial\Omega. \end{cases} \tag{5.7}$$

The unknown $\tilde{\mathbf{u}}$ is the predicted velocity, computed at every time step. The unknown ϕ is the increment in pressure for the n^{th} time step. It permits to compute the pressure as well as to give a correction to the predicted velocity, so as to obtain the corrected velocity $\mathbf{u}$. The velocity $\mathbf{u}$ is $\tilde{\mathbf{u}}$ projected on the set of zero-divergence velocities.

Theorem 8. *The semi-discretised scheme (5.4), (5.5), (5.6), (5.7) yields a zero-divergence velocity, for which the no-slip boundary conditions are ensured.*

Proof.

$$\mathbf{u}^{n+1} = \tilde{\mathbf{u}} - \frac{\Delta t}{\rho}\nabla \phi^{n+1}, \tag{5.8}$$

$$\begin{aligned} \nabla \cdot \mathbf{u}^{n+1} &= \nabla \cdot \tilde{\mathbf{u}} - \frac{\Delta t}{\rho}\Delta \phi^{n+1}, \\ &= \nabla \cdot \tilde{\mathbf{u}} - \frac{\Delta t}{\Delta t}\nabla \cdot \tilde{\mathbf{u}}, \\ &= 0. \end{aligned} \tag{5.9}$$

So the velocity field $\mathbf{u}^{n+1}$ has zero divergence. Furthermore, we can compute the no-slip boundary conditions of $\mathbf{u}^{n+1}$:

$$\begin{cases} \mathbf{u}^{n+1} \cdot \mathbf{n} = \tilde{\mathbf{u}} \cdot \mathbf{n} - \dfrac{\Delta t}{\rho}\nabla \phi^{n+1} \cdot \mathbf{n} = \mathbf{u}_{BC} \cdot \mathbf{n}, \\ \\ \mathbf{u}^{n+1} \cdot \mathbf{t} = \tilde{\mathbf{u}} \cdot \mathbf{t} - \dfrac{\Delta t}{\rho}\nabla \phi^{n+1} \cdot \mathbf{t} = \mathbf{u}_{BC} \cdot \mathbf{t}. \end{cases} \tag{5.10}$$

□

Finally, in spite of the semi-implicitness, we can lay out what the resulting scheme looks like:

$$
\begin{aligned}
\mathbf{u}^{n+1} &= \tilde{\mathbf{u}} - \frac{\Delta t}{\rho}\nabla\phi^{n+1}, \\
&= \mathbf{u}^n - \Delta t\ \mathbf{u}^n \cdot \nabla\tilde{\mathbf{u}} + \Delta t\ \nu\nabla^2\tilde{\mathbf{u}} + \Delta t\ \mathbf{f}^n - \frac{\Delta t}{\rho}\nabla\phi^{n+1}, \\
&= \mathbf{u}^n - \Delta t\ \mathbf{u}^n \cdot \nabla\tilde{\mathbf{u}} + \Delta t\ \nu\nabla^2\tilde{\mathbf{u}} + \Delta t\left(-\frac{1}{\rho}\nabla p^n + \mathbf{g}\right) \\
&\qquad -\frac{\Delta t}{\rho}\nabla\left(p^{n+1} - p^n\right), \\
&= \mathbf{u}^n - \Delta t\ \mathbf{u}^n \cdot \nabla\tilde{\mathbf{u}} + \Delta t\ \nu\nabla^2\tilde{\mathbf{u}} + \Delta t\ \mathbf{f}^{n+1}.
\end{aligned}
\tag{5.11}
$$

This yields the following expression:

$$
\begin{cases}
\dfrac{\mathbf{u}^{n+1} - \mathbf{u}^n}{\Delta t} + \mathbf{u}^n \cdot \nabla\tilde{\mathbf{u}} - \nu\nabla^2\tilde{\mathbf{u}} = \mathbf{g} - \dfrac{1}{\rho}\nabla p^{n+1}, \\
\nabla \cdot \mathbf{u}^{n+1} = 0.
\end{cases}
\tag{5.12}
$$

5.1.2 Discretisation and resolution of the prediction equation (5.4)

We recall that we work with $\mathbf{u}$ a staggered field having two components u_x and u_y in the x and y dimensions. The convection-diffusion term of Equation (5.4) is given by the following:

$$
\mathbf{u}^n \cdot \nabla\tilde{\mathbf{u}} = \begin{pmatrix} u_x^n \dfrac{\partial \tilde{u}_x}{\partial x} + u_y^n \dfrac{\partial \tilde{u}_x}{\partial y} \\ u_x^n \dfrac{\partial \tilde{u}_y}{\partial x} + u_y^n \dfrac{\partial \tilde{u}_y}{\partial y} \end{pmatrix}.
\tag{5.13}
$$

The viscosity term of Equation (5.4) is given as follows:

$$
-\nu\nabla^2\tilde{\mathbf{u}} = -\nu \begin{pmatrix} \nabla^2\tilde{u}_x \\ \nabla^2\tilde{u}_y \end{pmatrix} = -\nu \begin{pmatrix} \dfrac{\partial^2\tilde{u}_x}{\partial x^2} + \dfrac{\partial^2\tilde{u}_x}{\partial y^2} \\ \dfrac{\partial^2\tilde{u}_y}{\partial x^2} + \dfrac{\partial^2\tilde{u}_y}{\partial y^2} \end{pmatrix}.
\tag{5.14}
$$

Using these expressions we can rewrite Equation (5.4) for each component, noted z, where z is to be replaced by x or y:

$$
\frac{\tilde{u}_z - u_z^n}{\Delta t} + u_x^n \frac{\partial \tilde{u}_z}{\partial x} + u_y^n \frac{\partial \tilde{u}_z}{\partial y} - \nu\left(\frac{\partial^2\tilde{u}_z}{\partial x^2} + \frac{\partial^2\tilde{u}_z}{\partial y^2}\right) = f_z^n.
\tag{5.15}
$$

As a consequence, we can solve $\tilde{\mathbf{u}}$ component after component, independently one from another. In what follows, because the directions x and y play symmetrical roles, we may give expressions for the x direction only. Remember that this holds true because our computational domain Ω is square and parallel to the x and y directions. The boundary conditions do not couple the directions.

Upwind scheme

As explained before, we choose to use a staggered grid scheme on a 2D mesh of N^2 cells numbered using indices $(i,j) \in [\![0, N-1]\!]^2$. Here, it means that the scalar fields p, ϕ, divergences and Laplacians are located at the centre of cells. The vector fields $\mathbf{u}$, $\tilde{\mathbf{u}}$, $\mathbf{g}$ and gradients are located on the faces of the meshes.

We choose to use an upwind scheme for advected quantities. This means that if $u_x^n(i+\frac{1}{2},j) \geq 0$, then we choose the following equality:

$$\left(u_x^n \frac{\partial \tilde{u}_z}{\partial x}\right)(i+\frac{1}{2},j) = u_x^n(i+\frac{1}{2},j).\frac{\tilde{u}_z(i+\frac{1}{2},j) - \tilde{u}_z(i-\frac{1}{2},j)}{\Delta x}. \tag{5.16}$$

Let us remember that the $\tilde{u}_x$ and $\tilde{u}_y$ fields are defined on the faces perpendicular to their direction, since $\tilde{u}$ is a staggered vector field. Let us define the anti-staggered vector field U_y^n located on the faces parallel to y with the following formula:

$$\begin{aligned} U_y^n(i+\tfrac{1}{2},j) = \tfrac{1}{4} \quad (\quad & u_y^n(i,j+\tfrac{1}{2}) \; + \; u_y^n(i+1,j+\tfrac{1}{2}) \\ + \; & u_y^n(i,j-\tfrac{1}{2}) \; + \; u_y^n(i+1,j-\tfrac{1}{2}) \quad). \end{aligned} \tag{5.17}$$

Similarly, for the x direction on horizontal faces, we define U_x^n as follows:

$$\begin{aligned} U_x^n(i,j+\tfrac{1}{2}) = \tfrac{1}{4} \quad (\quad & u_x^n(i-\tfrac{1}{2},j+1) \; + \; u_x^n(i+\tfrac{1}{2},j+1) \\ + \; & u_x^n(i-\tfrac{1}{2},j) \; + \; u_x^n(i+\tfrac{1}{2},j) \quad). \end{aligned} \tag{5.18}$$

On the boundaries, when $i \in \{-1, N\}$ and when $j \in \{-1, N\}$, we use a linear interpolation from known values (i.e. a barycentre with positive and negative weights). For instance for $i = -1$,

$$U_x^n(-\frac{1}{2},j) = 2U_x^n(\frac{1}{2},j) - U_x^n(\frac{3}{2},j). \tag{5.19}$$

As $\mathbf{u}$ is the velocity field located on the faces perpendicular to the direction of its component, $\mathbf{U}$ is the velocity field located on the faces parallel to the direction of its components. We will call the operation of computing **U** thanks to **u** the "densification" of **u**.

Now let us assume $u_x^n(i+\frac{1}{2},j) \geq 0$ and $U_y^n(i+\frac{1}{2},j) \geq 0$. Using Equation (5.16), we discretise the prediction Equation (5.4) as follows, for $i \in [\![0, N-2]\!]$ and $j \in [\![1, N-2]\!]$:

$$\begin{aligned}
&\frac{\tilde{u}_x(i+\frac{1}{2},j) - u_x^n(i+\frac{1}{2},j)}{\Delta t} + u_x^n(i+\frac{1}{2},j)\frac{\tilde{u}_x(i+\frac{1}{2},j) - \tilde{u}_x(i-\frac{1}{2},j)}{\Delta x} \\
&\quad + U_y^n(i+\frac{1}{2},j)\frac{\tilde{u}_x(i+\frac{1}{2},j) - \tilde{u}_x(i+\frac{1}{2},j-1)}{\Delta y} \\
&\quad + \nu\frac{-\tilde{u}_x(i-\frac{1}{2},j) + 2\tilde{u}_x(i+\frac{1}{2},j) - \tilde{u}_x(i+\frac{3}{2},j)}{\Delta x^2} \\
&\quad + \nu\frac{-\tilde{u}_x(i+\frac{1}{2},j-1) + 2\tilde{u}_x(i+\frac{1}{2},j) - \tilde{u}_x(i+\frac{1}{2},j+1)}{\Delta y^2} \\
&\quad = -\frac{1}{\rho}\frac{p^n(i+1,j) - p^n(i,j)}{\Delta x} + g_x^n(i+\frac{1}{2},j), \\
&\quad = f_x^n(i+\frac{1}{2},j).
\end{aligned} \tag{5.20}$$

Matrix form of (5.20)

For each component $z \in \{x, y\}$, we can lay down the matrix form:

$$A_z^n \tilde{u}_z = b_z^n \tag{5.21}$$

where A_z^n is a square matrix of $N(N-1)$ rows and $N(N-1)$ columns. Let us write $ij = i + j(N-1)$ where $i \in [\![0, N-2]\!]$ and $j \in [\![0, N-1]\!]$. Then $ij \in [\![0, N(N-1)-1]\!]$. Here are the coefficients of A_x^n on the line ij, for $i \notin \{0, N-2\}$ and $j \notin \{0, N-1\}$:

- for $\tilde{u}_x(i+\frac{1}{2},j)$:

$$A(ij, ij) = 1 + u_x^n(i+\frac{1}{2})\frac{\Delta t}{\Delta x} + U_y^n(i+\frac{1}{2},j)\frac{\Delta t}{\Delta y} + 2\nu\left(\frac{\Delta t}{\Delta x^2} + \frac{\Delta t}{\Delta y^2}\right), \tag{5.22}$$

- for $\tilde{u}_x(i-\frac{1}{2},j)$:

$$A(ij, ij-1) = -u_x^n(i+\frac{1}{2},j)\frac{\Delta t}{\Delta x} - \nu\frac{\Delta t}{\Delta x^2}, \tag{5.23}$$

- for $\tilde{u}_x(i+\frac{3}{2},j)$:

$$A(ij,ij+1) = -\nu\frac{\Delta t}{\Delta x^2}, \tag{5.24}$$

- for $\tilde{u}_x(i+\frac{1}{2},j-1)$:

$$A(ij,ij-(N-1)) = -U_y^n(i+\frac{1}{2},j)\frac{\Delta t}{\Delta y} - \nu\frac{\Delta t}{\Delta y^2}, \tag{5.25}$$

- for $\tilde{u}_x(i+\frac{1}{2},j+1)$:

$$A(ij,ij+(N-1)) = -\nu\frac{\Delta t}{\Delta y^2}. \tag{5.26}$$

In the general case, u_x^n may have any value and has not necessarily positive components. So we can write an upwind advection scheme in a more general way. Here are the coefficients of A_x^n on line ij for $i \notin \{0, N-2\}$ and $j \notin \{0, N-1\}$:

- for $\tilde{u}_x(i+\frac{1}{2},j)$:

$$A(ij,ij) = 1 + |u_x^n(i+\frac{1}{2})|\frac{\Delta t}{\Delta x} + |U_y^n(i+\frac{1}{2},j)|\frac{\Delta t}{\Delta y} + 2\nu\left(\frac{\Delta t}{\Delta x^2} + \frac{\Delta t}{\Delta y^2}\right), \tag{5.27}$$

- for $\tilde{u}_x(i-\frac{1}{2},j)$:

$$A(ij,ij-1) = -\max(0, u_x^n(i+\frac{1}{2},j))\frac{\Delta t}{\Delta x} - \nu\frac{\Delta t}{\Delta x^2}, \tag{5.28}$$

- for $\tilde{u}_x(i+\frac{3}{2},j)$:

$$A(ij,ij+1) = \min(0, u_x^n(i+\frac{1}{2},j))\frac{\Delta t}{\Delta x} - \nu\frac{\Delta t}{\Delta x^2}, \tag{5.29}$$

- for $\tilde{u}_x(i+\frac{1}{2},j-1)$:

$$A(ij,ij-(N-1)) = -\max(0, U_y^n(i+\frac{1}{2},j))\frac{\Delta t}{\Delta y} - \nu\frac{\Delta t}{\Delta y^2}, \tag{5.30}$$

- for $\tilde{u}_x(i+\frac{1}{2},j+1)$:

$$A(ij,ij+(N-1)) = \min(0, U_y^n(i+\frac{1}{2},j))\frac{\Delta t}{\Delta y} - \nu\frac{\Delta t}{\Delta y^2}. \tag{5.31}$$

Boundary conditions expression in the right-hand side

When we are located away from the borders, that-is-to-say when $i \notin \{0, N-2\}$ and $j \notin \{0, N-1\}$, then the right-hand side b_x is given by a simple expression:

$$b_x(i+\frac{1}{2},j) = \Delta t\ f_x^n(i+\frac{1}{2},j) + u_x^n(i+\frac{1}{2},j). \tag{5.32}$$

As the domain is square, let us refer to the borders as on the West, South, East and North.

On the West border ($x = 0$ and $i = 0$)

$$\begin{aligned} f_x^n(\frac{1}{2},j) = & \frac{\tilde{u}_x(\frac{1}{2},j) - u_x^n(\frac{1}{2},j)}{\Delta t} \\ & + \max(0, u_x^n(\frac{1}{2},j))\frac{\tilde{u}_x(\frac{1}{2},j) - \tilde{u}_{BCx}(-\frac{1}{2},j)}{\Delta x} \\ & + U_y^n(\frac{1}{2},j)\frac{\tilde{u}_x(\frac{1}{2},j) - \tilde{u}_x(\frac{1}{2},j-1)}{\Delta y} \\ & + \nu\frac{-\tilde{u}_{BCx}(-\frac{1}{2},j) + 2\tilde{u}_x(\frac{1}{2},j) - \tilde{u}_x(\frac{3}{2},j)}{\Delta x^2} \\ & + \nu\frac{-\tilde{u}_x(\frac{1}{2},j-1) + 2\tilde{u}_x(\frac{1}{2},j) - \tilde{u}_x(\frac{1}{2},j+1)}{\Delta y^2}. \end{aligned} \tag{5.33}$$

This yields

$$\begin{aligned} b_x^n(\frac{1}{2},j) = & \Delta t\ f_x^n(\frac{1}{2},j) \\ & + \frac{\Delta t}{\Delta x}\max(0, u_x^n(\frac{1}{2},j))\tilde{u}_{BCx}(-\frac{1}{2},j) \\ & + \nu\frac{\Delta t}{\Delta x^2}\tilde{u}_{BCx}(-\frac{1}{2},j) + u_x^n(\frac{1}{2},j). \end{aligned} \tag{5.34}$$

On the South border ($y = 0$ and $j = 0$)

$$\begin{aligned} b_x^n(i+\frac{1}{2},0) = \Delta t\ f_x^n(i+\frac{1}{2},0) \\ +\frac{\Delta t}{\Delta x}\max(0,U_y^n(i+\frac{1}{2},0))\tilde{u}_{BCx}(i+\frac{1}{2},-1) \\ +\nu\frac{\Delta t}{\Delta y^2}\tilde{u}_{BCx}(i+\frac{1}{2},-1)+u_x^n(i+\frac{1}{2},0). \end{aligned} \tag{5.35}$$

On the East border ($x = x_{max}$ and $i = N-1$)

$$\begin{aligned} b_x^n(N-2+\frac{1}{2},j) = \Delta t\ f_x^n(N-\frac{3}{2},j) \\ -\frac{\Delta t}{\Delta x}\min(0,u_x^n(N-\frac{3}{2},j))\tilde{u}_{BCx}(N-\frac{1}{2},j) \\ +\nu\frac{\Delta t}{\Delta x^2}\tilde{u}_{BCx}(N-\frac{1}{2},j)+u_x^n(N-\frac{3}{2},j). \end{aligned} \tag{5.36}$$

On the North border ($y = y_{max}$ and $j = N-1$)

$$\begin{aligned} b_x^n(i+\frac{1}{2},N-1) = \Delta t\ f_x^n(i+\frac{1}{2},N-1) \\ -\frac{\Delta t}{\Delta x}\min(0,U_y^n(i+\frac{1}{2},N-1))\tilde{u}_{BCx}(i+\frac{1}{2},N) \\ +\nu\frac{\Delta t}{\Delta y^2}\tilde{u}_{BCx}(i+\frac{1}{2},N)+u_x^n(i+\frac{1}{2},N-1). \end{aligned} \tag{5.37}$$

$\tilde{\mathbf{u}}_{BC}(i+\frac{1}{2},j)$ on West and East borders As $\tilde{\mathbf{u}}_{BC}.\mathbf{n} = \mathbf{u}_{BC}^n\cdot\mathbf{n}$, we get

$$\begin{cases} \tilde{u}_{BCx}(-\frac{1}{2},j) = \mathbf{u}_{BC}^n(-\frac{1}{2},j)\cdot\mathbf{n}, \\ \tilde{u}_{BCx}(N-\frac{1}{2},j) = \mathbf{u}_{BC}^n(N-\frac{1}{2},j)\cdot\mathbf{n}. \end{cases} \tag{5.38}$$

And as $\tilde{\mathbf{u}}_{BC} \cdot \mathbf{t} = \mathbf{u}^n_{BC} \cdot \mathbf{t} + \frac{\Delta t}{\rho} \nabla\phi.\mathbf{t}$, we get

$$\tilde{u}_{BCy}(-\frac{1}{2}, j) = \mathbf{u}^n_{BC}(-\frac{1}{2}, j) \cdot \mathbf{t} + \frac{\Delta t}{\rho} \left(\frac{\partial \phi}{\partial y}\right)(-\frac{1}{2}, j). \tag{5.39}$$

