Numerical and parallel modeling of anisotropic fitted mesh for industrial quenching applications

Sacha El Aouad

PSL
UNIVERSITÉ PARIS

THÈSE DE DOCTORAT
DE L'UNIVERSITÉ PSL

Préparée à MINES Paris

Numerical and parallel modeling of anisotropic fitted mesh for industrial quenching applications.

—

Modélisation numérique et parallèle d'un maillage ajusté anisotrope pour l'application de trempe industrielle.

Soutenue par

Sacha El Aouad

Le 09 décembre 2022

École doctorale n°364

Sciences Fondamentales et Appliquées

Spécialité

Mathématiques Numériques, Calcul Intensif et Données

Composition du jury :

Johan Hoffman KTH, Royal Institute of Technology, Suède	*Président*
Suzanne Michelle Shontz Université du Kansas	*Rapporteur*
Joan Baiges Université polytechnique de Catalogne	*Rapporteur*
Giulia Lissoni SC-Consultants	*Examinatrice*
Aurélien LARCHER Mines Paris	*Co-encadrant de thèse*
Elie HACHEM Mines Paris	*Directeur de thèse*

MINES PARIS | PSL

To my parents: Djenny & Georges...

Acknowledgements

I would like to thanks my supervisors, Prof. Elie Hachem and Dr. Aurélien Larcher for their invaluable advice, continuous support, and patience during my PhD study.

I am also grateful to every professor, staff member and colleague here at CEMEF, for their moral support, help and encouragement. A special thanks goes to the CFL team, and all my friends, for a cherished time spent together, everyone of you impacted and inspired me.

Lastly, my biggest thanks to my parents for all the support you have shown me, without you this would not have been possible. And, to my sister, you are and will remain my inspiration. Thank you for the constant motivation and laughs, I couldn't have done it without you.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Contents

Résumé en Français

Les procédés de trempe des métaux sont des traitements thermiques largement utilisés dans l'industrie. Le processus de refroidissement a un impact direct sur la modification des propriétés mécaniques, le contrôle de la microstructure et la libération des contraintes résiduelles. De nombreuses entreprises souhaitent aujourd'hui maîtriser ce procédé de refroidissement qui combine plusieurs paramètres. Leur objectif est de pouvoir les optimiser afin d'obtenir la combinaison assurant les propriétés métallurigique souhaitées telles que la dureté et la limite d'élasticité. La trempe étant un processus complexe comprenant plusieurs phénomènes physique à la fois dans le fluide et le solide, la simulation numérique s'avère être un outil important pour le contrôle et la modélisation de ce processus.
Pour le processus de trempe, plusieurs types de géométries de complexités différentes sont étudiés et analysés. La génération de maillage de ces configurations complexes est une tâche difficile. Le cadre mathématique général de cette thèse s'attaque à ces défis en améliorant les méthodes pour la multi-physique, en particulier les couplages fluide-thermique et fluide-solide. L'objectif principal de la thèse est de faire évoluer les méthodes immergées vers un maillage conforme aux interfaces. Il s'agit d'une nouvelle méthode qui combine les méthodes immergées avec des approches d'ajustement. L'objectif est de développer un maillage anisotrope adaptatif simple, rapide et robuste pour la CFD.

1.1 The Quenching Process

Metal quenching processes are widely used heat treatments in the industry, especially in automotive, nuclear and aerospace industries because the cooling process has a direct impact on changing mechanical properties, controlling microstructure, and releasing residual stresses.

Heat treatment is a common method of modifying the mechanical properties (such as hardness, toughness, and strength, etc.) of certain metals without substantially changing its chemical composition hence tailoring it to the needs of the environment and the demands of of the job in which the metal is being used. During the quenching process, the homogeneity of the microstructure needs to be preserved to avoid distortion and cracks, that's why controlling the cooling rate is very important.

To perform the quenching process, a heated part is usually immersed in a medium (a liquid like oil, water, ... or a gas such as nitrogen, helium, ...) to extract the heat contained therein. The heat transfer is performed through a fluid-solid interface. Figure 1.1 shows the different steps of the quenching process in the case of a liquid quenching medium: after immersion of the part in the medium known as quenchant, a vapor film surrounding the solid insulates the part from cooling because of high thermal gradient. This is known as calefaction where radiation and conduction through the vapor are dominant. When the surface temperature of the metal is lower than a critical temperature, the liquid comes into contact with the surface and the liquid boils from the surface; this is nucleate boiling. The heat transfer during this phase is the most efficient of the whole process and the maximum cooling rate is reached. At the end of the process, when the surface temperature of the metal is lower than the temperature of vaporization of the quenchant, the boiling ceases and the cooling is achieved by convection.

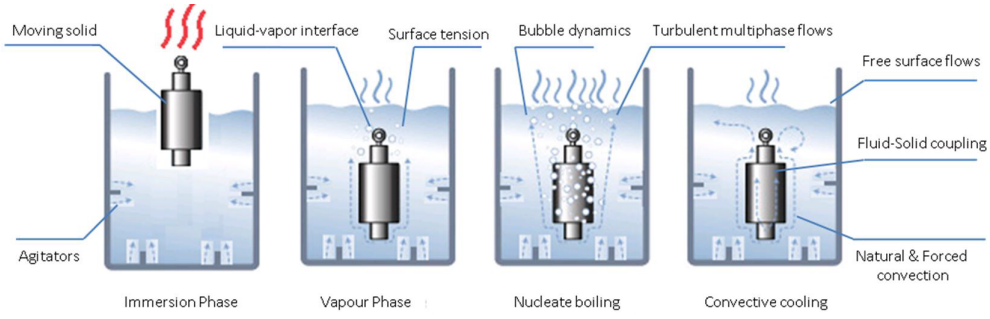


Figure 1.1: Schematic of the quenching process showing the stages and the complexity of the involved physics. Taken from: www.wizcol.com & INFINITY ANR Chair.