We thus have to compute $\left(\frac{\partial \phi}{\partial y}\right)$ on borders. We do it by linear interpolation (barycentric centres with positive and eventually negative weights):

$$\begin{aligned}
\left(\frac{\partial \phi}{\partial y}\right)(-\frac{1}{2}, j) &= \frac{\phi(-\frac{1}{2}, j+1) - \phi(-\frac{1}{2}, j-1)}{2\Delta y}, \\
&= \frac{1}{2\Delta y}\left(\frac{3}{2}\phi(0, j+1) - \frac{1}{2}\phi(1, j+1)\right) \\
&\qquad - \frac{1}{2\Delta y}\left(\frac{3}{2}\phi(0, j-1) - \frac{1}{2}\phi(1, j-1)\right), \\
&= \frac{1}{4\Delta y}\left(3\phi(0, j+1) - \phi(1, j+1)\right. \\
&\qquad \left. -3\phi(0, j-1) + \phi(1, j-1)\right).
\end{aligned} \tag{5.40}$$

In the corners of the domain, we are not able to apply the above mentioned formula. Hence we choose the following expression of linear interpolation:

$$\left(\frac{\partial \phi}{\partial y}\right)(-\frac{1}{2}, 0) = 2\left(\frac{\partial \phi}{\partial y}\right)(-\frac{1}{2}, 1) - \left(\frac{\partial \phi}{\partial y}\right)(-\frac{1}{2}, 2). \tag{5.41}$$

Value of $\tilde{u}_{BCx}(i + \frac{1}{2}, j)$ beyond the borders South and North According to our discretisation, we need a value for $\tilde{u}_{BCx}(i + \frac{1}{2}, -1)$, which is located below the South border (numbered $j = -\frac{1}{2}$). In theory, we could say it has little physical meaning. Though we give it the following value by linear interpolation:

$$\tilde{u}_{BCx}(i + \frac{1}{2}, -1) = 2\tilde{u}_{BCx}(i + \frac{1}{2}, -\frac{1}{2}) - u^n_x(i + \frac{1}{2}, 0). \tag{5.42}$$

5.1.3 Discretisation and resolution of the correction equations (5.5), (5.6) and (5.7)

Discretisation and resolution of (5.5)

We define the scalar d as $d = -\rho \nabla \cdot \tilde{\mathbf{u}}$. We rewrite the equation giving the increment of pressure ϕ as follows:

$$\begin{cases} -\Delta \phi^{n+1} = -\dfrac{\rho}{\Delta t} \nabla \cdot \tilde{\mathbf{u}} = \dfrac{d}{\Delta t}, \\ \\ \nabla \phi^{n+1} \cdot \mathbf{n} = 0 \text{ on } \partial\Omega. \end{cases} \tag{5.43}$$

The scalar ϕ is located at the centre of cells, so there are N^2 unknowns. We use the well-known five-point stencil for the Laplacian. For $(i,j) \in [\![1, N-2]\!]$, this leads to the following equality:

$$\begin{aligned} &\frac{-\phi(i-1,j) + 2\phi(i,j) - \phi(i+1,j)}{\Delta x^2} + \frac{-\phi(i,j-1) + 2\phi(i,j) - \phi(i,j+1)}{\Delta y^2} \\ &= -\frac{\rho(i,j)}{\Delta t} \left(\frac{\tilde{u}_x(i+\frac{1}{2},j) - \tilde{u}_x(i-\frac{1}{2},j)}{\Delta x} + \frac{\tilde{u}_y(i,j+\frac{1}{2}) - \tilde{u}_y(i,j-\frac{1}{2})}{\Delta y} \right), \\ &= \frac{d(i,j)}{\Delta t}. \end{aligned} \tag{5.44}$$

So we solve a system $C\phi = d$ where $C \in \mathbb{M}(N^2)$, where $\mathbb{M}(N^2)$ is the set of square matrices with N^2 rows and N^2 columns. Again, let us note $ij = i + jN$, for $(i,j) \in [\![1, N-2]\!]$:

$$C(ij,ij) = 2\left(\frac{\Delta t}{\Delta x^2} + \frac{\Delta t}{\Delta y^2}\right). \tag{5.45}$$

$$C(ij,ij-1) = C(ij,ij+1) = -\frac{\Delta t}{\Delta x^2}. \tag{5.46}$$

$$C(ij,ij-N) = C(ij,ij+N) = -\frac{\Delta t}{\Delta y^2}. \tag{5.47}$$

For the Neumann boundary conditions, we impose $\phi(-1,j) = \phi(0,j)$. Hence we get the following expressions for components of C when $i \in \{0, N-1\}$ or $j \in \{0, N-1\}$:

$$C(0+jN, 0+jN) = \frac{\Delta t}{\Delta x^2} + 2\frac{\Delta t}{\Delta y^2}, \tag{5.48}$$

$$C(N-1+jN, N-1+jN) = \frac{\Delta t}{\Delta x^2} + 2\frac{\Delta t}{\Delta y^2}, \tag{5.49}$$

$$C(i+0N, i+0N) = 2\frac{\Delta t}{\Delta x^2} + \frac{\Delta t}{\Delta y^2}, \tag{5.50}$$

$$C(i+(N-1)N, i+(N-1)N) = 2\frac{\Delta t}{\Delta x^2} + \frac{\Delta t}{\Delta y^2}. \tag{5.51}$$

Discretisation and resolution of (5.7)

We set Equation (5.7) as follows:

$$\frac{\mathbf{u}^{n+1} - \tilde{\mathbf{u}}}{\Delta t} = -\frac{1}{\rho}\nabla\phi^{n-1}. \tag{5.52}$$

We use $\nabla\phi_x^{n+1}(i+\frac{1}{2}, j) = \frac{\phi(i+1,j)-\phi(i,j)}{\Delta x}$ and $\nabla\phi_x^{n+1}(-\frac{1}{2}, j) = \nabla\phi_x^{n+1}(N-\frac{1}{2}, j) = 0$. Then we get a simple arithmetic equation to compute $\mathbf{u}^{n+1}$:

$$u_x^{n+1}(i+\frac{1}{2}, j) = \tilde{u}_x(i+\frac{1}{2}, j) - \frac{\Delta t}{\rho}\nabla\phi_x^{n+1}(i+\frac{1}{2}, j) \tag{5.53}$$

for $(i,j) \in [\![0, N-1]\!]^2$.

We can verify that the divergence of $\mathbf{u}$ is zero at time step $n+1$, according to the following development:

$$
\begin{aligned}
&(\nabla\cdot\mathbf{u}^{n+1})(i,j)\\
&\quad= \frac{u_x^{n+1}(i+\frac{1}{2},j)-u_x^{n+1}(i-\frac{1}{2},j)}{\Delta x}+\frac{u_y^{n+1}(i,j+\frac{1}{2})-u_y^{n+1}(i,j-\frac{1}{2})}{\Delta y},\\
&\quad= \frac{\tilde{u}_x^{n+1}(i+\frac{1}{2},j)-\tilde{u}_x^{n+1}(i-\frac{1}{2},j)}{\Delta x}\\
&\qquad\qquad -\frac{\Delta t}{\rho}\frac{\nabla\phi_x^{n+1}(i+\frac{1}{2},j)-\nabla\phi_x^{n+1}(i-\frac{1}{2},j)}{\Delta x}+(\ldots)_y\,,\\
&\quad= (\nabla\cdot\tilde{\mathbf{u}})(i,j)-\frac{\Delta t}{\rho}\frac{\phi(i+1,j)-2\phi(i,j)+\phi(i-1,j)}{\Delta x^2}+(\ldots)_y\,,\\
&\quad= (\nabla\cdot\tilde{\mathbf{u}})(i,j)-\frac{\Delta t}{\rho}\Delta\phi^{n+1}(i,j),\\
&\quad= 0.
\end{aligned}
\tag{5.54}
$$

Summary of time resolution

Let $\mathbf{v}$ be a dense vector field defined on the faces of the mesh. Let us call "computation of $\mathbf{v}$ on perpendicular faces" the computation of the value of v_x on vertical faces and of v_y on horizontal faces, in order to obtain a staggered vector field. Let us call "computation of $\mathbf{v}$ on parallel faces" the computation of the value of v_x on horizontal faces and of v_y on vertical faces, in order to obtain an anti-staggered vector field. For instance, once we know $\mathbf{v}$ on perpendicular faces, we can densify $\mathbf{v}$ to get its value on parallel faces. As before, let us call Ω^H the computational mesh, with its cells and faces. This said, we can lay down the necessary steps for the computation of time step $n+1$ knowing the value of the variables at time step n:

1. compute $\nabla\phi\cdot\mathbf{t}$ on parallel faces of $(\partial\Omega)^H$,
2. compute $\tilde{\mathbf{u}}_{BC}$ on parallel and perpendicular faces of $(\partial\Omega)^H$,
3. compute ∇p^n on perpendicular faces of $(\Omega\setminus\partial\Omega)^H$,
4. densify ∇p^n on parallel faces of $(\Omega\setminus\partial\Omega)^H$,

5. compute $\mathbf{f}^n$ on perpendicular faces of $(\Omega \setminus \partial\Omega)^H$,
6. compute b_z^n,
7. compute A_z^n,
8. solve $A_z^n \tilde{u}_z = b_z^n$ and get staggered field $\tilde{u}$,
9. compute d^n on cells of Ω^H,
10. compute C^n,
11. solve $C^n \phi^{n+1} = d^n$,
12. compute $\nabla\phi^{n+1}$ on perpendicular faces of $(\Omega \setminus \partial\Omega)^H$,
13. compute $\mathbf{u}^{n+1}$ on perpendicular faces of $(\Omega \setminus \partial\Omega)^H$,
14. densify $\mathbf{u}^{n+1}$ on parallel faces of $(\Omega \setminus \partial\Omega)^H$,
15. compute $p^{n+1} = p^n + \phi^{n+1}$.

5.1.4 Adaptive Mesh Refinement

Computation of coarse and fine levels

We decide to use only one level of refinement for the AMR. We make the decision to focus the AMR on the pressure variations ϕ. The reason is that this is what will be important later when we study two-fluid flows: see Section 5.2.1.

Equation (5.4) gives the predicted velocity $\tilde{\mathbf{u}}$. It is solved by resolution of the linear system (5.21) on the coarse level. For the fine level, we decide to compute the predicted velocity $\tilde{\mathbf{u}}$ from a linear interpolation $\mathbb{I}^h$ of the coarse level onto the fine level. This fine $\tilde{\mathbf{u}}$ is used for the fine version of Equation (5.7).

From $\tilde{\mathbf{u}}$ on the coarse grid, we also compute a coarse $\frac{d}{\rho} = \nabla \cdot \tilde{\mathbf{u}}$. This variable is linearly interpolated to get $\nabla \cdot \tilde{\mathbf{u}}$ on the fine grid. The divergence of the predicted velocity is used for the source term of Equation (5.5). Notice that the linear interpolation and the divergence operators do not commute, as explained in Section 5.1.4.

Equation (5.5) gives the increment of pressure ϕ. We fully use the AMR possibilities here by computing it using the LDC algorithm explained in

Section 4.1. This gives an accurate result on the coarse as well as on the fine level.

Equation (5.6) is just an addition, be it on the coarse or on the fine level.

Equation (5.7) is also mostly an addition. So on each level it is computed with the data associated to that level.

Linear interpolation formulas

We think it is important to explain how we proceed for the linear interpolation $\mathbb{I}^h$, for instance used after the coarse resolution of (5.20) to get the approximation of $\tilde{\mathbf{u}}$ on the fine grid. We here give an explanation in 1D, where the coarse grid Ω^H follows indices I (capital letters) and the fine grid $\Omega^h_\square$ follows indices i (small letters). The refinement coefficient is $r \in \mathbb{N}^*$. We focus on the coarse cell numbered I. We set $i = r \times I$. So the coarse cell I contains the fine cells i, $i+1$, …, $i+r-1$.

Let ϕ be a scalar field, located at the centre of cells: we note $\Phi(I)$ its coarse value on the coarse cell I and $\phi(i)$ its fine value on the fine cell i. Let k be one of the r integers such that $i \leq i+k \leq i+r-1$. If $\frac{1}{2r} + k\frac{1}{r} \leq \frac{r}{2}$, then cell $i+k$ is located in the left half of cell I and

$$\phi(i+k) = \frac{r-1-2k}{2r}\Phi(I-1) + \frac{r+1+2k}{2r}\Phi(I). \tag{5.55}$$

Similarly, if $\frac{1}{2r} + k\frac{1}{r} \geq \frac{r}{2}$, then cell $i+k$ is located in the right half of cell I and

$$\phi(i+k) = \frac{3r-2k-1}{2r}\Phi(I) + \frac{2k-r-1}{2r}\Phi(I+1). \tag{5.56}$$

Let $\mathbf{u}$ be a vector field, located on the faces: we note $\mathbf{U}(I+\frac{1}{2})$ its coarse value on the coarse face $I+\frac{1}{2}$ and $\mathbf{u}(i+\frac{1}{2})$ its fine value on the fine face $i+\frac{1}{2}$. Let k be one of the r integers such that $i+\frac{1}{2} \leq i-\frac{1}{2}+k \leq i-\frac{1}{2}+r$. In all cases, we are in between face $I-\frac{1}{2}$ and face $I+\frac{1}{2}$.

$$\mathbf{u}(i+k+\frac{1}{2}) = \left(1-\frac{k}{r}\right)\mathbf{U}(I-\frac{1}{2}) + \frac{k}{r}\mathbf{U}(I+\frac{1}{2}). \tag{5.57}$$

Source term on fine level

Let us note $\frac{D}{\rho}$ the divergence present in the source term of Equation (5.5) on the coarse level and $\frac{d}{\rho}$ the divergence in the source term on the fine level.

We saw earlier that we made the implementation choice that $\frac{d}{\rho}$ be the linear interpolation of $\frac{D}{\rho}$. In this section, we justify this choice.

Theorem 9. *For the definition of the divergence $\frac{d}{\rho}$ in the source term of Equation (5.5) on the fine level, it is not equivalent to choose the linear interpolation of $\frac{D}{\rho}$ or the discreet divergence of the velocity $\mathbf{u}$ on the fine level. The linear interpolation of $\frac{D}{\rho}$ indeed gives more information.*

Proof. For simplification, let us focus on a 1D explanation with a refinement coefficient of r. The velocity is unidimensional: $\mathbf{U} = U_x\widehat{\mathbf{e}}_{\mathbf{x}}$ and $\mathbf{u} = u_x\widehat{\mathbf{e}}_{\mathbf{x}}$. Consider a fine cell numbered $i + k$ inside the left half of the coarse cell numbered I. We saw earlier that we choose the following way to compute $\frac{d}{\rho}$:

$$\begin{aligned}
\frac{d}{\rho}(i+k) &= \mathbb{I}^h\left(\frac{D}{\rho}\right)(i+k), \\
&= \frac{r-1-2k}{2r}\frac{D}{\rho}(I-1) + \frac{r+1+2k}{2r}\frac{D}{\rho}(I), \\
&= \frac{r-1-2k}{2r}\frac{U_x(I-\frac{1}{2}) - U_x(I-\frac{3}{2})}{H} \\
&\quad + \frac{r+1+2k}{2r}\frac{U_x(I+\frac{1}{2}) - U_x(I-\frac{1}{2})}{H}, \\
&= \frac{1}{H}\frac{-r+1+2k}{2r}U_x(I-\frac{3}{2}) + \frac{1}{H}\frac{-2-4k}{2r}U_x(I-\frac{1}{2}) \\
&\quad + \frac{1}{H}\frac{r+1+2k}{2r}U_x(I+\frac{1}{2}).
\end{aligned} \tag{5.58}$$

To compute $\frac{d}{\rho}$, we do not use the divergence of the velocity $\mathbf{u}$ on the fine level. Indeed, it would lead to the following development:

$$\begin{aligned}
\nabla_h \mathbf{u}(i+k) &= \frac{u_x(i+k+\frac{1}{2}) - u_x(i+k-\frac{1}{2})}{h}, \\
&= \frac{r}{H}\left(\left(1-\frac{k}{r}\right)U_x(I-\frac{1}{2}) + \frac{k}{r}U_x(I+\frac{1}{2})\right. \\
&\qquad \left. -\left(1-\frac{k-1}{r}\right)U_x(I-\frac{1}{2}) - \frac{k-1}{r}U_x(I+\frac{1}{2})\right), \\
&= \frac{r}{rH}\left(U_x(I+\frac{1}{2}) - U_x(I-\frac{1}{2})\right), \\
&= \left(\nabla_H \cdot \mathbf{U}\right)(I).
\end{aligned} \tag{5.59}$$

In other words, according to Equation (5.59), all the fine cells $i+k$ contained in the coarse cell I host the same value for $\nabla_h \mathbf{u}(i+k)$: the divergence is equal to $\left(\nabla_H \cdot \mathbf{U}\right)(I)$. This is thus less precise than $\frac{d}{\rho}(i+k)$ shown in Equation (5.58). □

We can reformulate Theorem 9 by saying that from a discrete point of view, the linear interpolation and the divergence operators do not commute:

$$\nabla_h \circ \mathbb{I}^h \neq \mathbb{I}^h \circ \nabla_H. \tag{5.60}$$

5.1.5 Numerical results

We choose to use the test case of the lid-driven cavity. It is a well known benchmark to see whether the implementation of one-phase incompressible Navier-Stokes is correct. The literature provides us with the data for input as well as for the end results; see [133], [31]. We took the same data as in the tutorial for the nuclear code SATURN [122].

We still work with a 2D computational domain with the shape of a square; we give it dimensions of $L_{lid} \times L_{lid} = 1\,\mathrm{m} \times 1\,\mathrm{m}$. The upper side is called "the lid". The lid moves from left to right with a speed of $u_{lid} = 1\,\mathrm{m\,s^{-1}}$. The boundary conditions for $\mathbf{u}$ are no-slip conditions. It means that for all borders, $\mathbf{u}_{BC} \cdot \mathbf{n} = 0$. However $\mathbf{u}_{BC} \cdot \mathbf{t} = 0$ only for the three lower borders; for the lid we have $\mathbf{u}_{BC} \cdot \mathbf{t} = u_{lid} = 1\,\mathrm{m\,s^{-1}}$.

The kinematic viscosity in the standard test is $\nu_{lid-driven} = 10^{-3}\mathrm{m}^2\,\mathrm{s}^{-1}$. The density for the standard test is $\rho_{lid-driven} = 1\,\mathrm{kg}\,\mathrm{m}^{-3}$. We notice that this data is different from liquid water; $\nu_{water} = 10^{-6}\mathrm{m}^2\,\mathrm{s}^{-1}$ and $\rho_{water} = 1000\,\mathrm{kg}\,\mathrm{m}^{-3}$. We do not consider gravity effects, so $||\mathbf{g}|| = 0$.

Since L_{lid} is a characteristic length of the problem, we can compute a Reynolds $\mathcal{R}$ number associated to our simulation:

$$\mathcal{R} = \frac{L_{lid}\ u_{lid}}{\nu_{lid-driven}} = \frac{1 \times 1}{10^{-3}} = 1000. \tag{5.61}$$

This benchmark test presents a well-known permanent regime. It is independent of initial conditions: we use a initial velocity uniformly equal to zero and an initial pressure also uniformly equal to zero. The final state depends on the Reynolds number, though.

We represent our result of the permanent regime on Figure 5.2, with the colouring as a function of $||\mathbf{u}||$ and the streamlines given with the white arrows. Figure 5.3 also gives the permanent regime, with the colouring as a function of $||\mathbf{u}||$ and the isocontours of $||\mathbf{u}||$ given as white lines. A video of the transitional regime can be seen online at `https://youtu.be/esOHN--iW4Y`.

We can see on Figure 5.2 that the lid makes the liquid roll clock-wise in the square. This creates the principal whirlpool, a little up right to the centre of the square. The principal whirlpool drives the creation of a secondary whirlpool in the bottom right corner, and a tertiary one in the bottom left corner of the square. The number of whirlpools depends on the Reynolds number.

Because we knew what the final state would look like, we choose to use only one AMR patch. We position it in the upper right corner of the computational domain. We show the position on Figure 5.4 with a white rectangle. We set its position once and for all time steps, such that the centre of the primary whirlpool would be located inside it.

The best way to compare our result of final state to the benchmarks of the literature is to represent the velocity on two axis. Figure 5.5 shows $u_x(y)$ on the axis $x = 0.5\,\mathrm{m}$, as a function of the ordinate y. Figure 5.6 represents $u_y(x)$ on the axis $y = 0.5\,\mathrm{m}$, as a function of the abscissa x. The blue lines represent the data from our simulations. The green lines represent the data from Ghia et al. [71], which is often used as a benchmark, as for instance in [133], [31] or [122]. It is the computation on a single level (i.e. without

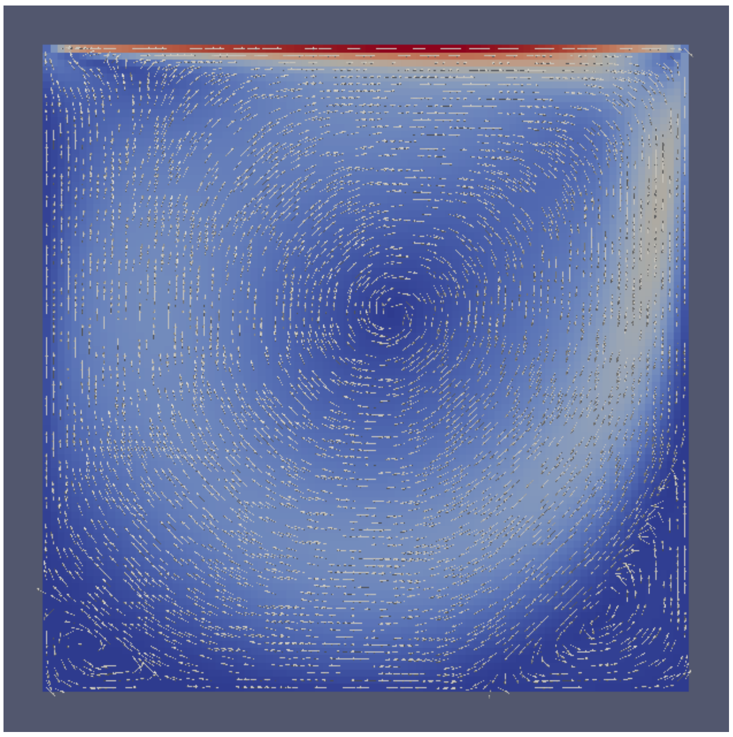


Figure 5.2 – Final state of lid-driven cavity simulation (streamlines of $\mathbf{u}$)

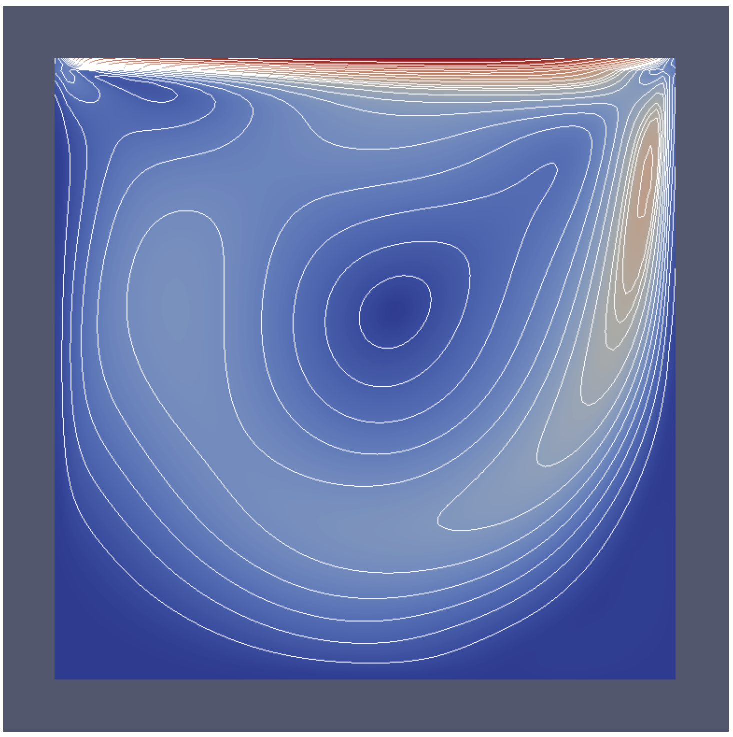


Figure 5.3 – Final state of lid-driven cavity simulation (isocontours of $||\mathbf{u}||$)

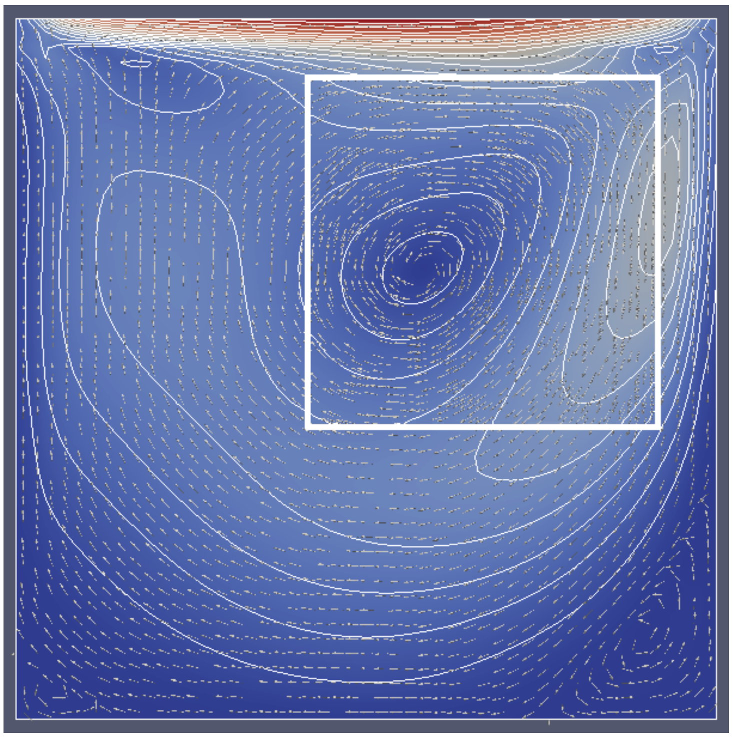


Figure 5.4 – Intermediary state of lid-driven cavity simulation (with white AMR patch, streamlines of $\mathbf{u}$ and isocontours of $||\mathbf{u}||$)

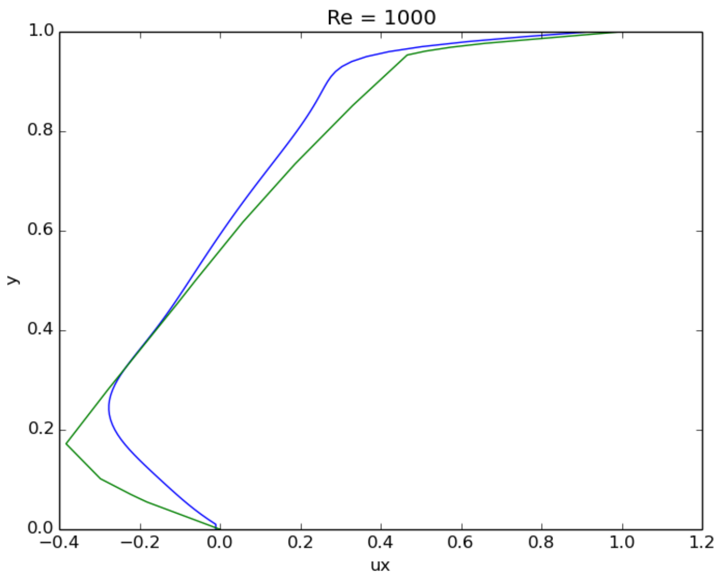


Figure 5.5 – u_x along vertical axis, as a function of y

AMR) with a 129 $\times$ 129 grid. The differences between literature and our results may come from our own mesh that may not have been fine enough: we have used a 48 $\times$ 48 grid with one level of refinement with a refinement coefficient $r = 2$. This is of course less precise than a grid of 96 $\times$ 96.

5.2 Two-fluid incompressible Navier-Stokes

For the two-fluid incompressible Navier-Stokes simulation too, we define a 2D computational domain with the shape of a square. As seen earlier in

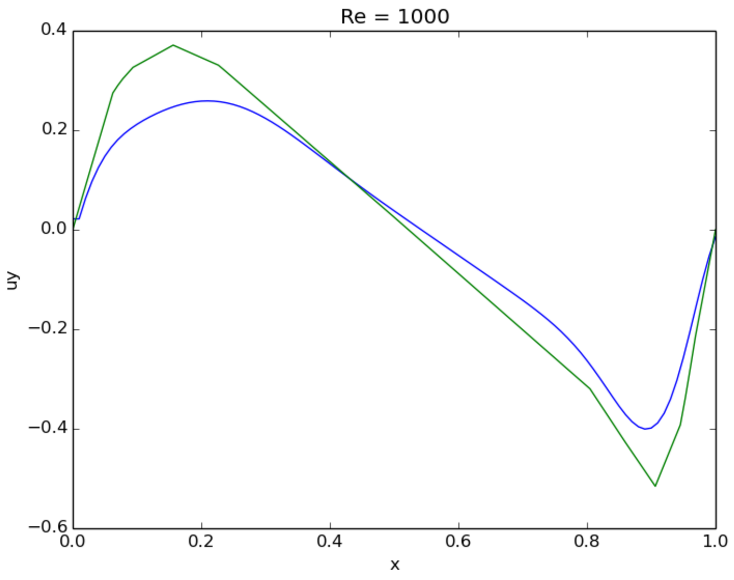


Figure 5.6 – u_y along horizontal axis, as a function of x

Section 2.1.3, we want to compute the following set of equations:

$$\begin{cases} \partial_t \mathbf{u} + \mathbf{u} \cdot \nabla \mathbf{u} - \nu \nabla^2 \mathbf{u} = \mathbf{f} = -\dfrac{1}{\rho} \nabla p + \mathbf{g} + \mathbf{f}_{others}, \\ \nabla \cdot \mathbf{u} = 0, \\ \partial_t Y + \mathbf{u} \nabla Y = 0, \\ \rho = \rho_g Y + \rho_l (1 - Y), \\ \mathbf{u} = \mathbf{u}_{BC} \text{ on } \partial\Omega \text{ (boundary conditions)}, \\ \mathbf{u}(t=0) = \mathbf{u}_0 \text{ on } \Omega \text{ (initial conditions)}, \\ Y(t=0) = Y_0 \text{ on } \Omega \text{ (initial conditions)}. \end{cases} \tag{5.62}$$

As seen before, Y is a function that equals either 0 or 1. Therefore, ρ equals either ρ_l or ρ_g. It is a function constant by pieces.

5.2.1 Implementation

Prediction-correction scheme

Here too we decide to use a prediction-correction scheme as done by Chorin and Temam. The model discretised in time is expressed as follows, for a time step numbered n and for a time incrementation Δt:

$$\begin{cases} \dfrac{\tilde{\mathbf{u}} - \mathbf{u}^n}{\Delta t} + \mathbf{u}^n \cdot \nabla \tilde{\mathbf{u}} - \nu \nabla^2 \tilde{\mathbf{u}} = -\dfrac{1}{\rho^n} \nabla p + \mathbf{g} + \mathbf{f}^n_{others}, \\ \\ \tilde{\mathbf{u}} = \mathbf{u}_{BC} \text{ on } \partial\Omega. \end{cases} \tag{5.63}$$

$$\begin{cases} -\nabla \cdot \left(\dfrac{1}{\rho^n} \nabla \phi^{n+1} \right) = -\dfrac{1}{\Delta t} \nabla \cdot \tilde{\mathbf{u}}, \\ \\ \nabla \phi^{n+1} \cdot \mathbf{n} = 0 \text{ on } \partial\Omega. \end{cases} \tag{5.64}$$

$$p^{n+1} = p^n + \phi^{n+1}. \tag{5.65}$$

$$\begin{cases} \dfrac{\mathbf{u}^{n+1} - \tilde{\mathbf{u}}}{\Delta t} = -\dfrac{1}{\rho^n} \nabla \phi^{n+1}, \\ \\ \mathbf{u}^{n+1} = \mathbf{u}_{BC} \text{ on } \partial\Omega. \end{cases} \tag{5.66}$$

$$\frac{Y^{n+1} - Y^n}{\Delta t} + \mathbf{u}^{n+1} \cdot \nabla Y^n = 0. \tag{5.67}$$

$$\rho^{n+1} = \rho_g Y^{n+1} + \rho_l (1 - Y^{n+1}). \tag{5.68}$$

Like for the one-phase incompressible Navier-Stokes simulation, we recognize $\tilde{\mathbf{u}}$ the prediction of velocity and ϕ the increment of pressure. We would like to point out a major difference with the one-phase incompressible model. Equation (5.64) reads as $-\nabla \cdot \left(\frac{1}{\rho^n} \nabla \phi^{n+1} \right) = -\frac{1}{\Delta t} \nabla \cdot \tilde{\mathbf{u}}$ whereas Equation (5.5) read as $-\frac{1}{\rho} \Delta \phi^{n+1} = -\frac{1}{\Delta t} \nabla \cdot \tilde{\mathbf{u}}$. This is because $\frac{1}{\rho}$ is not uniform any more. Of course, we can notice that if $\rho_l = \rho_g$, the two equations become equivalent.

Prediction step (5.63)

Similarly as for the one-phase case, we can solve $\tilde{\mathbf{u}}$ component by component:

$$\forall z \in \{x, y\}, \frac{\tilde{u}_z - u^n_z}{\Delta t} + u^n_x \frac{\partial \tilde{u}_z}{\partial x} + u^n_y \frac{\partial \tilde{u}_z}{\partial y} - \nu \left(\frac{\partial^2 \tilde{u}_z}{\partial x^2} + \frac{\partial^2 \tilde{u}_z}{\partial y^2} \right) = f^n_z. \tag{5.69}$$

Space discretisation

Like for the one-phase flow, we use a staggered grid. But here, we also locate the field $\frac{1}{\rho}$ on the faces, although it is a scalar. We compute $\left(\frac{1}{\rho}\right)^n$ as follows:

$$\left(\frac{1}{\rho}\right)^n (i+\frac{1}{2},j) = \frac{1}{2}\left(\frac{1}{\rho^n(i,j)} + \frac{1}{\rho^n(i+1,j)}\right), \tag{5.70}$$

$$\left(\frac{1}{\rho}\right)^n (i,j+\frac{1}{2}) = \frac{1}{2}\left(\frac{1}{\rho^n(i,j)} + \frac{1}{\rho^n(i,j+1)}\right). \tag{5.71}$$

On the borders, we decide to impose Neumann boundary conditions:

$$\begin{cases} \left(\frac{1}{\rho}\right)^n (-\frac{1}{2},j) = \frac{1}{\rho^n(0,j)}, \\ \left(\frac{1}{\rho}\right)^n (N-\frac{1}{2},j) = \frac{1}{\rho^n(N-1,j)}, \\ \left(\frac{1}{\rho}\right)^n (i,-\frac{1}{2}) = \frac{1}{\rho^n(i,0)}, \\ \left(\frac{1}{\rho}\right)^n (i,N-\frac{1}{2}) = \frac{1}{\rho^n(i,N-1)}. \end{cases} \tag{5.72}$$

For the advection part, we still choose an upwind scheme. Similarly to the one-phase computation, here too we need to "densify" $\mathbf{u}$ to get $\mathbf{U}$.

Matrix form of (5.63) and boundary conditions

The matrix form of Equation (5.63) is the same as the one for the one-phase computation (5.20):

$$A_z^n \tilde{u}_z = b_z^n. \tag{5.73}$$

Boundary conditions are also expressed the same way, so we will not rewrite them here. Please refer to Sections 5.1.2 and 5.1.2.

Discretisation and resolution of Equation (5.64)

Let us note $d = -\nabla \cdot \tilde{\mathbf{u}}$. We want to solve the following equation:

$$\begin{cases} -\nabla \cdot \left(\left(\frac{1}{\rho}\right)^n \nabla \phi^{n+1}\right) = -\frac{1}{\Delta t}\nabla \cdot \tilde{\mathbf{u}} = \frac{d}{\Delta t}, \\ \nabla \phi^{n+1} \cdot \mathbf{n} = 0 \text{ on } \partial\Omega. \end{cases} \tag{5.74}$$

The variable ϕ is located at the centre of cells, so there are N^2 unknowns.

$$d(i,j) = -\left(\frac{\tilde{u}_x(i+\frac{1}{2},j) - \tilde{u}_x(i-\frac{1}{2},j)}{\Delta x} + \frac{\tilde{u}_y(i,j+\frac{1}{2}) - \tilde{u}_y(i,j-\frac{1}{2})}{\Delta y}\right). \tag{5.75}$$

$$\begin{aligned}
&-\nabla \cdot (\left(\frac{1}{\rho}\right) \nabla\phi)(i,j) \\
&= -\frac{1}{\Delta x}\left(\left(\frac{1}{\rho}\nabla\phi\right)(i+\frac{1}{2},j) - \left(\frac{1}{\rho}\nabla\phi\right)(i-\frac{1}{2},j)\right) \\
&\quad -\frac{1}{\Delta y}\left(\left(\frac{1}{\rho}\nabla\phi\right)(i,j+\frac{1}{2}) - \left(\frac{1}{\rho}\nabla\phi\right)(i,j-\frac{1}{2})\right), \\
&= -\frac{1}{\Delta x}\left(\frac{1}{\rho}\right)(i+\frac{1}{2},j)\frac{\phi(i+1,j)-\phi(i,j)}{\Delta x} \qquad (5.76) \\
&\quad +\frac{1}{\Delta x}\left(\frac{1}{\rho}\right)(i-\frac{1}{2},j)\frac{\phi(i,j)-\phi(i-1,j)}{\Delta x} \\
&\quad -\frac{1}{\Delta y}\left(\frac{1}{\rho}\right)(i,j+\frac{1}{2})\frac{\phi(i,j+1)-\phi(i,j)}{\Delta y} \\
&\quad +\frac{1}{\Delta y}\left(\frac{1}{\rho}\right)(i,j-\frac{1}{2})\frac{\phi(i,j)-\phi(i,j-1)}{\Delta y}.
\end{aligned}$$

We solve a system $C\phi = d$ where $C \in \mathbb{M}(N^2)$.