1.2 General Context and Motivation

Many industrial companies have a strong desire nowadays to control this cooling process and to take into account the optimal combinations of quench parameters with their complexity to achieve the desired metallurgical properties such as hardness and yield strength. This demand is accentuated by the strict deadlines for designing new materials and high-quality products.

Although numerical simulation is a common tool in the metallurgical industry for forming processes, currently no software is sufficiently predictable in the face of the complexity of boiling multi-phase flows during immersed quenching. A precise numerical modeling that provides a detailed understanding of the complex behavior of fluid flow and its impact on part cooling is therefore critical. Indeed, it allows for the first time to reduce the time and cost of developing new materials (by reducing time experimentation), allowing for the continuous development of safe and reliable products that meet customer specifications, and for the second time to improve the design of quenching systems, limiting production costs and decreasing energy consumption.

Despite the industrial interest in modeling precisely the quenching process, there is neither a global study nor a global answer addressing this problem in an industrial context. In order to predict precisely the liquid to vapor phase transition during boiling as well as to study the optimal combinations of quench parameters to reduce residual stresses in solid parts, an innovative coupled numerical framework needs to be designed and implemented.

Different numerical codes for the quenching process have been developed since 1980 [1]. The work by Garwood et al. [2] is one of the first attempts to characterize a quench tank using computational fluid dynamics. Almost 20 years later, an agitated quench tank was analyzed during the heat treatment of an aluminum cylinder [3]. Numerical flow simulations are increasingly used in quench tank design, but considerable imprecision still exists, especially because of the assumptions made on the use of simple geometries and approximate quench environments.

Currently, there is a strong demand to introduce more realistic physical behavior and to accurately predict liquid-to-vapor phases during boiling as well as a fluid-solid-thermal coupling in solid phase transformation which are both related to final metallurgical properties and are very important features. Hence, precise numerical modeling offering a detailed understanding of the complex behavior of fluid flow and its impact on cooling is very important. Such a model will allow the improvement and control of quenching systems and the cost and time reduction of developing new materials, and therefore more reliable products that meet the customers' specifications and requirements. That is why The INFINITY ANR Chair has been created and joins together several industrial partners to be able to understand, control, and optimize the quenching process: ARCELORMITTAL, AREVA, AUBERT &

DUVAL, CEFIVAL, CMI, FAURECIA, INDUSTEEL, LISI AEROSPACE, MONTUPET, SAFRAN, SCC, and TSV.

(a) Solid treatment & Fluid resolution

(b) Fluid-Solid coupling

Figure 1.2: Partitioned and monolithic approaches of the quenching process.

To model fluid-solid problems, the fluid and the solid can be treated separately as seen in Figure 1.2a, or using a monolithic approach treating the two sub-domains as one (Figure 1.2b). Using the immersed volume method, which will also be presented in this work, the objective of this project is to consolidate a unified multi-scale framework around understanding and simulating the quenching process and to integrate it into the finite element software QOBEO software by the scientific editor SCC. Modeling the liquid to vapor phase change, predicting different boiling modes with the transition between them and modeling the fluid-solid heat coupling with solid phase transformation are mainly aimed.

1.3 Objective of the thesis

This PhD thesis concerns the framework of the immersed volume methods and it is referred as PhD3 from the WP4 of the industrial chair INFINITY. The title of WP4 was “Enhancement of the immersed method for quenching environments”. Note also that these developments will serve in the in-house C++ library Cimlib-CFD, developed by the CFL research group, bringing multi-component applications in other areas of engineering: renewable energy, medicine, biology and more.

The accuracy of a simulation is closely connected to the design of the fluid-solid mesh, whereas the reliability of a simulation is related to the sensitivity of the numerical approximations such as the degree of discretization to model parameters

and data. For the quenching process, several types of geometries with different complexities are studied and analyzed. The mesh generation of such complex setups is a challenging task and consumes too much manpower. The general mathematical framework of this thesis will tackle such challenges by enhancing methods for multi-physics, in particular fluid-thermal and fluid-solid couplings.

A finite element numerical immersed framework, based on an implicit representation of different phases (liquid, gas, solid) and components using a level-set description, is combined with an error estimator for anisotropic meshing. The existing immersed volume method is intensively used in the context of multi-phase flows and for the fluid-structure interactions in the context of heat and mass transfer [4–7]. It also offers great flexibility to deal with different and realistic industrial immersed solids. However, a user parameter that defines the thickness at the interface remains dependent on the test case and is also very hard to generalize it.

Therefore, the main objective of the thesis is to evolve the immersed methods toward a sharp interface. This is a novel method that combines the immersed methods with fitting approaches. The goal is to develop a simple, fast and robust anisotropic adaptive body fitted meshes for CFD. These type of meshes allows, while keeping high accuracy, to deal with realistic quenching processes as well as the movement of solids inside quenching devices.

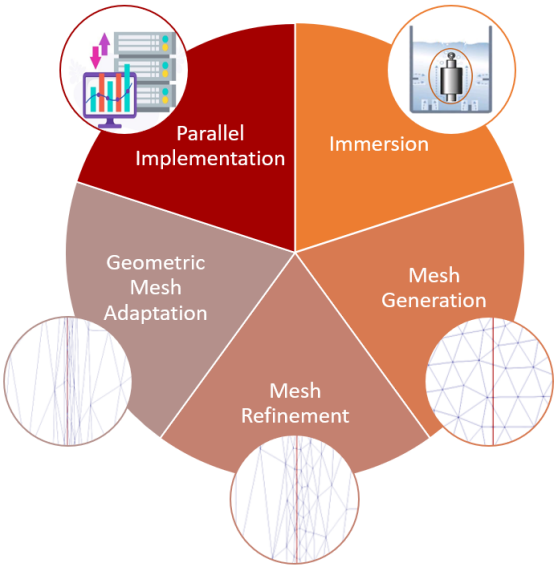


Figure 1.3: Main objectives of this thesis.