$$\begin{aligned}
C(ij,ij) = &\frac{\Delta t}{\Delta x^2}\left(\left(\frac{1}{\rho}\right)(i-\frac{1}{2},j) + \left(\frac{1}{\rho}\right)(i+\frac{1}{2},j)\right) \\
&+\frac{\Delta t}{\Delta y^2}\left(\left(\frac{1}{\rho}\right)(i,j-\frac{1}{2}) + \left(\frac{1}{\rho}\right)(i,j+\frac{1}{2})\right).
\end{aligned} \tag{5.77}$$

$$C(ij,ij-1) = -\frac{\Delta t}{\Delta x^2}\left(\frac{1}{\rho}\right)(i-\frac{1}{2},j). \tag{5.78}$$

$$C(ij,ij+1) = -\frac{\Delta t}{\Delta x^2}\left(\frac{1}{\rho}\right)(i+\frac{1}{2},j). \tag{5.79}$$

$$C(ij,ij-N+1) = -\frac{\Delta t}{\Delta y^2}\left(\frac{1}{\rho}\right)(i,j-\frac{1}{2}). \tag{5.80}$$

$$C(ij, ij+N-1) = -\frac{\Delta t}{\Delta y^2}\left(\frac{1}{\rho}\right)(i, j+\frac{1}{2}). \tag{5.81}$$

To respect the Neumann boundary conditions, we impose $\phi(-1, j) = \phi(0, j)$, so we get the following expression:

$$\begin{aligned} C(0+jN, 0+jN) = {} & \frac{\Delta t}{\Delta x^2}\left(\frac{1}{\rho}\right)(\frac{1}{2}, j) \\ & + \frac{\Delta t}{\Delta y^2}\left(\frac{1}{\rho}\right)(0, j-\frac{1}{2}) \\ & + \frac{\Delta t}{\Delta y^2}\left(\frac{1}{\rho}\right)(0, j+\frac{1}{2}). \end{aligned} \tag{5.82}$$

Resolution of (5.65), (5.66), (5.67), (5.68)

The resolution of Equation (5.65) is trivial:

$$p^{n+1}(i, j) = p^n(i, j) + \phi(i, j). \tag{5.83}$$

Equation (5.66) can be computed after the gradient of ϕ is calculated:

$$u_x^{n+1}(i+\frac{1}{2}, j) = \tilde{u}_x(i+\frac{1}{2}, j) - \Delta t\left(\frac{1}{\rho}\right)^n (i+\frac{1}{2}, j)\left(\nabla\phi\right)^{n+1}(i+\frac{1}{2}, j). \tag{5.84}$$

The resolution of the advection equation (5.67) is done with the anti-dissipative limited downwind scheme, as shown in earlier parts (see Section 3.4.1 for instance). The resolution of Equation (5.68) is trivial too:

$$\rho^{n+1}(i, j) = \rho_g Y^{n+1}(i, j) + \rho_l\left(1 - Y^{n+1}(i, j)\right). \tag{5.85}$$

Adaptive Mesh Refinement

Like for the one-phase computation, we decide to use only one level of refinement for the AMR. As we will see, we make the decision to focus the AMR on the pressure variations ϕ, as well on the colour function Y. Because of the discontinuity of $\left(\frac{1}{\rho}\right)$, the pressure p may be continuous but its gradient ∇p is not at the bubble interfaces; it is useful to refine with p in mind. Conversely, we used a uniform kinematic viscosity ν, so both $\tilde{\mathbf{u}}$ and $\nabla\cdot\tilde{\mathbf{u}}$ are continuous. As a consequence, we do not drive our choices of refinement because of $\tilde{\mathbf{u}}$.

Equation (5.63) gives the predicted velocity $\tilde{\mathbf{u}}$. It is solved by resolution of the linear system (5.73) on the coarse level. But for the fine level, we decide to compute the predicted velocity $\tilde{\mathbf{u}}$ from a linear interpolation of the coarse level onto the fine level. This fine $\tilde{\mathbf{u}}$ is used for the fine version of Equation (5.66).

From $\tilde{\mathbf{u}}$ on the coarse grid, we also compute a coarse $\nabla \cdot \tilde{\mathbf{u}}$. This variable is linearly interpolated to get $\nabla \cdot \tilde{\mathbf{u}}$ on the fine grid. The divergence of the predicted velocity is used for the source term of Equation (5.64).

Equation (5.64) gives the increment of pressure ϕ. We fully use the AMR possibilities here by computing it using the LDC algorithm explained in Section 4.1. This gives an accurate result to the elliptic equation on the coarse as well as on the fine level.

Equation (5.65) is just an addition, be it on the coarse or on the fine level.

Equation (5.66) is also mostly an addition. So on each level we compute it with the data associated to that level.

Equation (5.67) is the advection of Y with the newly computed $\mathbf{u}$. We compute the advection with the limited downwind scheme on the coarse level as well as on the fine level. In other words we proceed exactly as explained earlier in Section 3.4 for instance. This gives an accurate result to the hyperbolic equation on the coarse level as well as on the fine level.

Equation (5.68) is a simple arithmetic operation. So on each level we compute it with the data associated to that level.

5.2.2 Tension forces

In Equation (5.63), we gather physical forces in the acceleration $\mathbf{f}$. It includes the gravitational acceleration $\mathbf{g}$ and the pressure forces. In order to have a more physical representation of bubbles, we want to also include tension forces in the term $\mathbf{f}_{others}$. Let $\mathbf{T}_{surf}$ be the acceleration due to the surface tension of bubbles:

$$\mathbf{f} = -\frac{1}{\rho}\nabla p + \mathbf{g} + \mathbf{T}_{surf}. \tag{5.86}$$

Let τ_{surf} be a constant acceleration coefficient, with units of $\mathrm{m\,s^{-2}}$. Let $\mathbf{n}$ be the unit vector normal to the interface and pointing towards the liquid phase l. Let $\kappa = \nabla \cdot \mathbf{n}$ be the curvature of an interface between two fluids. From the theoretical point of view, the surface tension is located only on the interface itself and it is given by the following expression:

$$\mathbf{T}_{surf} = -\tau\kappa\mathbf{n}. \tag{5.87}$$

As mentioned in Section 2.2.2, calculating the curvature $\kappa = \nabla \cdot \mathbf{n}$ would have been very easy with a level-set function. Here instead, we have a discretised colour function Y. Because we kept Y intentionally as sharp as possible, the computation of its gradient is prone to present two issues. It is too much located on just the interface and none of its surroundings, and its direction is often either horizontal or vertical (instead of pointing towards the inside of the bubble). So we have to find some way for a satisfying expression of a discretised $\mathbf{T}_{surf}$.

Such a way is proposed by Brackbill et al. in [33]. We use a "smoothing" operator $\mathbb{S}$ on Y^h to get $Y^h_{smooth} = \mathbb{S}(Y^h)$, a "mollified" colour function. The operator $\mathbb{S}$ is just the average of Y^h on the five closest cells:

$$\begin{aligned} Y^h_{smooth}(i,j) &= \mathbb{S}(Y^h)(i,j) \\ &= \frac{1}{5}\big(Y^h(i,j) + Y^h(i-1,j) + Y^h(i+1,j) \\ &\qquad + Y^h(i,j-1) + Y^h(i,j+1)\big). \end{aligned} \tag{5.88}$$

We notice that when $\Delta x \to 0$, Y^h_{smooth} tends to Y, similarly to Y^h. We set $\mathbf{N}$ as the gradient of Y^h_{smooth}:

$$\mathbf{N} = \nabla Y^h_{smooth} \tag{5.89}$$

The vector field $\mathbf{N}$ is normal to the interface and it points towards the inside of bubbles. The unit normal is thus given by the following expression:

$$\mathbf{n} = -\frac{\nabla Y^h_{smooth}}{||\nabla Y^h_{smooth}||}. \tag{5.90}$$

As a consequence, the surface tension acceleration of Equation (5.87) is given by the following formula:

$$\mathbf{T}_{surf} = -\tau\ \nabla \cdot \mathbf{n}\ \mathbf{N}. \tag{5.91}$$

At some point, we also have to give a value to the curvature on faces:

$$\kappa(i+\frac{1}{2},j) = \frac{\kappa(i,j) + \kappa(i+1,j)}{2}, \tag{5.92}$$

$$\kappa(i,j+\frac{1}{2}) = \frac{\kappa(i,j)+\kappa(i,j+1)}{2}. \tag{5.93}$$

In the end, this leads us to the following formulas:

$$T_{surf,x}(i+\frac{1}{2},j) = \tau(\nabla\cdot\mathbf{n})(i+\frac{1}{2},j)\nabla Y^h_{smooth}(i+\frac{1}{2},j), \tag{5.94}$$

$$T_{surf,y}(i,j+\frac{1}{2}) = \tau(\nabla\cdot\mathbf{n})(i,j+\frac{1}{2})\nabla Y^h_{smooth}(i,j+\frac{1}{2}). \tag{5.95}$$

Also see [127] for more information about surface tension.

5.2.3 Numerical results

We want to make the most complete simulation possible in order to test out any eventual limit of our computation.

We choose a 2D square domain of dimensions $1\,\mathrm{m}\times 1\,\mathrm{m}$. For the boundary conditions of the velocity, we choose the same ones as for the lid-driven cavity test case: no-slip conditions, with the upper boundary (the lid) moving horizontally to the right at a speed of $u_{lid} = 1\,\mathrm{m\,s^{-1}}$.

We give the surrounding fluid, indexed l, a kinematic viscosity of $\nu_l = \nu_{water} = 10^{-6}\mathrm{m^2\,s^{-1}}$, and a density of $\rho_l = \rho_{water} = 1000\,\mathrm{kg\,m^{-3}}$. We give the dispersed fluid, indexed g, the same kinematic viscosity $\nu_g = 10^{-6}\mathrm{m^2\,s^{-1}}$. But we choose $\rho_g = \rho_l/2 = 500\,\mathrm{kg\,m^{-3}}$.

Gravity $||\mathbf{g}||$ is set to $9.8\,\mathrm{m\,s^{-2}}$ in the downwards direction. The surface tension parameter τ_{surf} is set to $10\,\mathrm{m\,s^{-2}}$.

For the initial conditions, we choose to consider two bubbles of liquid 1. We place one in the shape of a circle in the bottom left corner. We place the other in the shape of an ellipse (to test out the tension forces) in the upper right corner. Figure 5.7 represents this initial state.

An image of an intermediary state is given at Figure 5.8. A video can be seen online at `https://youtu.be/1l_cdEDH-2I`. We can see the AMR patches as white rectangles and the streamlines of the velocity as white little arrows. The left bubble goes up due to Archimede's force. After going up a little, the right bubble is quickly drifted by the lid and stuck in a whirlpool in the upper right quadrant.

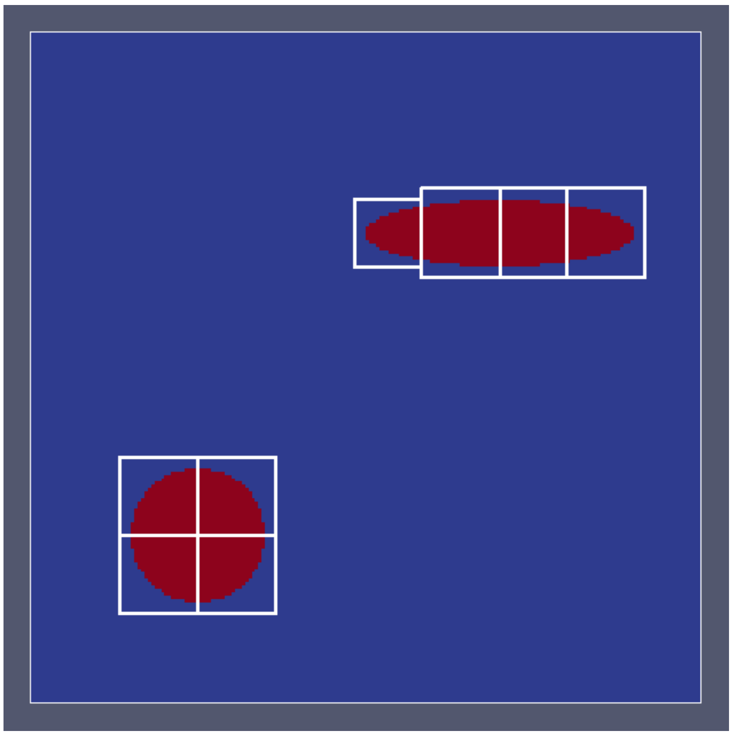


Figure 5.7 – Initial state for two-fluid incompressible Navier-Stokes simulation, with AMR patches shown as white rectangles

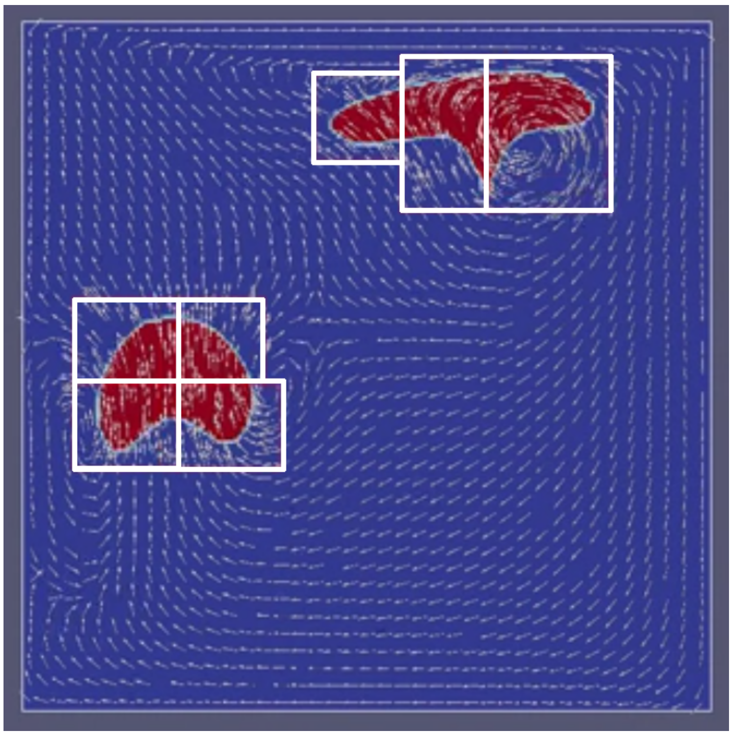


Figure 5.8 – Intermediary state for two-fluid incompressible Navier-Stokes simulation, with AMR patches shown as white rectangles

Chapter 6

Outreach

6.1 Perspectives

In this thesis, we have reviewed a series of models for flows with two separated phases. As part of our list, the ones we simulated were the incompressible Navier-Stokes model and the Abstract Bubble Vibration model. We think that future work should also focus on the Diphasic Low-Mach Number model, as it is especially adapted for nuclear Thermal-Hydraulics [47]. As far as our choice of modelisation of an interface, we were highly satisfied by the front capturing method. In particular, the transport of the colour function Y showed to be precise enough, thanks to the Després-Lagoutière advection scheme and Adaptive Mesh Refinement.

Speaking of which, we are convinced that patch-based AMR will carry further improvements to simulations. It would be interesting to continue our research efforts to implement the $n_{min} - n_{max}$ algorithm with distributed memory parallelisation technologies, as for instance MPI [147]. Further developments would then have to be made, in order to get higher speed-ups and cluster-enabled computation [117]. The patch-based approach seems to be also very promising for multi-model computations. One could imagine that we would use a first model for the coarse level computation, and a second model for the fine level. In this case, a lot remains to be done at least for the convergence analysis.

The ABV model permitted us to test drive the LDC algorithm and hyperbolic-elliptic coupling. We think there is still room for a more complete proof of LDC with different interpolations, as well as cell-centred data combined

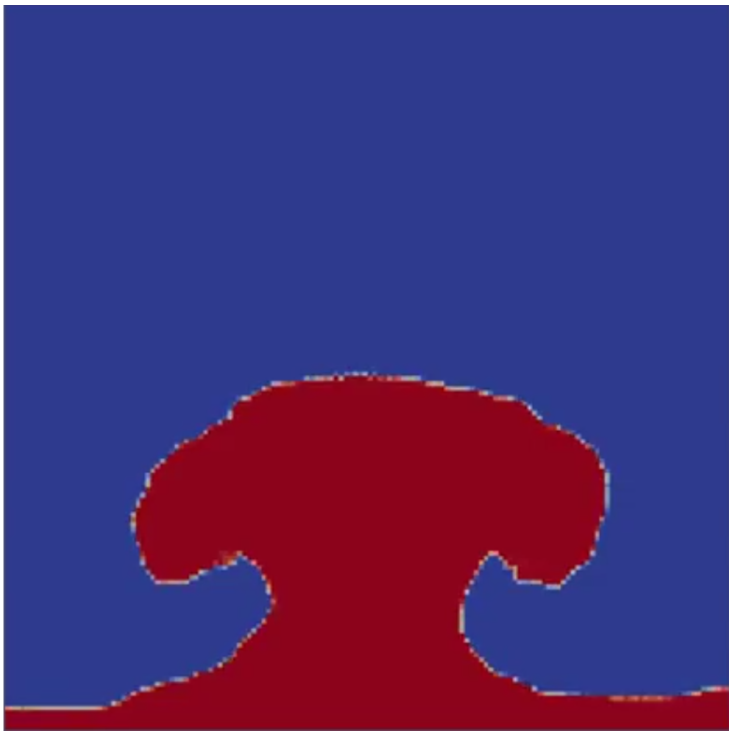


Figure 6.1 – Mushroom-shaped Rayleigh-Taylor instability

with ghost cells. As we verified ABV with the volume formula, we think it would be good to make more tests with a wider selection of grids and a more precise quantification of the error depending on whether we use AMR or not. Still, we think that our implementation with CDMATH will be useful to be fully comprehended by anyone who would want to reuse it. Both from a theoretical point of view and a coding point of view, we made special efforts for it to be adopted by potential other users.

We applied the tools we designed for one- and two-phase incompressible Navier-Stokes model simulations. For the lid-driven cavity simulation, we would have liked to get even more precise results; even closer to the literature benchmarks. This would likely have been achieved with more precise grids. The two-fluid simulation seems realistic, but we would have much enjoyed to test our code with a classic case of Kelvin-Helmholtz [86, 69] or Rayleigh–Taylor [124] instabilities for example, for benchmark purposes. The latter is the unstable superposition of two fluids with different densities, the heavier being on top. Figure 6.1 shows preliminary results and a video can be seen online at `https://youtu.be/l_TOop-prqM`.

Further developments may also expand more on phase transition. Of course it would depend on the model used for the other phenomenons [80], but we could for instance try with the Stefan model for phase change [85]. More complete simulations could then be validated with experimental data at the bubble scale [112, 111], in order to improve knowledge of important thermal-hydraulic phenomenons occurring in nuclear power plants such as the Departure of Nuclear Boiling.

6.2 Communications

During the three years, we had many opportunities to present our research results:

- Poster presented at the CEA-GAMNI seminar in January 2014 at Institut Henri Poincaré in Paris, on the topic "*Analysis of Interfacial Forces on the Physics of Two-Phase Flow and Hyperbolicity of the Two-Fluid Model*", which is not linked to the topic of the PhD thesis but rather stemming from prior work realised during the Masters of Science.

- Poster presented at the Congrès National d'Analyse Numérique (CANUM 2014) in April 2014 in Carry-le-Rouet, on the topic "*Analysis of the efficiency and relevance of patch creation algorithms in the case of AMR of thin flagged areas*" which is linked to the topic of this thesis.

- Participation to "Ma Thèse en 180 secondes" (MT180) in April 2014. It consists in an international competition which purpose is to popularise one's PhD topic in three minutes.

- Poster presented at the CEA-GAMNI seminar in January 2015 at Institut Henri Poincaré in Paris, on the topic "*Direct Numerical Simulation of bubbles using Adaptive Mesh Refinement on parallel architecture*".