1.4 Layout of the thesis

This thesis is divided into six chapters. Chapter 1 introduced the general objective and motivation behind this work. Chapter 2 presents the Eulerian numerical framework and the partial differential equations to be solved. Chapter 3 is dedicated to the description of mesh generation methods as well as mesh adaptation techniques, especially the metric based anisotropic mesh adaptation. In chapter 4 the importance of capturing the fluid-solid interface of immersed geometry is highlighted, the existing methods to deal with this challenge are described, and the proposed approach creating an anisotropic fitted mesh is explained in detail. The parallel implementation and the extension of the proposed method in 3D are presented in chapter 5. In chapter 6, the validation of the framework and the proposed algorithm for several industrial applications is also detailed. And finally, the conclusion and possible extension and perspectives of the work are explored.

Bibliography

[1] D. S. Mackenzie, History of quenching, International Heat Treatment and Surface Engineering 2 (2) (2008) 68–73. 4

[2] D. Garwood, J. Lucas, R. Wallis, J. Ward, Modeling of the flow distribution in an oil quench tank, Journal of Materials Engineering and Performance 1 (6) (1992) 781–787. 4

[3] V. Srinivasan, K.-M. Moon, D. Greif, D. M. Wang, M.-h. Kim, Numerical simulation of immersion quenching process of an engine cylinder head, Applied Mathematical Modelling 34 (8) (2010) 2111–2128. 4

[4] E. Hachem, B. Rivaux, T. Kloczko, H. Digonnet, T. Coupez, Stabilized finite element method for incompressible flows with high reynolds number, Journal of Computational Physics 229 (23) (2010) 8643–8665. 6

[5] E. Hachem, S. Feghali, R. Codina, T. Coupez, Immersed stress method for fluid–structure interaction using anisotropic mesh adaptation, International Journal For Numerical Methods in Engineering 94 (9) (2013) 805–825.

[6] T. Coupez, E. Hachem, Solution of high-reynolds incompressible flow with stabilized finite element and adaptive anisotropic meshing, Computer Methods in Applied Mechanics and Engineering 267 (2013) 65–85.

[7] E. Hachem, S. Feghali, R. Codina, T. Coupez, Anisotropic adaptive meshing and monolithic variational multiscale method for fluid–structure interaction, Computers & Structures 122 (2013) 88–100. 6

Chapter 2

Numerical Framework

Contents

Résumé en Français

Les prédictions des phénomènes d'écoulement des fluides et de transfert de chaleur conjugué font l'objet d'études et de recherches intensives d'un point de vue numérique et industriel dans un grand nombre d'applications telles que la conception d'échangeurs de chaleur, l'étude du nucléaire, la conservation de l'énergie, les processus de chauffage et de trempe, etc. En fonction des études expérimentales d'un modèle simple, la prédiction des résultats d'applications plus complexes n'est pas toujours aussi facile. On a alors recours aux simulations numériques pour fournir une meilleure modélisation de l'application en termes de temps et de ressources, en particulier lorsqu'il s'agit de systèmes à plusieurs composants.
Ce chapitre se concentre sur la modélisation et la résolution d'applications d'écoulement de fluides ainsi que de problèmes de transfert de chaleur pour des problèmes multi-composants ou multi-phases. Après avoir passé en revue la manière de détecter et de saisir les différents composants impliqués, l'ensemble des équations utilisées et les méthodes de stabilisation sont brièvement décrites. Enfin, quelques applications numériques sont résolues pour évaluer la validité et la précision du cadre proposé.

2.1 Introduction

The predictions of fluid flow phenomena and conjugate heat transfer are subject to intensive study and research from a numerical and industrial point of view in a large number of applications such as heat exchanger design, nuclear study, energy conservation, heating and quenching processes, etc. Depending on experimental investigations of a simple model, the prediction of the results of more complex applications isn't always that easy. That is why numerical simulations are used to provide better modeling of the application in terms of time and resources, especially when dealing with multi-component systems.

This chapter focuses on modeling and solving fluid flow applications as well as heat transfer problems for multi-component or multi-phase problems. After reviewing how to detect and capture the different components involved, the set of equations used and stabilization methods are briefly described. And finally, some numerical applications are solved to assess the validity and accuracy of the proposed framework.

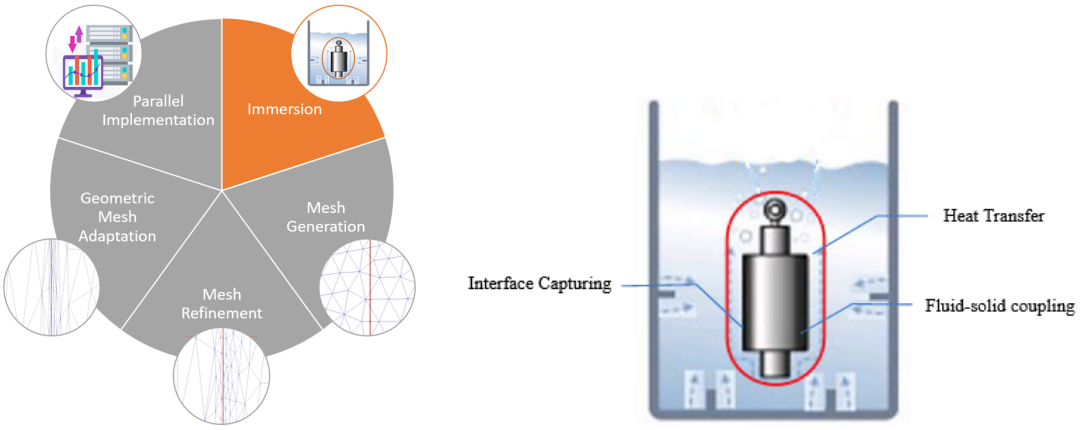


Figure 2.1: Schematic of the numerical framework.