- Presentation at the LRC-MANON workgshop in February 2015 at the Jacques-Louis Lions laboratory in Paris VI, on the topic "*Direct Numerical Simulation of bubbles using Adaptive Mesh Refinement on parallel architecture*".

- Presentation at the European Nuclear Young Generation Forum (ENYGF) in June 2015 at the Cité des Sciences et de l'Industrie, on the topic "*Direct Numerical Simulation of bubbles using Adaptive Mesh Refinement on parallel architecture*".

- Presentation at the "*Multiphase Flows*" congress in September 2015 at the Institut d'Études Scientifiques de Cargèse, on the topic "*Direct Numerical Simulation of bubbles using Adaptive Mesh Refinement on parallel architecture*".

- Presentation at the "*Low Velocity Flows*" congress in November 2015 at Université Paris Descartes, on the topic "*Direct Numerical Simulation of bubbles using Adaptive Mesh Refinement on parallel architecture*".

- Presentation at the Congrès National d'Analyse Numérique (CANUM 2016) in May 2016 at Obernai, on the topic "*Direct Numerical Simulation of bubbles using Adaptive Mesh Refinement on parallel architecture*".

We also presented our work multiple times internally; to the laboratory, to the department, to the wider CEA Saclay audience, to the DGA R&D policy makers...

6.3 Supervision and funding

We realised this work under supervision from the French national laboratory CEA and from École Polytechnique, both belonging to Université Paris Saclay. We received joint financial support from CEA and from the Research and Development service of the French Ministry of Defence; DGA.

Appendix A

Implementation of the clustering algorithms

For factorisation and hence readability purposes, we implemented the three algorithms together in a single program. Hereunder follows a simplified expression of a part of MEDCoupling, the library in which we did the implementation. The syntax is inspired from C++.

```
/*!
 * This generic algorithm can be degenerated into 3 child ones,
 * depending on the arguments given; in particular depending
 * on whether they are equal to 0 or not.
 * 1/ If  minimumPatchLength nmin = 0,
 *   maximumPatchLength nmax = infinite,
 *   and maximumPatchVolume Nmax = 0,
 *   then we have the Berger-Rigoutsos algorithm.
 *   This algorithm was developed in 1991 and seems appropriate
 *   for sequential programming.
 * 2/ If maximumPatchLength nmax = infinite,
 *   then we have the Livne algorithm.
 *   This algorithm was developed in 2004 and is an improvement
 *   of the Berger-Rigoutsos algorithm.
 * 3/ If maximumPatchVolume Nmax = infinite,
 *   then we have the nmin-nmax algorithm.
 *   This algorithm was developed by Arthur TALPAERT in 2014
 *   and is an improvement of the Livne algorithm.
 *   It is especially appropriate for parallel computing,
 *   where one patch would be given to one CPU.
 *  See Arthur TALPAERT's 2014 CANUM poster for more information.
 */
vector<Patch>
createPatchesFromCriterion(BoxSplittingOptions bso,
    vector<int> factors)
{
  /*
   * Here initialisation of refinement criteria:
   *  - eta_goal
   *  - eta_threshold
   *  - nmin_goal
   *  - nmax_goal
   *  - Nmax_goal
   */
  vector<Patch> listOfPatches;
  vector<Patch> listOfPatchesOK;

  while (not listOfPatches.empty())
  {
    vector<Patch> listOfPatchesTmp;
    for(Iterator it=listOfPatches.begin();
      it!=listOfPatches.end();
      it++)
    {
      /*
       * Here initialisation of local patch characteristics:
       *  - eta_local
       *  - nmin_local
       *  - nmax_local
       *  - widestAxis
       *  - Nmax_local
       */
      int cutPlace;
      if ( eta_local >= eta_threshold
        and (N_local > Nmax_goal or nmax_local > nmax_goal)
        and (DissectBigPatch(bso,it,widestAxis,
          lmax_local,cutPlace)) )
      // Action 1
      {
        DealWithCut(nmin_goal, it, widestAxis,
          cutPlace, listOfPatchesTmp);
        continue;
      }
      if ( FindHole(bso,it,widestAxis,cutPlace) )
      // Action 2
      {
         DealWithCut(lmin_goal, it, widestAxis,
           cutPlace, listOfPatchesTmp);
         continue;
      }
      if ( FindInflection(bso,it,cutPlace,widestAxis) )
      // Action 3
      {
        DealWithCut(nmin_goal, it, widestAxis,
          cutPlace, listOfPatchesTmp);
        continue;
      }
      if ( RespectEtaLmaxNmax(bso,it,widestAxis,
        nmax_local,cutPlace) )
      // Action 4
      {
        DealWithCut(nmin_goal, it, widestAxis,
          cutPlace, listOfPatchesTmp);
        continue;
      }
      else
        listOfPatchesOK.push_back(DealWithNoCut(it));
    }
    listOfPatches = listOfPatchesTmp;
  }
  return listOfPatchesOK;
}

bool
RespectEtaLmaxNmax(BoxSplittingOptions bso,
    Patch patchToBeSplit,
    int widestAxis,
    int nmax_local,
    int cutPlace)
{
  if ( eta_local <= eta_goal )
  {
    if ( nmax_local >= 2*nmin_goal )
      cutPlace = nmax_local/2 + patchToBeSplit.firstCoord() -1;
    else
      return false;
  }
  else
  {
    if ( N_local > Nmax_goal or nmax_local > nmax_goal )
      return DissectBigPatch(bso,
        patchToBeSplit,
        widestAxis,
        nmax_local,
        cutPlace);
    else
      return false;
  }
  return true;
}
```

Appendix B

Poisson problem

The purpose is to numerically solve the 2D Poisson problem, respectively with periodic, Neumann and Dirichlet boundary conditions:

$$\left\{\begin{array}{l} -\Delta\phi = s \text{ on } \Omega, \\ \forall(x,y), \phi(x+L_x,y) = \phi(x,y), \\ \forall(x,y), \phi(x,y+L_y) = \phi(x,y). \end{array}\right. \tag{B.1}$$

$$\left\{\begin{array}{l} -\Delta\phi = s \text{ on } \Omega, \\ \nabla\phi \cdot \mathbf{n} = 0 \text{ on } \partial\Omega. \end{array}\right. \tag{B.2}$$

$$\left\{\begin{array}{l} -\Delta\phi = s \text{ on } \Omega, \\ \phi = \phi_{BC} \text{ on } \partial\Omega. \end{array}\right. \tag{B.3}$$

where $L_x \times L_y$ are the dimensions of the computational domain, and ϕ_{BC} is given. We use an analytic form of s such that we know the theoretical solution ϕ_{exact}. This way, we can verify our computation methods.

To discretise the Laplacian operator $-\Delta$, we use a very commonly used technique centred on cells. Let $(i,j) \in [\![0, n_x-1]\!] \times [\![0, n_y-1]\!]$ be discrete coordinates for the mesh attached to the computational domain Ω. Let Δx be the elementary space step in the x direction and Δy be the elementary space step in the y direction. For all $(i,j) \in [\![1, n_x-2]\!] \times [\![1, n_y-2]\!]$, we use the following formula:

$$\begin{aligned} -\Delta\phi(i,j) = & \frac{-\phi(i-1,j) + 2\phi(i,j) - \phi(i+1,j)}{\Delta x} \\ & \quad + \frac{-\phi(i,j-1) + 2\phi(i,j) - \phi(i,j-1)}{\Delta y}. \end{aligned} \tag{B.4}$$

For the cells near the boundaries, it depends on the chosen boundary conditions. The scheme is classically called the finite differences "five-point stencil" because the evaluation of $-\Delta\phi(i,j)$ uses the value of ϕ on the cell numbered (i,j) plus on its four neighbouring cells.

Let us assume we have found an approximation $(\phi_{approx,i})_i$, where $i \in [\![1, n_{cells}]\!]$. We define $(\phi_{exact,i})_i = \phi_{exact}(x_i)$ where $(x_i)_i$ are the centres of all the cells. Let us also define v_i the elementary volume of cell i. We measure the L^2 error with the following formula:

$$\begin{aligned} L^2 error &= \|\phi_{exact} - \phi_{approx}\|_{L^2}, \\ &= \left(\sum_i v_i(\phi_{exact,i} - \phi_{approx,i})^2\right)^{\frac{1}{2}}. \end{aligned} \tag{B.5}$$

The error is a function of the elementary grid size. We expect it to decrease with refinement of the grids.

We made successful tests with periodic and Neumann boundary conditions for the Poisson problem. We detail here similar success for non-homogeneous Dirichlet conditions (B.3). We use the example from [57], where the source term is the addition of a "gentle" trigonometric function and an exponential spike located in the centre. Let us note $\varrho = \sqrt{x^2+y^2+z^2}$ the distance to the centre and $\chi(\varrho) = \mathbb{1}_{\varrho \leq \epsilon}$ an indicator function. Then the source term is given by the following formula:

$$\begin{aligned} s(x,y,z) = 2k^2\pi^2 &\cos(k\pi x)\cos(k\pi y) \\ &-4\eta\chi(\varrho)\exp\left(\frac{1}{\epsilon^2}\right)\exp\left(\frac{-1}{\epsilon^2-\varrho^2}\right)\frac{\varrho^2+\varrho^4-\epsilon^4}{(\epsilon^2-\varrho^2)^4}. \end{aligned} \tag{B.6}$$

The analytic solution is thus given by the following:

$$\phi_{exact} = \cos(k\pi x)\cos(k\pi y) + \eta\chi(\varrho)\exp\left(\frac{1}{\epsilon^2}\right)\exp\left(\frac{-1}{\epsilon^2-\varrho^2}\right). \tag{B.7}$$

As for numerical values, we use $k = 1/2$, $\eta = 10$ and $\epsilon = 1/4$. We use a 2D computational domain of size $[0,1] \times [0,1]$. We take a series of Cartesian meshes where $\Delta x = \Delta y$ and hence $n_x = n_y$. We use one patch, not centred but we ensure that it still covered the exponential spike. The refinement coefficient r is equal to 2.

The plot of $\log(L^2 error)$ as a function of $\log(\Delta x \Delta y)$ is shown in Figure B.1. The numbers (4, 5, 6) indicate the number of LDC iterations needed before convergence of the algorithm.

As the red dotted line is nearly parallel to the green one, we can conclude we are close to a second order convergence of the LDC method.

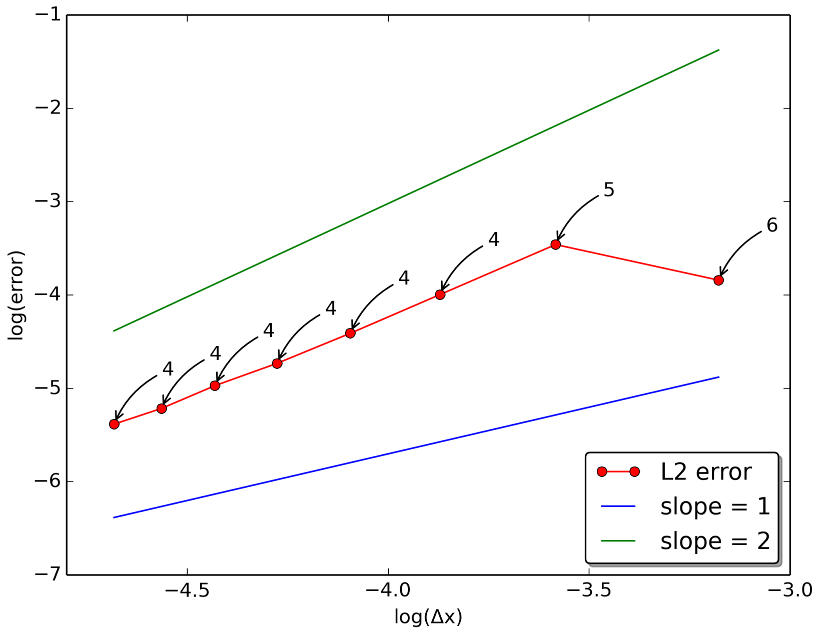


Figure B.1 – $\log(L^2 error)$ as a function of $\log(\Delta x \Delta y)$

Appendix C

Structure et résumé de la thèse

C.1 Introduction

Le manuscrit issu du travail de thèse a été rédigé en anglais. Cette annexe C en un résumé substantiel en français. Le chapitre 1 est une introduction générale au sujet. Nous expliquons le contexte et les objectifs de nos recherches dans la section 1.1 ; l'étude des écoulements diphasiques dans les réacteurs nucléaires. Les écoulements de ce type sont en effet présents dans le cœur des réacteurs à eau bouillante et dans les générateurs de vapeur. Leur simulation fine permet d'augmenter encore la sûreté et l'efficacité de ces systèmes industriels.

Nous consacrons ensuite la section 1.2 à la différentiation des échelles de simulation numérique faite pour la discipline de la thermohydraulique nucléaire. Dans l'ordre de la plus grande échelle à la plus petite, nous avons l'échelle système, l'échelle composant, l'échelle de la 3D locale et la simulation numérique directe (SND). Plus l'échelle est grande, plus l'effort de modélisation est grand. Plus l'échelle est petite, plus le calcul informatique est lourd mais aussi plus il peut représenter des phénomènes précis. Le travail de thèse porte sur la simulation des bulles, à l'échelle du centimètre (en espace) et de la seconde (en temps).

La section 1.3 est l'exposition de la structure et du résumé de la thèse en anglais, c'est-à-dire la traduction de cette présente annexe C.

C.2 Modèles d'écoulements à deux phase séparées

Le chapitre 2 présente un ensemble de modèles thermohydrauliques existants, avec un accent tout particulier pour les problèmes à deux phases séparées. Nous commençons avec les modèles compressibles communs : les systèmes d'Euler ainsi que de Navier-Stokes. Nous expliquons le premier avec uniquement une expression monophasique dans la section 2.1.1 et le deuxième avec une expression diphasique dans la section 2.1.2. Dans la section 2.1.3, nous prenons l'hypothèse de l'incompressibilité, afin d'obtenir le modèles de Navier-Stokes incompressible :

$$\begin{cases} \partial_t \mathbf{u} + \mathbf{u} \cdot \nabla \mathbf{u} - \nu \nabla^2 \mathbf{u} = \mathbf{f} = -\dfrac{1}{\rho} \nabla p + \mathbf{g} + \mathbf{f}_{autres}, \\ \nabla \cdot \mathbf{u} = 0, \\ \mathbf{u} = \mathbf{u}_{CL} \text{ sur } \partial\Omega \text{ (conditions aux limites)}, \\ \mathbf{u}(t=0) = \mathbf{u}_0 \text{ sur } \Omega \text{ (conditions initiales)}. \end{cases} \tag{C.1}$$

L'équation (C.1) donne l'expression monophasique, avec des notations qui sont définies précisément dans le manuscrit. L'expression diphasique est donnée par l'équation (C.2) :

$$\begin{cases} \partial_t \mathbf{u} + \mathbf{u} \cdot \nabla \mathbf{u} - \nu \nabla^2 \mathbf{u} = \mathbf{f} = -\dfrac{1}{\rho} \nabla p + \mathbf{g} + \mathbf{f}_{autres}, \\ \nabla \cdot \mathbf{u} = 0, \\ \partial_t Y + \mathbf{u} \cdot \nabla Y = 0, \\ \rho = \rho_g Y + \rho_l (1 - Y), \\ \mathbf{u} = \mathbf{u}_{CL} \text{ sur } \partial\Omega \text{ (conditions aux limites)}, \\ \mathbf{u}(t=0) = \mathbf{u}_0 \text{ sur } \Omega \text{ (conditions initiales)}, \\ Y(t=0) = Y_0 \text{ sur } \Omega \text{ (conditions initiales)}. \end{cases} \tag{C.2}$$

Nous poursuivons dans la section 2.1.4 avec les modèles utilisés dans les codes nucléaires ; les modèles respectivement à 6 et 7 équations, aussi connus sous le nom de modèles de mélange. Dans la section 2.1.5, nous nous attardons sur les conditions bas-Mach ; quand les écoulements se font à vitesse faible comparé à la vitesse du son. Nous passons en revue le modèle diphasique à bas nombre de Mach (DLMN), proposé lui aussi pour les conditions du nucléaire. Nous présentons enfin dans la section 2.1.6 le modèle de vibration de bulle abstraite (ABV) qui est un couplage entre une équation hyperbolique et une équation elliptique.

Puisque nous sommes intéressés par la SND des bulles, nous nous devons de choisir les bons outils pour modéliser leur interface numériquement. Nous présentons deux techniques importantes dans la section 2.2. Le suivi de front, comme expliqué dans la section 2.2.1, revient à modéliser l'écoulement avec une perspective de convection (ou « lagrangienne »). Au contraire, comme expliqué dans la section 2.2.2, nous faisons correspondre la capture d'interface à une perspective eulérienne. Nous expliquons pourquoi nous choisissons cette dernière. Notre implémentation est le transport d'une fonction couleur Y qui vaut soit 1 dans la phase gazeuse, soit 0 dans la phase liquide. Nous aurons à trouver le bon schéma de discrétisation afin de faire en sorte que le saut de 1 à 0 reste le plus abrupt possible.

C.3 Capture de front avec maillage adaptatif : schémas aux différences finies et parallélisation

Nous montrons dans le chapitre 3 que la Simulation Numérique Directe est l'échelle de calcul la plus précise et par corrolaire la plus chère. Le maillage adaptatif (AMR) est l'enrichissement d'un sous-ensemble du domaine de calcul – la région d'intérêt – avec plus de détail. Nous expliquons comment l'AMR est bénéfique à la SND en passant en revue la littérature sur le raffinement de maillage. Dans la section 3.1, nous différencions les techniques suivantes : le maillage anisotrope, l'adaptation r, l'adaptation p, l'adaptation h, l'adaptation s. Nous décidons de nous focaliser sur l'adaptation de maillage par patchs sur grilles cartésiennes dans la section 3.2. Cela signifie que nous recouvrons la dite zone d'intérêt avec un ou plusieurs niveaux de patchs, qui sont dotés d'une discrétisation spatiale plus fine. La figure C.1 montre un test standard pour les équations d'advection, le test 2D de Kothe-Rider [119]. Nous avons ici utilisé l'AMR par patchs avec un seul niveau de rafinement. Nous montrons le niveau grossier (en réalité, le domaine de calcul dans sa to-

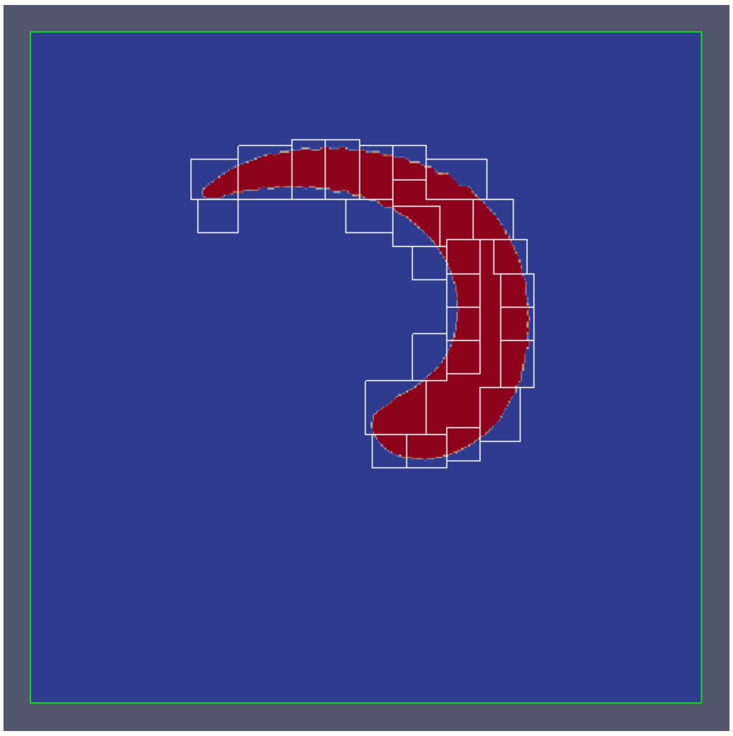


FIGURE C.1 – Test 2D de Kothe-Rider, avec patchs AMR visibles

talité) à l'intérieur d'un carré vert, il est discrétisé avec un maillage de taille 100×100. Nous montrons le niveau fin à l'intérieur de nombreux rectangles blancs, qui sont les patchs avec un coefficient de rafinement de $r = 4$. Ils recouvrent une surface que nous avons déterminée comme étant d'intérêt : l'interface entre le liquide et le gaz. Nous pouvons dire que la simulation a une discrétisation équivalente de 400×400.

Nous présentons plusieurs façons de définir les patchs ; l'algorithme de Berger-Rigoutsos [22], l'algorithme de Livne [99]. Nous proposons aussi notre propre amélioration, l'algorithme $n_{min} - n_{max}$. Il contrant la taille des patchs avec une longueur minimale n_{min} et une longueur maximale n_{max} ; un troisième paramètre étant l'efficacité minimale η_{min} du recouvrement. Nous introduisons trois fonctions de qualité destinées à comparer les recouvrements par patchs : l'efficacité moyenne η, l'écart-type normalisé des tailles σ et la quadrature moyenne γ. Nous comparons les trois algorithmes dans la perspective du calcul sur de multiples processeurs dans la section 3.3. Pour cela nous déterminons la valeur des fonctions de qualité sur quelques centaines de recouvrements qui utilisent les trois algorithmes. Nous nous déciderons pour $n_{min} - n_{max}$. Nous décidons de tester notre choix dans la section 3.4

avec une simultion d'advection 3D dite de Kothe-Rider. Nous présentons différents schémas d'advection, dont le schéma décentré amont et WENO. Nous choisissons le schéma limiteur à décentrement aval introduit Després et Lagoutière [51], car il capture et transporte la fonction couleur Y avec une diffusion minime sur un très faible nombre de mailles. En localisant les patchs de raffinement sur l'interface entre le liquide et le gaz, nous obtenons des résultalts encourgeants en ce qui concerne l'accélération de temps de calcul quand nous utilisons le calcul parralèle.

C.4 Équations elliptiques et modèle de Vibration de Bulle Abstraite

Comme nous l'expliquons dans le chapitre 4, les équations elliptiques présentent un défi supplémentaire si on cherche à les représenter avec l'AMR par patchs, puisqu'elles requièrent de l'information du domaine de calcul entier. C'est pourquoi, dans la section 4.1, nous choisissons d'utiliser l'algorithme de Correction de Défaut Local (LDC) [74]. Comme présenté schématiquement sur la figure C.2, LDC est un processus itératif qu'on réalise jusqu'à ce qu'on détermine qu'on a atteint la convergence. À chaque itération, les solutions calculées sur le maillage grossier et sur les maillages fins (les patchs) s'enrichissent les unes les autres. La solution calculée sur le maillage grossier définit des conditions aux limites de Dirichlet sur les bords des patchs. Nous résolvons ensuite l'équation elliptique sur le maillage fin avec les susnommées conditions aux limites. La solution calculée sur le maillage fin permet alors de déduire l'éponyme Correction de Défaut Local sur la zone raffinée : elle remplacera le terme source au niveau grossier, ce qui amène à une nouvelle résolution grossière et une nouvelle itération LDC.

Dans la section suivante, nous rappelons la preuve de sa convergence établie par Ferket et al. ferketreusken1996methodoncompositegrid et Anthonissen et al. [10]. Nous proposons ensuite notre variante de LDC, qui se différencie de la littérature de deux façons. Premièrement, nous utilisons des valeurs au centre des mailles et non des valeurs aux nœuds des maillages. C'est pourquoi, pour l'étape d'interpolation des conditions aux limites de Dirichlet du niveau grossier au fin, nous équipons les patchs de cellules fantômes autour de leurs bords. Deuxièmement, la plupart du temps, nous avons une situation où le recouvrement est constitué de nombreux patchs, qui souvent se touchent les uns les autres. Donc nous décidons de les considérer comme une partition du niveau fin : nous utilisons l'algorithme itératif de Schwarz de

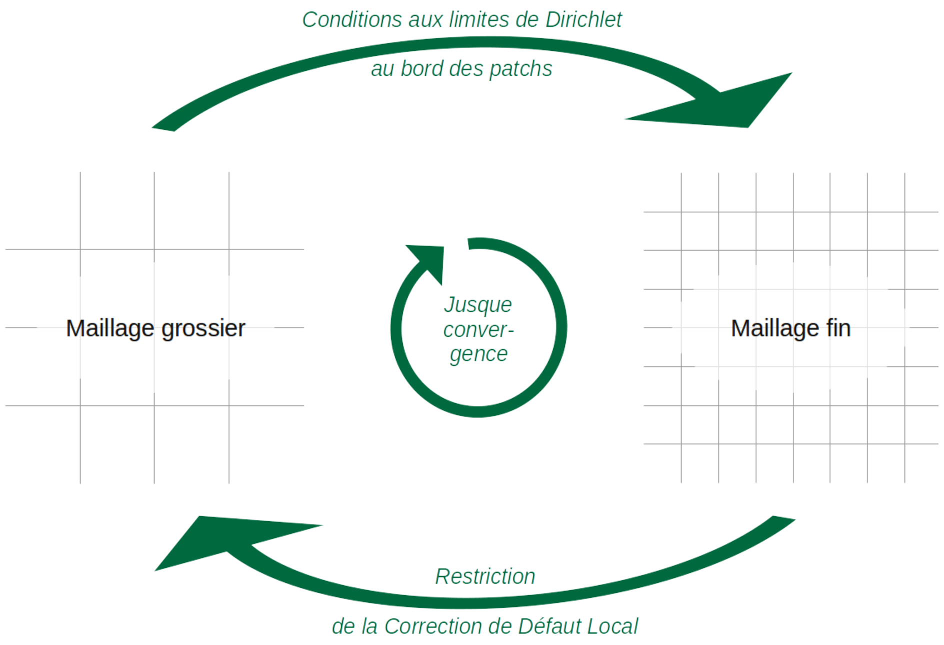


FIGURE C.2 – Schéma simplifié de l'algorithme LDC

décomposition de domaine pour déterminer une solution acceptable sur le niveau fin, sans discontinuité à l'interface entre patchs. Nous testons notre implémentation avec le modèle ABV dans la section 4.2, en 2D ainsi qu'en 3D. Nous localisons les patchs sur l'interface de la bulle. Nous utilisons une formule qui donne le volume de la bulle en fonction du temps pour vérifier que nous obtenons des résultats convaincants. Nous avons implémenté les outils AMR que nous avons utilisés dans une bibliothèque au code source libre appelée CDMATH et conçue pour aider d'autres numériciens et ingénieurs [44, 145]. Nous présentons comment CDMATH a été conçu et comment l'AMR en est une extension dans la section 4.3. Avec cette boîte à outils, chacune peut implémenter l'AMR aussi facilement en 2D et en 3D.

C.5 Application à Navier-Stokes incompressible

Enfin, nous appliquons dans le chapitre 5 le résultat de notre travail sur le maillage adaptatif à des simulations plus réalistes. Nous représentons les systèmes de Navier-Stokes incompressibles sur des maillages décallés (les variables scalaires au centre des mailles, les vecteurs sur les faces). Nous utilisons un niveau d'AMR. Dans la section 5.1, nous détaillons de façon complète les schémas numériques pour une simulation monophasique de l'équation (C.1). Dans la thèse, nous détaillons le schéma prédicteur-correcteur suivant, dans l'esprit du travail de Chorin [38, 39] et Temam [131, 132] :

$$\begin{cases} \dfrac{\tilde{\mathbf{u}} - \mathbf{u}^n}{\Delta t} + \mathbf{u}^n \cdot \nabla \tilde{\mathbf{u}} - \nu \nabla^2 \tilde{\mathbf{u}} = \mathbf{f}^n = \mathbf{g} - \dfrac{1}{\rho} \nabla p^n, \\ \tilde{\mathbf{u}} = \tilde{\mathbf{u}}_{CL} \text{ sur } \partial\Omega, \\ \tilde{\mathbf{u}}_{CL} \cdot \mathbf{n} = \mathbf{u}_{CL} \cdot \mathbf{n}, \\ \tilde{\mathbf{u}}_{CL} \cdot \mathbf{t} = \mathbf{u}_{CL} \cdot \mathbf{t} + \dfrac{\Delta t}{\rho} \nabla \phi^n \cdot \mathbf{t}. \end{cases} \tag{C.3}$$

$$\begin{cases} -\dfrac{1}{\rho} \Delta \phi^{n+1} = -\dfrac{1}{\Delta t} \nabla \cdot \tilde{\mathbf{u}}, \\ \nabla \phi^{n+1} \cdot \mathbf{n} = 0 \text{ sur } \partial\Omega. \end{cases} \tag{C.4}$$

$$p^{n+1} = p^n + \phi^{n+1}. \tag{C.5}$$

$$\begin{cases} \dfrac{\mathbf{u}^{n+1} - \tilde{\mathbf{u}}}{\Delta t} = -\dfrac{1}{\rho} \nabla \phi^{n+1}, \\ \mathbf{u}^{n+1} = \mathbf{u}_{CL} \text{ sur } \partial\Omega. \end{cases} \tag{C.6}$$

Comme expliqué plus tard, nous utilisons l'algorithme LDC pour calculer finement l'incrément de pression ϕ. Nous voyons qu'il est essentiel que le terme source de l'équation (C.4) – c'est-à-dire la divergence de la vitesse prédite $\nabla \cdot \tilde{\mathbf{u}}$ – soit interpolée linéairement du niveau grossier sur le niveau fin. Nous vérifions notre implémentation en utilisant le cas test classique dans la littérature de la cavité entraînée. Un régime permanent existe et est bien décrit : en fonction du nombre de Reynolds, plusieurs tourbillons sont créés dans le domaine de calcul, donc nous avons placé un patch de raffinement sur le tourbillon principal. La figure C.3 représente le régime permanent, avec la coloration comme fonction de la vitesse $||\mathbf{u}||$ et les isocontours de $||\mathbf{u}||$ tracés par des lignes blanches. Nous représentons la localisation du patch du niveau fin par un carré blanc. Une vidéo du régime transitoire est accessible en ligne sur `https://youtu.be/esOHN--iW4Y`.

Dans la section 5.2, nous passons à des situations diphasiques, représentées par l'équation (C.2). Nous utilisons aussi des maillages décallées, avec cependant une exception : nous localisons l'inverse $\left(\frac{1}{\rho}\right)$ de la masse volumique au

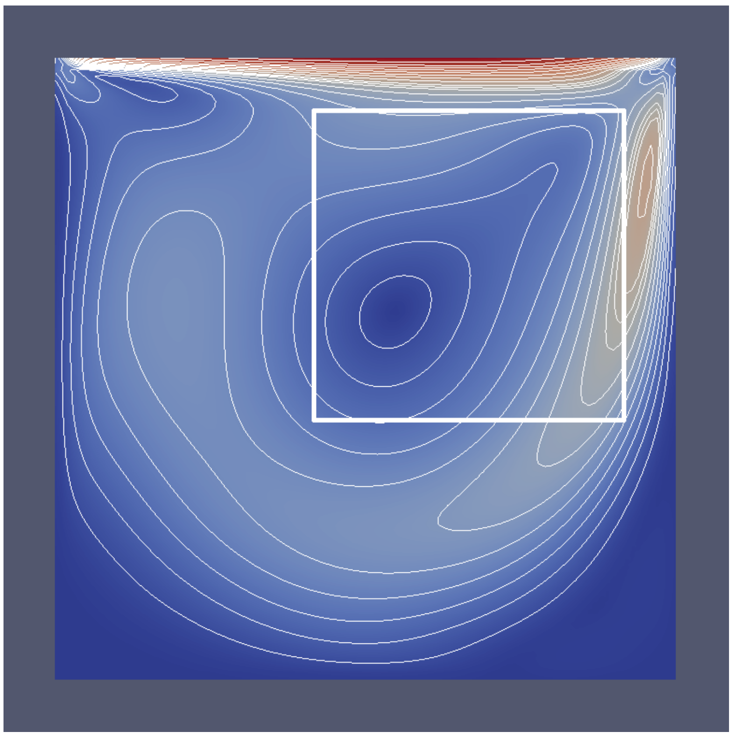


Figure C.3 – Simulation de la cavité entraînée (isocontours de $||\mathbf{u}||$)

centre des faces et non au centre des mailles, bien qu'elle soit un scalaire. Ici aussi nous utilisons un schéma prédicteur-correcteur dont les équations sont expliquées en détail dans la thèse :

$$\begin{cases} \dfrac{\tilde{\mathbf{u}} - \mathbf{u}^n}{\Delta t} + \mathbf{u}^n \cdot \nabla \tilde{\mathbf{u}} - \nu \nabla^2 \tilde{\mathbf{u}} = \mathbf{g}^{n+1} - \dfrac{1}{\rho^n} \nabla p, \\ \\ \tilde{\mathbf{u}} = \mathbf{u}_{CL} \text{ sur } \partial\Omega. \end{cases} \tag{C.7}$$

$$\begin{cases} -\nabla \cdot \left(\dfrac{1}{\rho^n} \nabla \phi^{n+1} \right) = -\dfrac{1}{\Delta t} \nabla \cdot \tilde{\mathbf{u}}, \\ \\ \nabla \phi^{n+1} \cdot \mathbf{n} = 0 \text{ sur } \partial\Omega. \end{cases} \tag{C.8}$$

$$p^{n+1} = p^n + \phi^{n+1}. \tag{C.9}$$

$$\begin{cases} \dfrac{\mathbf{u}^{n+1} - \tilde{\mathbf{u}}}{\Delta t} = -\dfrac{1}{\rho^n} \nabla \phi^{n+1}, \\ \\ \mathbf{u}^{n+1} = \mathbf{u}_{CL} \text{ sur } \partial\Omega. \end{cases} \tag{C.10}$$

$$\frac{Y^{n+1} - Y^n}{\Delta t} + \mathbf{u}^{n+1} \cdot \nabla Y^n = 0. \tag{C.11}$$

$$\rho^{n+1} = \rho_g Y^{n+1} + \rho_l (1 - Y^{n+1}). \tag{C.12}$$

Similairement à ce que nous avons fait pour la simulation monophasique, nous utilisons un niveau de rafinement adaptatif. Nous créons les patchs dynamiquement à chaque étape en temps et nous les plaçons de telle façon à ce qu'ils suivent l'interface entre le gas et le liquide, comme dans les chaptires 3 et 4. Nous proposons une approche AMR originale qui tire bénéfice du travail accompli auparavant. Nous décidons d'abord de calculer l'incrément de pression ϕ avec l'algorithme LDC appliqué à l'équation (C.8), qui est elliptique. Puis nous décidons de calculer l'advection hyperbolique de l'équation (C.11) aussi avec la précision de l'AMR. Nous expliquoins pourquoi il est essentiel que nous interpolions le terme source $\nabla \cdot \tilde{\mathbf{u}}$ de l'équation (C.8) du niveau grossier au niveau fin. Par conséquent, nous avons un calcul précis des variables ϕ, p, Y et ρ à la localisation de leurs discontinuités ou des sauts de leurs dérivées. En outre, en dépit de l'incrément brutal de la présence Y, nous sommes capables de donner un modèle satisfaisant de la tension de surface des bulles. Cela nous permet d'obtenir des évolutions réalistiques de bulles non-stationnaires en raison de la gravité, de la viscosité, de l'inertie, de la tension de surface et des forces de pression. La figure C.4 est une capture d'une telle simulation, dont la video peut être visionnée

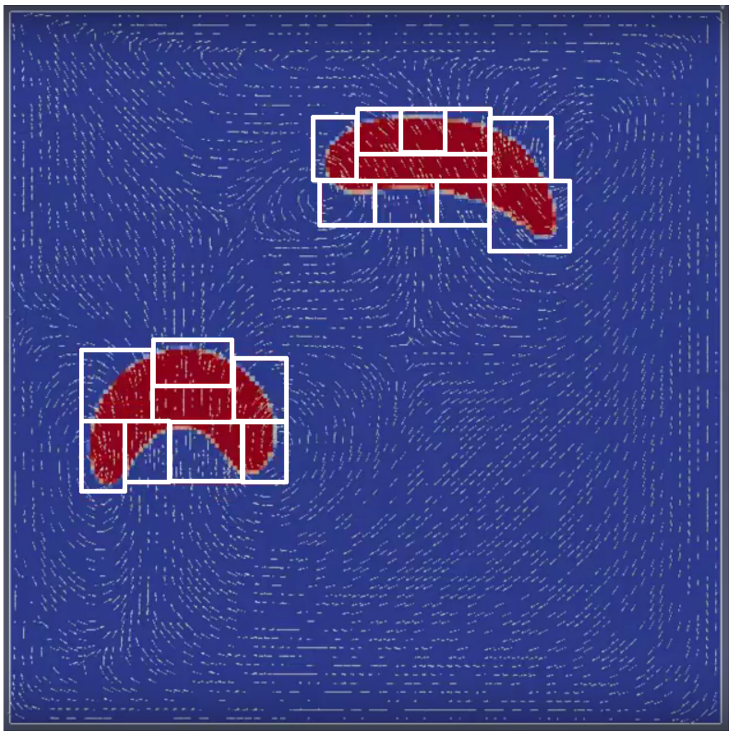


FIGURE C.4 – Simulation de deux bulles, avec les patchs AMR affichés en rectangles blancs

en ligne sur `https://youtu.be/zJEjP6JYEYQ`. Nous pouvons voir les patchs AMR comme des rectangles blancs sur les interfaces et les lignes de courant de la vitesse comme de petites flèches blanches.

C.6 Communication

Le dernier chapitre, chapitre 6 est simplement la liste de nos communications réalisées durant le doctorat, ainsi que les institutions encadrant et finançant le travail de thèse.

Bibliography

[1] Rémi Abgrall. “How to prevent pressure oscillations in multicomponent flow calculations: a quasi conservative approach”. In: *Journal of Computational Physics* 125.1 (1996), pp. 150–160 (cit. on p. 55).

[2] Rémi Abgrall and Smadar Karni. “Computations of compressible multifluids”. In: *Journal of Computational Physics* 169.2 (2001), pp. 594–623 (cit. on p. 54).

[3] Grégoire Allaire. *Analyse numérique et optimisation, une introduction à la modélisation mathématique et à la simulation numérique.* École Polytechnique, 2010. Chap. 2 (cit. on p. 86).

[4] Ann S. Almgren, John B. Bell, Mike J. Lijewski, Zarija Lukić and Ethan van Andel. “Nyx: A massively parallel AMR code for computational cosmology”. In: *The Astrophysical Journal* 765.1 (2013), p. 39 (cit. on p. 77).

[5] A.S. Almgren, J.B. Bell, C.A. Rendleman and M. Zingale. “Low Mach number modeling of type Ia supernovae. I. Hydrodynamics”. In: *The Astrophysical Journal* 637.2 (2006), p. 922 (cit. on p. 48).