2.2 The Level-set Method

The level-set method is an Eulerian method used to track interfaces. First proposed by S. Osher in 1988 [1], it was then further developed and applied to incompressible flows [2]. The method is very popular and is used in a large variety of applications, such as microstructural simulations in metallurgy, fast phase change in boiling, fluid-structure interaction, and multi-fluid flow. The level-set method allows us to

implicitly track the interface as the zero isovalue of a scalar function ϕ, the level-set function.

2.2.1 Standard Level-set Method

Let $\Omega \in \mathbb{R}^d$ be the computational domain with d the space dimension and Γ the interface separating two domains Ω_1 and Ω_2. The level-set function can be computed by considering a signed distance from the interface for each in point x in the domain:

$$\phi(x) = \begin{cases} dist(x,\Gamma) & \text{if} \quad x \in \Omega_1 \\ 0 & \text{if} \quad x \in \Gamma \\ -dist(x,\Gamma) & \text{if} \quad x \in \Omega_2 \end{cases} \tag{2.1}$$

To keep track of the evolution of the interface in time, the level-set ϕ must be updated accordingly. Therefore, its evolution function is represented by the transport equation:

$$\frac{d\phi}{d_t} = \frac{\partial\phi}{\partial_t} + u \cdot \nabla\phi = 0 \tag{2.2}$$

It's important to note that for any Eurlerian distance function f, the following Eikonal equation is achieved $||\nabla f|| = 1$.

When the level-set is convected by a certain velocity, the gradient of the solution $\nabla\phi$ has no theoretical limitation, thus it could tend to infinity, yielding numerical problems in the resolution. A re-initialization step is then needed so the level-set function remains an Eulerian distance function during the whole computation. In [3], the most classic re-initialization method is introduced using the following Hamilton-Jacobi equation:

$$\frac{\partial\phi}{\partial\tau} + S(\phi)(||\nabla\phi|| - 1) = 0 \tag{2.3}$$

with τ a virtual time step in which the re-initialization equation 2.3 is solved at each increment of the physical time domain and S the sign distance of ϕ. Hence ϕ will converge towards $||\nabla\phi|| = 1$, keeping the zero isovalue of the level-set unchanged. The zero isovalue is then implicitly computed. This property is important to keep since all physical properties at the interface will be computed and distributed in space according to the level-set function.

Since solving equation 2.3 is an iterative procedure, the computational cost can be restrictive for some large applications. A better numerical behaviour of the level-set method can be achieved by using a filtering level-set function [4–7]. In order to solve simultaneously the Hamilton-Jacobi equation 2.3, and transport the level-set function, the convected level-set method proposed in [7] is used.

2.2.2 Convected Level-set Method

The convected level-set method consists of filtering and truncating the level-set function using the following filter:

$$\tilde{\phi} = Etanh\left(\frac{\phi}{E}\right) \tag{2.4}$$

with E being a scalar parameter representing the thickness of truncation.

Since the filtered level-set function (equation 2.4) and its derivative $\tilde{\phi}'$ are bounded:

$$\lim_{\phi\to\pm\infty}\tilde{\phi} = \pm E \qquad and \qquad \lim_{\phi\to\pm\infty}\tilde{\phi}' = 0 \tag{2.5}$$

the Dirichlet boundary conditions can be imposed and the gradient of the level-set near the interface is close to 0.

As a signed distance function, the truncated level-set now verifies:

$$||\nabla\tilde{\phi}|| = 1 - \left(\frac{\tilde{\phi}}{E}\right)^2. \tag{2.6}$$

For the sake of simplicity, the tilde is dropped, and ϕ represents the truncated level-set. Its gradient can be linearized in function of the level-set at a previous time step by:

$$||\nabla\phi|| = \frac{\nabla\phi^-}{||\nabla\phi^-||}\nabla\phi. \tag{2.7}$$

The Hamilton-Jacobi equation is then merged into the convective form:

$$\frac{\partial\phi}{\partial\tau} + S(\phi)\frac{\nabla\phi^-}{\|\nabla\phi^-\|}\nabla\phi = S(\phi)\left(1 - \left(\frac{\phi}{E}\right)^2\right). \tag{2.8}$$

Using $U = s(\phi)\frac{\nabla\phi^-}{\|\nabla\phi^-\|}$ as the re-initialization velocity, the transport and the re-initialization equations can be combined to insure that the level-set remains a signed function distance as seen in [7]:

$$\frac{\partial\phi}{\partial t} + u\cdot\nabla\phi + \lambda S(\phi)\left(\|\nabla\phi\| - \left(1 - \left(\frac{\phi}{E}\right)^2\right)\right) = 0. \tag{2.9}$$

with $\lambda := \partial\tau/\,\partial t$.

The equation to solve now reads:

$$\frac{\partial\phi}{\partial t} + (u + \lambda U)\cdot\nabla\phi = \lambda S(\phi)\left(1 - \left(\frac{\phi}{E}\right)^2\right) \tag{2.10}$$

In [7–9], the proposed method is shown to reduce the computational cost and ensure better mass conservation than the classical level-set method.

2.3 Mixing laws

Working in a monolithic approach, a multi-component or multi-phase system is considered and solved as one singular domain. The different subdomains are detected and differentiated using level-set functions, and then the properties of each subdomain are assigned and distributed. To distribute the physical and material properties of the different subdomains such as the density ρ, the viscosity μ, and the conductivity k, a mixing law is used. The assignment of the properties is given through a smoothed Heaviside function applied over a narrow band ϵ, to account for the sharp variations along the different materials involved. This allows to achieve better continuity at the interface for the fluid-solid mixture and treats the different components as one composite domain:

$$H(x) = \begin{cases} 1 & \text{for } \phi(x) > \epsilon \\ \frac{1}{2}(1 + \frac{\phi(x)}{\epsilon} + \frac{1}{\pi}\sin(\frac{\pi\phi(x)}{\epsilon})) & \text{for } |\phi(x)| \leq \epsilon \\ 0 & \text{for } \phi(x) < -\epsilon \end{cases} \tag{2.11}$$

The different properties are then expressed using the following laws:

$$\rho = \rho_{fluid} H(\phi(x)) + \rho_{solid}(1 - H(\phi(x)) \tag{2.12}$$

$$\mu = \mu_{fluid} H(\phi(x)) + \mu_{solid}(1 - H(\phi(x)) \tag{2.13}$$

where ρ_{fluid}, μ_{fluid}, ρ_{solid} and μ_{solid} are the densities and dynamic viscosity of the fluid and solid, respectively.