[6] Annalisa Ambroso, Christophe Chalons, Frédéric Coquel and Thomas Galié. “Relaxation and numerical approximation of a two-fluid two-pressure diphasic model”. In: *ESAIM: Mathematical Modelling and Numerical Analysis* 43.6 (2009), pp. 1063–1097 (cit. on p. 46).

[7] Gene M. Amdahl. “Validity of the single processor approach to achieving large scale computing capabilities”. In: *Proceedings of the April 18-20, 1967, spring joint computer conference.* ACM. 1967, pp. 483–485 (cit. on p. 80).

[8] Ph. Angot, J.-P. Caltagirone and K. Khadra. “Une méthode adaptative de raffinement local: la correction du flux à l’interface”. In: *Comptes rendus de l’Académie des sciences. Série 1, Mathématique* 315.6 (1992), pp. 739–745 (cit. on pp. 74, 101).

[9] Martijn Johannes Hermanus Anthonissen. “Local defect correction techniques: analysis and application to combustion”. PhD thesis. Eindhoven University of Technology, Apr. 2001 (cit. on pp. 108, 109).

[10] M.J.H Anthonissen, R.M.M. Mattheij and J.H.M. ten Thije Boonkkamp. “Convergence analysis of the local defect correction method for diffusion equations”. In: *Numerische Mathematik* 95.3 (2003), pp. 401–425 (cit. on pp. 32, 105, 108, 120, 185).

[11] Klaus Umminger, Lars Dennhardt, Simon Schollenberger and Bernhard Schoen. “Integral Test Facility PKL: Experimental PWR Accident Investigation”. In: *Science and Technology of Nuclear Installations* 2012 (2012) (cit. on p. 27).

[12] Harm Askes and Antonio Rodríguez-Ferran. “A combined rh-adaptive scheme based on domain subdivision. Formulation and linear examples”. In: *International Journal for Numerical Methods in Engineering* 51.3 (2001), pp. 253–273 (cit. on p. 65).

[13] Arnoldo Badillo. “Quantitative phase-field modeling for boiling phenomena”. In: *Physical Review E* 86.4 (2012), p. 041603 (cit. on p. 55).

[14] M.R. Baer and J.W. Nunziato. “A two-phase mixture theory for the deflagration-to-detonation transition (DDT) in reactive granular materials”. In: *Int. J. Multiphase Flow* 12.6 (1986), pp. 861–889 (cit. on p. 46).

[15] Satish Balay, William D. Gropp, Lois Curfman McInnes and Barry F. Smith. “Efficient Management of Parallelism in Object Oriented Numerical Software Libraries”. In: *Modern Software Tools in Scientific Computing*. Ed. by E. Arge, A. M. Bruaset and H. P. Langtangen. Birkhäuser Press, 1997, pp. 163–202 (cit. on pp. 124, 130).

[16] Satish Balay et al. *PETSc Users Manual*. ANL-95/11 – Revision 3.5. 2014. URL: `http://www.mcs.anl.gov/petsc` (cit. on pp. 124, 130).

[17] Satish Balay et al. *PETSc Web page*. 2014. URL: `http://www.mcs.anl.gov/petsc` (cit. on pp. 124, 130).

[18] L. Barbié, I. Ramière and F. Lebon. “Strategies involving the local defect correction multi-level refinement method for solving three-dimensional linear elastic problems”. In: *Computers & Structures* 130 (2014), pp. 73–90 (cit. on p. 59).

[19] F.B. Barros, S.P.B. Proenca and C.S. de Barcellos. “Generalized finite element method in structural nonlinear analysis–a p-adaptive strategy”. In: *Computational Mechanics* 33.2 (2004), pp. 95–107 (cit. on pp. 62, 63).

[20] M. J. Berger and P. Collela. “Local adaptive mesh refinement for hyperbolic partial differential equations”. In: *Journal of Computational Physics* 82 (1989), pp. 64–84 (cit. on pp. 66, 69).

[21] Marsha J. Berger and Joseph Oliger. “Adpative mesh refinement for hyperbolic partial equations”. In: *Journal of Computational Physics* 53.3 (1984), pp. 484–512 (cit. on pp. 66, 69).

[22] Marsha J. Berger and Isidore Rigoutsos. “An Algorithm for Point Clustering and Grid Generation”. In: *IEEE Transactions Systems, Man and Cybernetics* 21.5 (Sept. 1991), pp. 1278–1286 (cit. on pp. 31, 69, 184).

[23] Manuel Bernard, Stéphane Dellacherie, Gloria Faccanoni, Bérénice Grec, Olivier Lafitte, Tan-Trung Nguyen and Yohan Penel. “Study of a low Mach nuclear core model for single-phase flows”. In: *ESAIM: Proceedings*. Vol. 38. EDP Sciences. 2012, pp. 118–134 (cit. on p. 129).

[24] Aude Bernard-Champmartin and Florian De Vuyst. “A low diffusive Lagrandge-Remap scheme for the simulation of violent air-water free-surface flows”. In: *Journal of Computational Physics* 274 (2014), pp. 19–49 (cit. on p. 90).

[25] Ray A. Berry, John W. Peterson, Hongbin Zhang, Richard C. Martineau, Haihua Zhao, Ling Zou and David Andrs. *Relap-7 theory manual*. Idaho National Laboratory, Tech. Rep. INL/EXT-14-31366. 2014 (cit. on p. 28).

[26] G.C. Bessette, E.B. Becker, L.M. Taylor and D.L. Littlefield. “Modeling of impact problems using an h-adaptive, explicit lagrangian finite element method in three dimensions”. In: *Computer methods in applied mechanics and engineering* 192.13 (2003), pp. 1649–1679 (cit. on p. 64).

[27] Dominique Bestion. “4: Modèles à l’échelle 3D locale pour les écoulements bouillants et la prédiction du DNB”. INSTN course “Thermohydraulique diphasique dans les réacteurs”. Mar. 2014 (cit. on p. 27).

[28] Dominique Bestion. “The physical closure laws in the CATHARE code”. In: *Nuclear Engineering and Design* 124.3 (1990), pp. 229–245 (cit. on pp. 28, 45).

[29] Ulrich Bieder, Valérie Barthel, Frédéric Ducros, Patrick Quéméré and Simone Vandroux. “CFD calculations of wire wrapped fuel bundles: modeling and validation strategies”. In: *CFD4NRS-3, Bethesda, USA* (2010) (cit. on p. 28).

[30] Guillaume Bois. “Heat and mass transfers at liquid/vapor interfaces with phase-change: proposal for a large-scale modeling of interfaces”. PhD thesis. Université de Grenoble, Feb. 2011 (cit. on pp. 28, 52, 55).

[31] O. Botella and R. Peyret. “Benchmark spectral results on the lid-driven cavity flow”. In: *Computers & Fluids* 27.4 (1998), pp. 421–433 (cit. on pp. 150, 151).

[32] Franck Boyer, Céline Lapuerta, Sebastian Minjeaud and Bruno Piar. “A local adaptive refinement method with multigrid preconditionning illustrated by multiphase flows simulations”. In: *ESAIM: Proceedings*. Vol. 27. EDP Sciences. 2009, pp. 15–53 (cit. on p. 67).

[33] J.U. Brackbill, D.B. Kothe and C. Zemach. “A Continuum Method for Modeling Surface Tension”. In: *Journal of Computational Physics* 100 (1992), pp. 335–354 (cit. on p. 162).

[34] M.J. Burwell, G. Lerchl, J. Miro, V. Teschendorff and K. Wolfert. “The thermalhydraulic code ATHLET for analysis of PWR and BWR systems”. In: *Fourth international topical meeting on nuclear reactor thermal-hydraulics (NURETH-4). Proceedings. Vol. 2.* 1989 (cit. on p. 28).

[35] Weiming Cao, Weizhang Huang and Robert D Russell. “Comparison of two-dimensional r-adaptive finite element methods using various error indicators”. In: *Mathematics and Computers in Simulation* 56.2 (2001), pp. 127–143 (cit. on p. 60).

[36] Arnab Chaudhuri, Abdellah Hadjadj, Ashwin Chinnayya and Sandrine Palerm. “Numerical study of compressible mixing layers using high-order WENO schemes”. In: *Journal of Scientific Computing* 47.2 (2011), pp. 170–197 (cit. on p. 89).

[37] David L. Chopp. “Some improvements of the fast marching method”. In: *SIAM Journal on Scientific Computing* 23.1 (2001), pp. 230–244 (cit. on p. 54).

[38] Alexandre Joel Chorin. “A numerical method for solving incompressible viscous flow problems”. In: *Journal of Computational Physics* 2 (1967), pp. 12–26 (cit. on pp. 34, 133, 135, 187).

[39] Alexandre Joel Chorin. “Numerical solution of the Navier-Stokes equations”. In: *Mathematics of computation* 22.104 (1968), pp. 745–762 (cit. on pp. 34, 133, 135, 187).

[40] R. Clift, J. R. Grace and M. E. Weber. *Bubbles, Drops, and Particles.* 111 Fifth Avenue, New York, New York 10003: Academic Press, 1978 (cit. on p. 29).

[41] Frédéric Coquel, Jean-Marc Hérard, Khaled Saleh and Nicolas Seguin. “A robust entropy-satisfying finite volume scheme for the isentropic Baer-Nunziato model”. In: *ESAIM: Mathematical Modelling and Numerical Analysis* 48.1 (2014), pp. 165–206 (cit. on p. 46).

[42] Tim Cramer, Dirk Schmidl, Michael Klemm and Dieter an Mey. “OpenMP Programming on Intel® Xeon Phi™ Coprocessors: An Early Performance Comparison”. In: (2012) (cit. on p. 77).

[43] F. Dabbene, J. Brinster, D. Abdo, E. Porcheron, P. Lemaitre, G. Mignot, R. Kapulla, S. Paranjape, M. Kamnev and A. Khizbullin. “Experimental Activities On Stratification And Mixing of a Gas Mixture under the Conditions of a Severe Accident with Intervention Of Mitigating Measures Performed in the ERCOSAM-SAMARA Projects”. In: *ICAPP 2015*. ICAPP. Nice, France, May 2015 (cit. on p. 27).

[44] S. Dellacherie, K. Herilus, J. Jung, A. Mekkas and A. Talpaert. “Simulation numérique de systèmes de loi de conservation dans un environnement SALOME/C++ *via* la bibliothèque CDMATH”. In preparation (cit. on pp. 33, 129, 186).

[45] Stéphane Dellacherie. “Étude et discrétisation de modèles cinétiques et de modèles fluides à bas nombre de Mach”. Mémoire en vue d’obtenir l’Habilitation à Diriger des Recherches en Mathématiques. Université Pierre et Marie Curie – Paris VI, Feb. 2011 (cit. on p. 46).

[46] Stéphane Dellacherie. “Numerical resolution of a potential diphasic low Mach number system”. In: *Journal of Computational Physics* 39.3 (2007). Ed. by SMAI EDP Sciences, pp. 487–514 (cit. on pp. 43, 48, 90).

[47] Stéphane Dellacherie. “On a diphasic low Mach number system”. In: *ESAIM: M2AN* 223 (2005), pp. 151–187 (cit. on pp. 48, 167).

[48] Stéphane Dellacherie. “On a low-Mach nuclear core model”. In: *ESAIM: Proceedings*. Vol. 35. EDP Sciences, 17, Avenue du Hoggar, Parc d’Activité de Courtabœuf, BP 112, F-91944 Les Ulis Cedex A, France: EDP Sciences, SMAI, Mar. 2012, pp. 79–106 (cit. on pp. 48, 129).

[49] Stéphane Dellacherie, Gloria Faccanoni, Bérénice Grec, Ethem Nayir and Yohan Penel. “2D numerical simulation of a low Mach nuclear core model with stiffened gas using Freefem++”. In: *ESAIM: Proceedings and Surveys* 45 (2014), pp. 138–147 (cit. on p. 129).

[50] Stéphane Dellacherie and Olivier Lafitte. *Existence et unicité d'une solution classique à un modèle abstrait de vibration de bulles de type hyperbolique-elliptique.* Tech. rep. CRM-3200. CRM, Montréal, Canada: Centre de Recherches Mathématiques, 2005 (cit. on pp. 50, 51).

[51] Bruno Després and Frédéric Lagoutière. "Contact Discontinuity Capturing Schemes for Linear Advection and Compressible Gas Dynamics". In: *Journal of Scientific Computing* 16.4 (2002), pp. 479–524 (cit. on pp. 32, 89, 90, 185).

[52] Bruno Després, Frédéric Lagoutière, Emmanuel Labourasse and Isabelle Marmajou. "An antidissipative transport scheme on unstructured meshes for multicomponent flows". In: *The International Journal on Finite Volumes* (2010), pp. 30–65 (cit. on p. 90).

[53] Hachmi Ben Dhia. "Problèmes mécaniques multi-échelles: la méthode Arlequin". In: *Comptes Rendus de l'Académie des Sciences-Series IIB-Mechanics-Physics-Astronomy* 326.12 (1998), pp. 899–904 (cit. on p. 65).

[54] Stéphanie Delage-Santacreu, Stéphane Vincent and Jean-Paul Caltagirone. "Tracking Fronts in One and Two-phase Incompressible Flows Using an Adaptive Mesh Refinement Approach". In: *Journal of Scientific Computing* 41 (2009), pp. 221–237 (cit. on pp. 80, 100).

[55] John Dolbow. "Level Sets I". CEA-EDF-INRIA Numerical Analysis Summer School. June 2014 (cit. on pp. 51, 52, 55).

[56] John Dolbow. "Level Sets II". CEA-EDF-INRIA Numerical Analysis Summer School. June 2014 (cit. on p. 53).

[57] Komla Domelevo and Pascal Omnes. "A finite volume method for the Laplace equation on almost arbitrary two-dimensional grids". In: *ESAIM: Mathematical Modelling and Numerical Analysis* 39.06 (2005), pp. 1203–1249 (cit. on p. 178).

[58] J Donea, Antonio Huerta, J-Ph Ponthot and A Rodriguez-Ferran. "Encyclopedia of Computational Mechanics". In: vol. 1: Fundamentals. Wiley & Sons, 2004. Chap. 14: Arbitrary Lagrangian-Eulerian Methods (cit. on p. 62).

[59] EDF, CEA and OPEN CASCADE. *SALOME Platform.* Accessed: 2015-10-14. 2005 – 2015. URL: `http://salome-platform.org/` (cit. on p. 130).

[60] Douglas Enright, Ronald Fedkiw, Joel Ferziger and Ian Mitchell. “A hybrid particle level set method for improved interface capturing”. In: *Journal of Computational Physics* 183.1 (2002), pp. 83–116 (cit. on p. 54).

[61] *Extended Finite Element Method*. URL: `http://www-2.unipv.it/compmech/xfem.html` (cit. on p. 54).

[62] Gloria Faccanoni. “Étude d’un modèle fin de changement de phase liquide-vapeur. Contribution à l’étude de la crise d’ébullition”. Thèse en cotutelle entre l’École Polytechnique et l’Università di Trento, Italie. Thèse. École Polytechnique X, Nov. 2008. URL: `http://hal.archives-ouvertes.fr/tel-00363460` (cit. on pp. 26, 28, 43).

[63] Peter J.J. Ferket and Arnold A. Reusken. “A finite difference discretization method for elliptic problems on composite grids”. In: *Computing* 56.4 (1996), pp. 343–369 (cit. on pp. 32, 106).

[64] Peter J.J. Ferket and Arnold A. Reusken. “Further analysis of the Local Defect Correction method”. In: *Computing* 56.2 (1996), pp. 117–139 (cit. on pp. 102, 106).

[65] Alexandru Fikl. *Adaptive Mesh Refinement with p4est*. Tech. rep. Digiteo Labs - bât. 565 - PC 190, CEA Saclay, 91191 Gif-sur-Yvette cedex: Sup Galilée, CEA, Maison de la Simulation, Oct. 2014 (cit. on pp. 64, 67).

[66] J. Fish. “The s-version of the finite element method”. In: *Computers & Structures* 43.3 (1992), pp. 539–547 (cit. on pp. 65, 66).

[67] Alain Forestier. “Cours sur les écoulements diphasiques, Troisième partie”. 2013 (cit. on p. 46).

[68] Charles Fribourg. “Réacteurs nucléaires de propulsion navale”. In: *Techniques de l’Ingénieur* BN3141 V1 (Jan. 2002) (cit. on p. 27).

[69] Marie Billaud Friess and Samuel Kokh. “An anti-diffusive Lagrange-remap scheme for multi-material compressible flows with an arbitrary number of components”. In: *ESAIM: Proceedings*. Vol. 35. EDP Sciences. 2012, pp. 203–209 (cit. on p. 168).

[70] Jean-Frédéric Gerbeau, Claude Le Bris and Tony Lelièvre. “Mathematical methods for the Magnetohydrodynamics of liquid metals”. In: Oxford University Press, 2006. Chap. 5: Numerical simulation of two-fluid problems (cit. on p. 62).

[71] U. Ghia, K.N. Ghia and C.T. Shin. “High-Re Solutions for Incompressible Flow Using the Navier-Stokes Equations and a Multigrid Method”. In: *Journal of Computational Physics* 48 (Jan. 1982), pp. 387–411 (cit. on p. 151).

[72] S. Ghosh and S.K. Manna. “r-adapted arbitrary Lagrangian-Eulerian finite-element method in metal-forming simulation”. In: *Journal of materials engineering and performance* 2.2 (1993), pp. 271–282 (cit. on p. 62).

[73] Hans-Peter Gittel, Matthias Günther and Gerhard Ströhmer. “Remarks on a nonlinear transport problem”. In: *Journal of Differential Equations* 256.3 (2014), pp. 957–988. ISSN: 0022-0396 (cit. on p. 50).

[74] W. Hackbusch. “Local defect correction method and domain decomposition techniques”. In: *Computing [Suppl.]* 5 (1984), pp. 89–113 (cit. on pp. 32, 101, 185).

[75] Wolfgang Hackbusch. *Integral equations: theory and numerical treatment.* Vol. 120. Birkhäuser, 2012 (cit. on p. 115).

[76] Manuel Bernard, Stéphane Dellacherie, Gloria Faccanoni, Bérénice Grec and Yohan Penel. “Study of a low Mach nuclear core model for two-phase flows with phase transition I: stiffened gas law”. In: *ESAIM: Mathematical Modelling and Numerical Analysis* 48 (2014), pp. 1639–1679 (cit. on pp. 48, 129).

[77] Francis H. Harlow and J. Eddie Welch. “Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface”. In: *Physics of fluids* 8.12 (1965), p. 2182 (cit. on p. 133).

[78] A. Harten, B. Engquist, S. Osher and S. Chakravarthy. “Uniformly high-order accurate non-oscillatory schemes III”. In: *Journal of Computational Physics* 71 (1987), pp. 231–303 (cit. on p. 87).

[79] A. Harten and S. Osher. “Uniformly high-order accurate non-oscillatory schemes I”. In: *SIAM Journal of Numerical Analysis* 24 (1987), pp. 279–309 (cit. on p. 87).

[80] Philippe Helluy and Nicolas Seguin. “Relaxation models of phase transition flows”. In: *ESAIM: Mathematical Modelling and Numerical Analysis* 40.2 (2006), pp. 331–352 (cit. on p. 169).

[81] Jean-Marc Hérard and Jonathan Jung. “An interface condition to compute compressible flows in variable cross section ducts”. In: *Comptes Rendus Mathématique* 354.3 (2016), pp. 323–327 (cit. on p. 130).

[82] Ikosaeder. *Schematic description of PLIC reconstruction.* Mar. 2014. URL: `https://commons.wikimedia.org/wiki/File:Vof_scheme_with_plic.png` (cit. on p. 56).