For the thermal conductivity k, using a linear distribution will lead to abrupt changes along the interface [10]. A harmonic law is preferred to ensure the conservation of heat flux along the interface as follows:

$$k = \left(\frac{H(\phi(x))}{k_{fluid}} + \frac{1 - H(\phi(x))}{k_{solid}} \right)^{-1} \tag{2.14}$$

2.4 Numerical Resolution of the Navier-Stokes Equations

Let $\Omega \subset \mathbb{R}^d$, with d the space dimension, and $d\Omega$ its boundary. We consider the following velocity $\mathbf{u}$ - pressure p formulation of the Navier-Stokes equations for unsteady incompressible flows:

$$\begin{cases} \rho(\partial_t \mathbf{u} + \mathbf{u} \cdot \nabla \mathbf{u}) - \nabla \cdot \sigma = f \\ \nabla \cdot \mathbf{u} = 0 \end{cases} \tag{2.15}$$

where ρ is the density, f the body force vector per unity density and σ the stress tensor such as:

$$\sigma = 2\mu\epsilon(\mathbf{u}) - p\mathbf{I_d} \tag{2.16}$$

with μ the dynamic viscosity, $\mathbf{I_d}$ the identity tensor, and ϵ the strain-rate tensor defined as:

$$\epsilon(\mathbf{u}) = \frac{1}{2}(\nabla\mathbf{u} + \nabla\mathbf{u}^T) \tag{2.17}$$

2.4.1 Weak formulation

The weak formulation of (2.15)-(2.16) with velocity space $V \subset [H^1(\Omega)]^d$ and pressure space $Q = \{q \in L^2(\Omega) : \int_\Omega q = 0\}$ consists in finding $(u, p)\text{in}V \times Q$ such that:

$$\begin{cases} ((\rho(\partial_t\mathbf{u} + \mathbf{u}\cdot\nabla\mathbf{u})), \mathbf{w}) + (2\mu\epsilon(\mathbf{u}) : \epsilon(\mathbf{w})) - (p, \nabla\cdot\mathbf{w}) = (\mathbf{f}, \mathbf{w}), & \forall\mathbf{w} \in V \\ (\nabla\cdot\mathbf{u}, q) = 0, & \forall q \in \Omega \end{cases} \tag{2.18}$$

with (.,.) the $\mathscr{L}$ inner product over Ω.

To prevent spurious oscillations resulting from the convection-dominated regimes and solve the pressure instability problem, a variational multiscale method for the Navier– Stokes equations is used [11–13].

2.4.2 The Variational Multiscale Method

Let $\tau_\mathbf{h}$ be an admissible mesh constructed as a triangulation of Ω, and V_h and Q_h the finite dimensional spaces approximations of the function spaces V and Q, respectively. To ensure the stability of (2.18), the choice of V_h and Q_h must fulfill a compatibility condition [14].

In this work, $\mathbb{P}_1/\mathbb{P}_1$ elements with a Variational Multiscale method (VMS) [15] are used to ensure the stabilization. All unknowns are divided into coarse and fine components. The fine scales are solved in an approximate manner and modeled in function of the residual basis terms. Their effect is then transferred into the large

scale equations. The coarse and fine components of the velocity and pressure fields are:

$$\mathbf{u} = \mathbf{u}_h + \mathbf{u}' \tag{2.19}$$
$$p = p_h + p' \tag{2.20}$$

as well as their respective weight functions:

$$\mathbf{w} = \mathbf{w}_h + \mathbf{w}' \tag{2.21}$$
$$q = q_h + q' \tag{2.22}$$

The enriched function spaces are defined as $V = V_h \oplus V', V_0 = V_{h,0} \oplus V'$and $Q = Q_h \oplus Q'$. Therefore, the resulting finite element approximation of the time-dependent Navier–Stokes problem consists in finding $(u, p)\ inV \times Q$ such that:

$$\begin{cases} (\rho(\partial_t(\mathbf{u}_h + \mathbf{u}') + (\mathbf{u}_h + \mathbf{u}') \cdot \nabla(\mathbf{u}_h + \mathbf{u}')), (\mathbf{w}_h + \mathbf{w}')) \\ +(2\mu\epsilon(\mathbf{u}_h + \mathbf{u}') : \epsilon((\mathbf{w}_h + \mathbf{w}'))) - ((p_h + p'), \nabla \cdot (\mathbf{w}_h + \mathbf{w}')) \\ \qquad = (\mathbf{f}, (\mathbf{w}_h + \mathbf{w}')), & \forall \mathbf{w} \in V_0 \\ \\ (\nabla \cdot (\mathbf{u}_h + \mathbf{u}'), (q_h + q')) = 0, & \forall q \in Q \end{cases} \tag{2.23}$$

The stabilized formulation is then derived from equation (2.23) by forming fine and large scale problems. The fine-scale problem is defined on element interiors, and $\mathbf{u}'$ and p' are written in terms of the time-dependent large-scale variables using consistently derived residual-based terms.

Then, $\mathbf{u}'$ and p' are directly replaced into the large-scale problem, which gives rise to additional terms in the Finite Element formulation, tuned by a local stabilizing parameter. These terms are responsible for the enhanced stability compared to the standard Galerkin formulation. Finally, the coarse-scale equations can be computed:

$$\begin{cases} (\rho(\partial_t(\mathbf{u}_h + \mathbf{u}_h \cdot \nabla \mathbf{u}_h), \mathbf{w}_h) - (\tau_1 \mathcal{R}_m, \rho \mathbf{u}_h \cdot \nabla \mathbf{w}_h) + (2\mu\epsilon(\mathbf{u}_h) : \epsilon(\mathbf{w}_h)) \\ \qquad -(p_h, \nabla \cdot \mathbf{w}_h) - (\tau_2 \mathcal{R}_c, \nabla \cdot \mathbf{w}_h) = (\mathbf{f}, \mathbf{w}_h), & \forall \mathbf{w_h} \in V_{h0} \\ \\ (\nabla \cdot \mathbf{u}_h, q_h) - (\tau_1 \mathcal{R}_m, \nabla q_h) = 0, & \forall q_h \in Q_h \end{cases} \tag{2.24}$$

with $\mathcal{R}_m$ and $\mathcal{R}_c$ are piece-wise constant momentum and continuity residuals:

$$\mathcal{R}_m = \rho(\partial_t \mathbf{u}_h + \mathbf{u}_h \cdot \nabla \mathbf{u}_h) - \nabla p_h \tag{2.25}$$
$$\mathcal{R}_c = -\nabla \mathbf{u}_h \tag{2.26}$$

and τ_1 and τ_2 are piece-wise defined stabilization parameters adopted from [16]. More details about the formulation of VMS can be found in [13].

Space discretization being achieved using finite elements, the time discretization ($\partial_t \mathbf{u}_h$) can be achieved using Backward Difference Formula of order σ as BDF-σ, with $\sigma = 1$ to 4. To do so, a semi-implicit BDF scheme using Newton-Gregory backward polynomials can be used. The reader is referred to [17, 18] for more details on the BDF-σ temporal schemes.

2.5 Convection-Diffusion-Reaction Equation

In this section, the general equation of convection-diffusion-reaction (CDR) is described and solved. The solution of the CDR equation is an important feature for the numerical modeling of a wide range of fluid mechanics applications such as heat transfer equations, turbulence models, and the quenching process. The Standard Galerkin Finite Element method is typically used to solve these types of problems because of its simplicity. However, a standard Galerkin formulation causes spurious oscillations in convection-dominated regimes especially in high gradient regions. To overcome these numerical oscillations and gain in stability as well as accuracy, many methods have been proposed [19–21].

The following sections will focus on the description of the convection-diffusion-reaction equation, followed by an introduction of the standard Galerkin finite element method and finally the stabilization formulation chosen in order to overcome the numerical oscillations: the Streamline Upwind Petrov-Galerkin (SUPG) formulation.

2.5.1 Governing Equation

The CDR equation, over the bounded domain $\Omega \subset \mathbb{R}^d$ with boundary domain $d\Omega$, consists in finding a scalar $\alpha(x,t)$ such that:

$$\begin{cases} \partial_t \alpha + u \cdot \nabla \alpha - \nabla \cdot (k \nabla \alpha) + r\alpha = f & \text{in } \Omega \times [0,T] \\ \alpha(.,0) = \alpha_0 & \text{in } \omega \times [0,T] \\ \alpha = g & \text{on } d\Omega \times [0,T] \end{cases} \tag{2.27}$$

where k and r are respectively the positive diffusion factor and reaction coefficient, u a given divergence free velocity, f is a given source term, α_0 is the initial data and g is a given boundary condition. Note that the solution α can be the level-set, the temperature or any scalar. For this problem, only the Dirichlet Boundary condition $\alpha = 0$ on $d\Omega$ is considered.

2.5.2 The Standard Galerkin Finite Element Method

In order to solve equation 2.27 using the finite element method, its variational formulation needs to be derived. First, the function spaces are defined as follows:

$$H_0^1(\Omega) = \left\{ w \mid w \in H^1(\Omega), w = 0 \text{ on } d\Omega \right\} \tag{2.28}$$

with

$$H^1(\Omega) = \left\{ w \mid w \in \mathscr{L}^2(\Omega), \nabla w \in \mathscr{L}^2(\Omega) \right\}, \tag{2.29}$$

and $\mathscr{L}^2(\Omega)$ being the Hilbert vector space given by:

$$\mathscr{L}^2(\Omega) = \left\{ w(x) \mid \int_\Omega |w|^2 dx < \infty \right\} \tag{2.30}$$

In the following, the integral notation is represented as:

$$(a, b)_\Omega := \int_\Omega ab \, dx \tag{2.31}$$

The Galerkin variational formulation of the convection-diffusion-reaction equation is obtained by multiplying equation 2.27 by a test function $w \in H_0^1(\Omega)$ and integrating over the computational domain. The weak formulation is then:

$$\begin{cases} \text{Find } \alpha \in H_0^1(\Omega) \text{ such that:} \\ (\partial_t \alpha, w) + (k\nabla\alpha, \nabla w) + (u \cdot \nabla\alpha, w) + (r\alpha, w) = (f, w) \quad \forall w \in H_0^1(\Omega) \end{cases} \tag{2.32}$$

Considering the finite element partition $\mathscr{T}_h$ of Ω subdivided into simplex elements K, the functional spaces are now subdivided by discrete spaces: $H_{0,h}^1(\Omega)$ and $H_h^1(\Omega)$. The Galerkin finite element formulation then reads:

$$\begin{cases} \text{Find } \alpha_h \in H_{0,h}^1(\Omega) \text{ such that:} \\ (\partial_t \alpha_h, w_h) + \Big(k\nabla\alpha_h, \nabla w_h) + (u_h \cdot \nabla\alpha_h, w_h) + (r\alpha, w_h) = (f, w_h) \quad \forall w \in H_{0,h}^1(\Omega) \end{cases} \tag{2.33}$$

This formulation being unstable leads to spurious numerical oscillations especially when working in convection-dominant regimes, leading to the need for stabilization. In this work, the SUPG numerical scheme is used. This scheme commonly used nowadays to solve heat transfer finite element application, has proven to be efficient in reducing and eliminating the spurious oscillations related to the Galerkin formulation.