[83] D. Jamet and B. Mathieu. "Simulation numérique directe des écoulements diphasiques". INSTN course "Thermohydraulique diphasique dans les réacteurs nucléaires". Mar. 2014 (cit. on p. 28).

[84] Jean-Christophe Jouhaud. "Méthode d'adaptation de maillages structurés par enrichissement, application à la résolution des équations d'Euler et de Navier-Stokes". PhD thesis. Université Bordeaux I, Oct. 1997 (cit. on p. 66).

[85] Damir Juric and Grétar Tryggvason. "A front-tracking method for dendritic solidification". In: *Journal of Computational Physics* 123.1 (1996), pp. 127–148 (cit. on p. 169).

[86] R.E. Kelly. "The stability of an unsteady Kelvin-Helmholtz flow". In: *Journal of Fluid Mechanics* 22.03 (1965), pp. 547–560 (cit. on p. 168).

[87] K. Khadra, Ph. Angot, J.P. Caltagirone and P. Morel. "Concept de zoom adaptatif en architecture multigrille locale; étude comparative des méthodes LDC, FAC et FIC". In: *Modélisation mathématique et analyse numérique* 30.1 (1996), pp. 39–82 (cit. on p. 74).

[88] Khodor Khadra, Philippe Angot and Jean-Paul Caltagirone. "A comparison of locally adaptive multigrid methods: LDC, FAC AND FIC". In: (1993) (cit. on p. 74).

[89] Samuel Kokh. "Aspects numériques et théoriques de la modélisation des écoulements diphasiques compressibles par des méthodes de capture d'interface". PhD thesis. Université de Paris VI, 2001 (cit. on p. 43).

[90] Samuel Kokh and Grégoire Allaire. "Numerical simulation of 2-D two-phase flows with interface". In: *Godunov Methods.* Springer, 2001, pp. 513–518 (cit. on p. 57).

[91] Samuel Kokh and Frédéric Lagoutière. "An anti-diffusive numerical scheme for the simulation of interfaces between compressible fluids by means of a five-equation model". In: *Journal of Computational Physics* 229.8 (2010), pp. 2773–2809 (cit. on p. 90).

[92] Tomasz Kozlowski, Aaron Wysocki, Ivan Gajev, Yunlin Xu, Thomas Downar, Kostadin Ivanov, Jeffrey Magedanz, Matthew Hardgrove, Jose March-Leuba, Nathanael Hudson et al. "Analysis of the OECD/NRC Oskarshamn-2 BWR stability benchmark". In: *Annals of Nuclear Energy* 67 (2014), pp. 4–12 (cit. on p. 27).

[93] Alexei Kudryavtsev and Abdellah Hadjadj. “Visualisation graphique en mécanique des fluides numérique”. In: *CR de 9ème Colloque Francophone de Visualisation et de Traitement d’Images en Mécanique des Fluides – FLUVISU*. June 2001, pp. 55–62 (cit. on p. 87).

[94] Frédéric Lagoutière. “Modélisation mathématique et résolution numérique de problèmes de fluides compressibles à plusieurs constituants”. Organisme d’accueil : Commissariat à l’Énergie Atomique, centre de Bruyères-le-Châtel. PhD thesis. Université Pierre et Marie Curie, Dec. 2000 (cit. on pp. 55, 90).

[95] Xu-Dong Liu, Stanley Osher and Tony Chan. “Weigthed essentially non-oscillatory schemes”. In: *Journal of Computational Physics* 115.1 (1994), pp. 200–212 (cit. on p. 87).

[96] Oren E. Livne. *Clustering on Single Refinement Level: Berger-Rigoustos Algorithm*. Tech. rep. UUSCI-2006-001. Scientific Computing and Imaging Institute, University of Utah, Salt Lake City, UT 84112, USA: University of Utah, Jan. 2006 (cit. on pp. 69, 70).

[97] Oren E. Livne. *Clustering Patches for Multi-Level Refined Grids: a Hierarchical Approach*. Tech. rep. UUSCI-2006-004. Scientific Computing and Imaging Institute, University of Utah, Salt Lake City, UT 84112, USA: University of Utah, Jan. 2006 (cit. on p. 74).

[98] Oren E. Livne. *Efficient Update of Persistent Patches in the Berger Rigoutsous Algorithm*. Tech. rep. UUSCI-2006-003. Scientific Computing and Imaging Institute, University of Utah, Salt Lake City, UT 84112, USA: University of Utah, Jan. 2006 (cit. on p. 73).

[99] Oren E. Livne. *Minimum and Maximum Patch Size Clustering on a Single Refinement Level*. Tech. rep. UUSCI-2006-002. Scientific Computing and Imaging Institute, University of Utah, Salt Lake City, UT 84112, USA: University of Utah, Jan. 2006 (cit. on pp. 31, 70, 184).

[100] A. Majda. *Equations for low Mach number combustion*. Tech. rep. 112. Berkeley, California: University of California at Berkeley, 1982 (cit. on p. 48).

[101] Stephen F. McCormick. “Multilevel adaptive methods for partial differential equations”. In: *SIAM* 6 (1989) (cit. on pp. 74, 101).

[102] Steve McCormick. “Fast adaptive composite grid (FAC) methods: theory for the variational case”. In: *Computing [Suppl.]* 5 (1984). Ed. by Springer Vienna, pp. 115–121 (cit. on pp. 74, 101).

[103] Steve McCormick and Jim Thomas. “The fast adaptve composite grid (FAC) method for elliptic equations”. In: *Mathematics of Computation* 46 (1986), pp. 439–456 (cit. on pp. 74, 101).

[104] Anouar Mekkas. “Résolution numérique d’un modèle de vibration de bulle abstraite”. CEA, ONERA. MA thesis. SupGalilée, Centre Mathématique et Informatique, École Centrale Marseille, July 2008 (cit. on pp. 85, 123).

[105] Victorien Menier. “3D parallel anisotropic unsteady mesh adaptation for the high-fidelity prediction of bubble motion”. In: *ESAIM: Proceedings and Surveys* 50 (Mar. 2015), pp. 169–188 (cit. on pp. 60, 61, 85).

[106] Sebastian Minjeaud, Frank Boyer, Céline Lapuerta and Bruno Piar. *Local refinement and multigrid preconditioning for a ternary Cahn-Hilliard/Navier-Stokes model.* Poster at the CANUM 2014 Congress in Carry-Le-Rouet. Apr. 2014 (cit. on pp. 64, 65, 67).

[107] Gordon E. Moore. “Cramming more components onto integrated circuits”. In: *Proceedings of the IEEE* 86.1 (Jan. 1998), pp. 82–85 (cit. on p. 76).

[108] William F. Noh and Paul Woodward. “SLIC (Simple Line Interface Calculation)”. In: *Proceedings of the Fifth International Conference on Numerical Methods in Fluid Dynamics June 28–July 2, 1976 Twente University, Enschede.* Springer. 1976, pp. 330–340 (cit. on p. 55).

[109] P. Obry, J.L. Cheissoux, M. Grandotto, J.P. Gaillard, E. De Langre and M. Bernard. “An advanced steam generator design 3D code, Thermal Hydraulics of Advanced Heat Exchangers”. In: *Proceedings of the 1990 ASME Winter Annual Meeting, Dallas, Texas, USA.* 1990, pp. 15–21 (cit. on p. 28).

[110] Stanley Osher and James A. Sethian. “Fronts Propagating with Curvature Dependent Speed: Algorithms Based on Hamilton-Jacobi Formulations”. In: *Journal of Computational Physics* 79 (1988), pp. 12–49 (cit. on p. 53).

[111] Maria Elena Pasquini. “Experimental study of vapor bubble dynamics”. PhD thesis. Université Pierre et Marie Curie – Paris VI, May 2015. URL: `https://tel.archives-ouvertes.fr/tel-01208170` (cit. on p. 169).

[112] M.E. Pasquini, B. Cariteau, C. Josserand and P. Salvatore. “Experimental study on subcooled nucleate boiling from a single artificial cavity”. In: International Conference on Heat Transfer, Fluid Mechanics and Thermodynamics. 2014 (cit. on p. 169).

[113] Yohan Penel. “Étude théorique et numérique de la déformation d’une interface séparant deux fluides non-miscibles à bas nombre de Mach”. Organisme d’accueil : Commissariat à l’Énergie Atomique, centre de Saclay. Thèse. Université Paris 13, Dec. 2010 (cit. on pp. 48, 50).

[114] Yohan Penel. “Well-posedness of a low Mach number system”. In: *C. R. Acad. Sci.* 1.350 (2012), pp. 51–55 (cit. on p. 48).

[115] Yohan Penel, Stéphane Dellacherie and Olivier Lafitte. “Theoretical Study of an Abstract Bubble Vibration Model”. In: *Zeitschrift für Analysis und Ihre Anwendungen* 32.1 (2013), pp. 19–36 (cit. on p. 50).

[116] Yohan Penel, Anouar Mekkas, Stéphane Dellacherie, Juliet Ryan and Michel Borrel. “Application of an AMR strategy to an abstract bubble vibration model”. In: *19th AIAA Comp. Fluid Dyn. Conf Proc* (June 2009) (cit. on p. 123).

[117] Marc Pérache. “Contribution à l’élaboration d’environnements de programmation dédiés au calcul scientifique hautes performances”. PhD thesis. Université Bordeaux 1, Oct. 2006 (cit. on p. 167).

[118] Guillaume Prigent. “Modélisation et simulation numérique d’écoulements diphasiques pour la microfluidique”. PhD thesis. Université Paris-Sud, Jan. 2013 (cit. on p. 48).

[119] William J. Rider and Douglas B. Kothe. “Reconstructing volume tracking”. In: *Journal of Computational Physics* 141.2 (Apr. 1998), pp. 112–152 (cit. on pp. 31, 84, 183).

[120] Philip L. Roe. “Some contributions to the modelling of discontinuous flows”. In: *Large-scale computations in fluid mechanics.* Vol. 1. 1985, pp. 163–193 (cit. on pp. 86, 90).

[121] Yohei Sato and Bojan Ničeno. “A conservative local interface sharpening scheme for the constrained interpolation profile method”. In: *International Journal for Numerical Methods in Fluids* 70.4 (2012), pp. 441–467 (cit. on p. 55).

[122] *SATURN version 3.0 tutorial – Shear driven cavity flow.* EDF R&D, Fluid Dynamics, Power Generation and Environment Department, Single Phase Thermal-Hydraulics Group. 6 quai Watier, F-78401 Chatou cedex, May 2013 (cit. on pp. 150, 151).

[123] Thomas P. Scholcz, Alexander H. van Zuijlen and Hester Bijl. "A multi-model incremental adaptive strategy to accelerate partitioned fluid-structure algorithms using space-mapping". In: *Coupled Problems 2011: Proceedings of the 4th International Conference on Computational Methods for Coupled Problems in Science and Engineering, Kos, Greece, 20-22 June 2011*. CIMNE. 2011 (cit. on p. 76).

[124] David H. Sharp. "An overview of Rayleigh-Taylor instability". In: *Physica D: Non-linear Phenomena* 12.1 (1984), pp. 3–18 (cit. on p. 168).

[125] Chi-Wang Shu and Stanley Osher. "Efficient implementation of Essentially Non-Oscillatory shock-capturing schemes". In: *Journal of Computational Physics* 77 (1988), pp. 439–471 (cit. on p. 87).

[126] Chi-Wang Shu and Stanley Osher. "Efficient implementation of Essentially Non-Oscillatory shock-capturing schemes, II". In: *Journal of Computational Physics* 83 (1989), pp. 32–78 (cit. on p. 87).

[127] Mark Sussman, Kayne M. Smith, M. Yousuff Hussaini, Mitsuhiro Ohta and R. Zhi-Wei. "A sharp interface method for incompressible two-phase flows". In: *Journal of Computational Physics* 221.2 (2007), pp. 469–505 (cit. on p. 163).

[128] Arthur Talpaert. *Analysis of interfacial forces on the physics of two-phase flow and hyperbolicity of the two-fluid model*. 2013 (cit. on p. 46).

[129] Arthur Talpaert, Grégoire Allaire, Stéphane Dellacherie and Anouar Mekkas. *Direct Numerical Simulation of bubbles using Adaptive Mesh Refinement on parallel architecture*. Poster at the CEA-GAMNI seminar at IHP in Paris. Feb. 2015 (cit. on p. 96).

[130] Arthur Talpaert, Stéphane Dellacherie and Anouar Mekkas. *Analysis of the efficiency and relevance of patch creation algorithms in the case of AMR of thin flagged areas*. Poster at the CANUM 2014 Congress in Carry-Le-Rouet. Apr. 2014 (cit. on p. 71).

[131] Roger Temam. "Sur l'approximation de la solution des équations de Navier-Stokes par la méthode des pas fractionnaires (I)". In: *Archive for Rational Mechanics and Analysis* 32.2 (1969), pp. 135–153 (cit. on pp. 34, 133, 135, 187).

[132] Roger Temam. "Sur l'approximation de la solution des équations de Navier-Stokes par la méthode des pas fractionnaires (II)". In: *Archive for Rational Mechanics and Analysis* 33.5 (1969), pp. 377–385 (cit. on pp. 34, 133, 135, 187).

[133] Abhijat. S. Tilak. “A Pressure Based Unstructured Grid Method for Fluid Flow and Heat Transfer”. In: *Eleventh annual conference of the CFD Society of Canada (CFD 2003). Proceedings.* May 2003 (cit. on pp. 150, 151).

[134] I. Tkatschenko, E. Studer, J.-P. Magnaud, L. Blumenfeld, H. Simon and H. Paillère. “Status of the MISTRA programme for the validation of containment thermal-hydraulic codes”. In: (2005) (cit. on p. 27).

[135] Neil E Todreas and Mujid S Kazimi. *Nuclear systems: thermal hydraulic fundamentals.* Vol. 1. CRC Press, 2012 (cit. on p. 28).

[136] Eleuterio F. Toro. *Riemann solvers and numerical methods for fluid dynamics: a practical introduction.* 3rd ed. Springer Science & Business Media, 2009 (cit. on pp. 86, 90).

[137] I. Toumi, A. Bergeron, D. Gallo, E. Royer and D. Caruge. “FLICA-4: a three-dimensional two-phase flow computer code with advanced numerical methods for nuclear applications”. In: *Nuclear Engineering and Design* 200.1 (2000), pp. 139–155 (cit. on p. 28).

[138] Grétar Tryggvason, Asghar Esmaeeli, Jiacai Lu and Souvik Biswas. “Direct numerical simulations of gas/liquid multiphase flows”. In: *Fluid dynamics research* 38.9 (2006), pp. 660–681 (cit. on p. 51).

[139] Grétar Tryggvason, Ruben Scardovelli and Stéphane Zaleski. *Direct numerical simulations of gas–liquid multiphase flows.* Cambridge University Press, 2011 (cit. on p. 51).

[140] Salih Ozen Unverdi and Grétar Tryggvason. “A front-tracking method for viscous, incompressible, multi-fluid flows”. In: *Journal of computational physics* 100.1 (1992), pp. 25–37 (cit. on p. 51).

[141] José Valenciano and Robert G. Owens. “An h–p adaptive spectral element method for Stokes flow”. In: *Applied Numerical Mathematics* 33.1 (2000), pp. 365–371 (cit. on p. 65).

[142] Mathias Viellieber and Andreas G. Class. “Sub channel scale resolution in homogenization codes by the Coarse-Grid-CFD”. In: *THINS 2014 International Workshop.* THINS. Modena, Italy, Jan. 2014 (cit. on p. 76).

[143] J.-P. Vila. “High-order schemes and entropy condition for nonlinear hyperbolic systems of conservation laws”. In: *Mathematics of computation* 50.181 (1988), pp. 53–73 (cit. on p. 87).

[144] M. M. El-Wakil. *Nuclear Heat Transport.* Third. 555 North Kensington Avenue, La Grange Park, Illinois 60525, USA: The American Nuclear Society, May 1978 (cit. on p. 27).

[145] CDMATH workgroup. *CDMATH source repository.* Sept. 2016. URL: `https://github.com/PROJECT-CDMATH/CDMATH` (cit. on pp. 33, 129, 186).

[146] Stéphane Zaleski, Jie Li and Sauro Succi. "Two-dimensional Navier-Stokes simulation of deformation and breakup of liquid patches". In: *Physical review letters* 75.2 (1995), p. 244 (cit. on p. 55).

[147] D. Zuzio and J.L. Estivalezes. "An efficient block parallel AMR method for two phase interfacial flow simulations". In: *Computers & Fluids* 44.1 (2011), pp. 339–357 (cit. on p. 167).

Titre : Simulation numérique directe de bulles sur maillage adaptatif avec algorithmes distribués

Mots clefs : Thermohydraulique nucléaire, écoulements diphasiques, simulation numérique de bulles, conditions bas-Mach, maillage adaptatif (AMR), algorithmes de recouvrement par patchs, multi-niveau, calcul parallèle, parallélisme informatique, schémas numériques, Després-Lagoutière, schéma d'advection à limitateur de flux aval, équations de Navier-Stokes, couplage hyperbolique-elliptique, cavité entraînée, tension de surface.

Résumé : Ce travail de thèse présente l'implémentation de la simulation d'écoulements diphasiques dans des conditions de réacteurs nucléaires à caloporteur eau, à l'échelle de bulles individuelles. Pour ce faire, nous étudions plusieurs modèles d'écoulements thermohydrauliques et nous focalisons sur une technique de capture d'interface mince entre phases liquide et vapeur. Nous passons ainsi en revue quelques techniques possibles de maillage adaptatif (AMR) et nous fournissons des outils algorithmiques et informatiques adaptés à l'AMR par patchs dont l'objectif est d'améliorer localement la précision dans des régions d'intérêt. Plus précisément, nous introduisons un algorithme de génération de patchs conçu dans l'optique du calcul parallèle équilibré. Cette approche nous permet de capturer finement des changements situés à l'interface, comme nous le montrons pour des cas tests d'advection ainsi que pour des modèles avec couplage hyperbolique-elliptique. Les calculs que nous présentons incluent également la simulation du système de Navier-Stokes incompressible qui modélise la déformation de l'interface entre deux fluides non-miscibles.

Title: Direct Numerical Simulation of Bubbles with Adaptive Mesh Refinement with Distributed Algorithms

Keywords: Nuclear Thermal-Hydraulics, two-phase flows, numerical simulation of bubbles, low-Mach conditions, Adaptive Mesh Refinement (AMR), patch covering algorithms, multilevel, parallel computing, multiprocessing, numerical schemes, Després-Lagoutière, limited downwind advection scheme, Navier-Stokes equations, hyperbolic-elliptic coupling, lid-driven cavity, surface tension.

Abstract: This PhD work presents the implementation of the simulation of two-phase flows in conditions of water-cooled nuclear reactors, at the scale of individual bubbles. To achieve that, we study several models for Thermal-Hydraulic flows and we focus on a technique for the capture of the thin interface between liquid and vapour phases. We thus review some possible techniques for Adaptive Mesh Refinement (AMR) and provide algorithmic and computational tools adapted to patch-based AMR, which aim is to locally improve the precision in regions of interest. More precisely, we introduce a patch-covering algorithm designed with balanced parallel computing in mind. This approach lets us finely capture changes located at the interface, as we show for advection test cases as well as for models with hyperbolic-elliptic coupling. The computations we present also include the simulation of the incompressible Navier-Stokes system, which models the shape changes of the interface between two non-miscible fluids.