2.5.3 The Streamline Upwind Petrov-Galerkin Method

The SUPG method enhances the stability of the solution by adding a stabilizing term to the original Galerkin formulation. The stabilizing term is written within each element domain of the residual $\mathscr{R}_{u_h}$ of the standard equation to be solved by an operator applied on the test function. Artificial diffusion is added only in the direction of the flow by adding weighted residuals to the variational formulation of the problem. Additional weight $\tau_{\mathrm{K}} u \nabla w_h$ is added to the standard Galerkin test functions w_h in the upwind direction for all terms of the equation, hence adding more weight in the upstream direction and reducing the weight in the downstream direction. The modified test function becomes:

$$w_{h_{\text{modified}}} = w_h + \tau_{\mathrm{K}} u \nabla w_h. \tag{2.34}$$

Equation 2.33 then reads:

$$\begin{cases} \text{Find } \alpha_h \in H^1_{0,h}(\Omega) \text{ such that:} \\ (\partial_t \alpha_h + u \cdot \nabla \alpha_h, w_h) + \sum\limits_{\mathrm{K} \in \mathscr{T}_h} \tau_{\mathrm{K}} (\partial_t \alpha_h, u \cdot \nabla w_h)_{\mathrm{K}} + (k \nabla \alpha_h, \nabla w_h) + (r \alpha_h, w_h) \\ \quad + \sum\limits_{\mathrm{K} \in \mathscr{T}_h} \tau_{\mathrm{K}} (\mathscr{R}_{\alpha_h}, u \cdot \nabla w_h)_{\mathrm{K}} = (f, w_h) + \sum\limits_{\mathrm{K} \in \mathscr{T}_h} \tau_{\mathrm{K}} (f, u \cdot \nabla w_h)_{\mathrm{K}} \quad \forall w_h \in H^1_{0,h}(\Omega) \end{cases} \tag{2.35}$$

where τ_K is a stabilization parameter that tunes the amplitude of the added weight and $\mathscr{R}_{u_h}$ the appropriate residual of equation 2.33 given by:

$$\mathscr{R}_{\alpha_h} = -\nabla \cdot (k \nabla \alpha_h) + u \cdot \nabla \alpha_h + r \alpha_h \tag{2.36}$$

This stabilization adds numerical diffusion in the neighborhood of sharp gradient and boundary layers. In this work, the stabilizing parameter τ_K is chosen from [22] as:

$$\tau_{\mathrm{K}} = \frac{h_m}{2\|u\|_2} \left(\coth\left(\mathrm{P}e_{\mathrm{K}}\right) - \frac{1}{\mathrm{P}_{e_{\mathrm{K}}}} \right) \tag{2.37}$$

with $\mathrm{P}e_{\mathrm{K}}$ is the local Peclet number defined as :

$$\mathrm{P}e_{\mathrm{K}} = \frac{\|u\|_2 h_m}{2|\mathrm{K}|} \tag{2.38}$$

and h_m is the average mesh size.

2.6 Numerical Applications

To validate the proposed numerical framework, several test cases are presented in this section. We consider the open cavity problem to assess the ability of Navier-Stokes to deal with critical Reynolds number flow. Then the coupling of the flow solver and the mixing law is considered to solve a dam break flow hence a multi-phase problem.

2.6.1 Open Cavity Flow

We first consider a 2D open square cavity of side h, upon which lies a channel of height $0.5h$ as represented in Figure 2.2. A uniform velocity stream is prescribed at the inlet of the boundary located at the left side of the channel. Free-slip condition with zero tangential stress is applied on the upper boundary of the channel, and on the lower boundary for $x \in [1.2h, -0.4h] \cup [1.75h, 2.5h]$. The no slip-boundary conditions are imposed on the remainder of the lower boundary of the channel and the cavity walls.

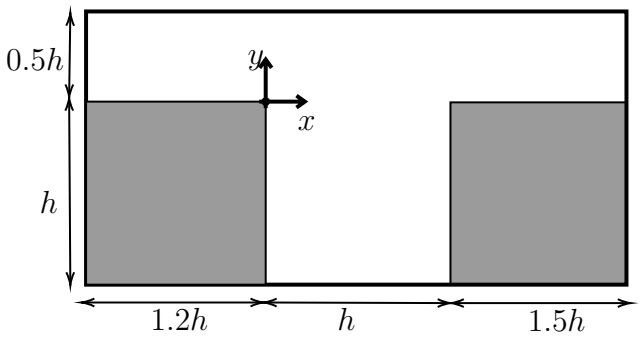


Figure 2.2: Setup of the open cavity problem.

To solve the Navier-Stokes equations, the VMS solver, presented in section 2.4.2, and the BDF time discretization have been used. The velocity evolution for $h = 1$ at a Reynolds number of $Re = 6000$ is shown in figure 2.5. The velocity field and boundary conditions lead to the formation of a recirculating eddy in the square cavity.

Plots 2.3 and 2.4 show the velocity V_x obtained on two different sensors positioned at a height $y = 0.05h$, and $x = 0.25h$ and $0.75h$. Well-defined oscillations emerge after around 30 time units and the solution slowly converges around 50 time units, with their amplitude being more important for the sensor positioned at $0.75h$ due to the recirculating eddy formed in the cavity.

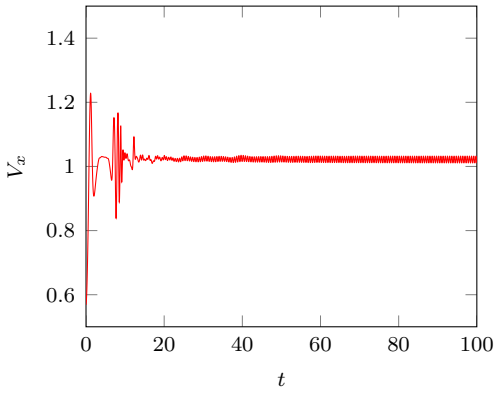


Figure 2.3: Velocity evolution at the sensor positioned at $0.25h$.

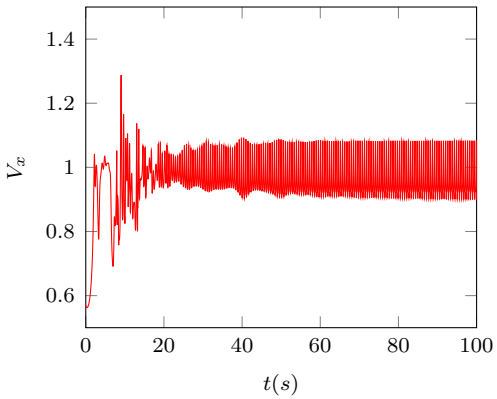


Figure 2.4: Velocity evolution at the sensor positioned at $0.75h$.

2.6.2 Dam break flow

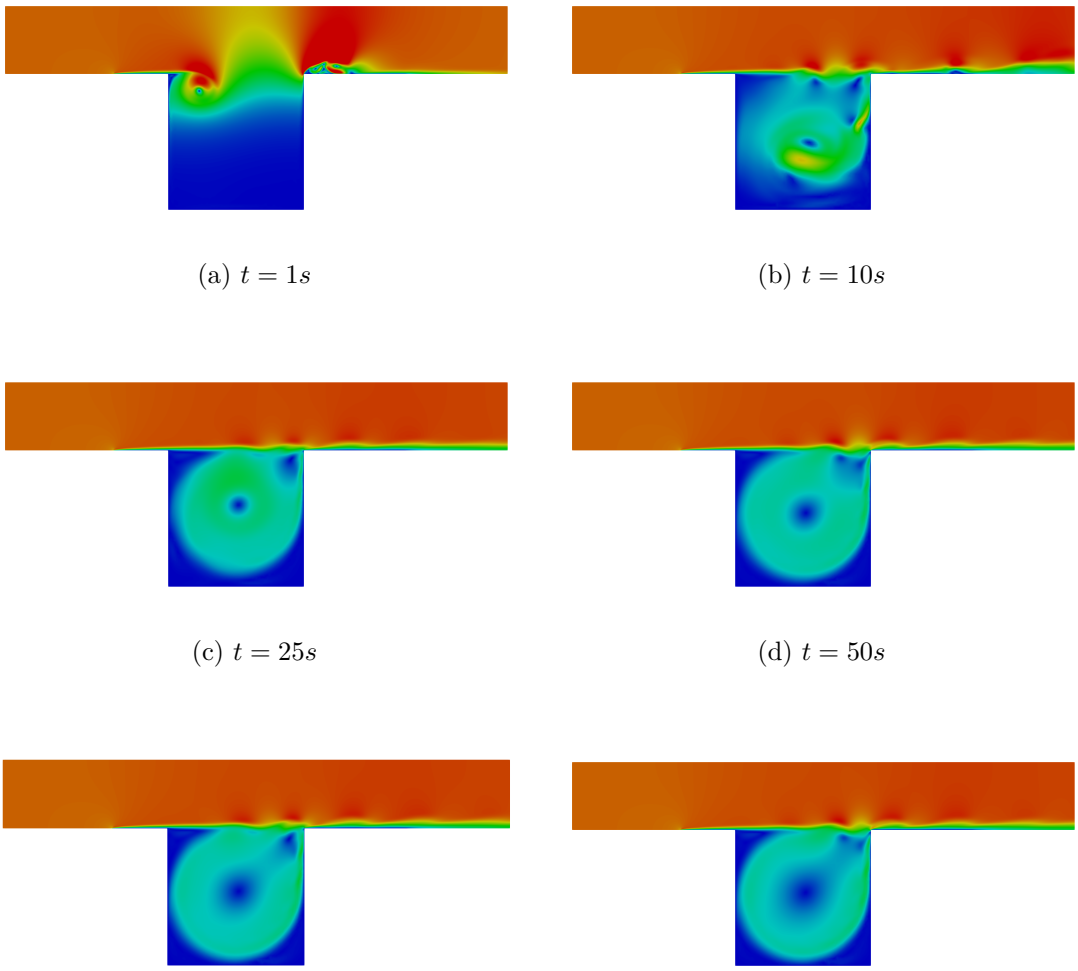
Next, a multi-phase problem is considered for the numerical validation of our framework. Its reliability is tested against the well-known dam-break case of Lobovský et al. [23]. The configuration of the problem is shown in figure 2.6: a water column with height $D = 0.15m$ and a depth of $W = 0.25m$ is confined on the left hand side of the enclosure of size $H = 0.3m$ and $L = 0.7m$, filled with air. No-slip boundary condition is enforced on all rigid walls. The physical properties of air and water considered are represented in table 2.1. The interface water-air is detected and captured using the level-set function described in section 2.2. The convected level-set allows us to follow the evolution and displacement of the water phase in the dam. Equations 2.12 and 2.13 are coupled with the Navier-stokes equations in order to simulate and solve the dam-break problem.

The solution obtained is coherent to the experimental results from [23] and the simulation results obtained, using the Volume of Fluid (VOF) model, in[24], in which the water column is on the right hand side (Figure 2.8).

Table 2.1: Numerical parameters considered for water and air, with ρ the fluid density, and μ the dynamic viscosity.

	$\rho(kg/m^3)$	$\mu(Pa.s)$
Water	1000	$855 \cdot 10^{-6}$
Air	1	$184 \cdot 10^{-7}$

Note that the characteristics and behavior of the flood wave resulting from the instantaneous break of a dam over dry can be divided into four distinct stages namely:



(a) $t = 1s$ (b) $t = 10s$

(c) $t = 25s$ (d) $t = 50s$

(e) $t = 75s$ (f) $t = 100s$

Figure 2.5: Velocity profile of the cavity flow at Re=6000.

1. collapsing and propagating
2. impacting and jetting
3. plunging and surging
4. rebounding and splashing

For more details see also Figure 2.7 and [23, 24]. This same hydrodynamic behavior can also be observed in the results obtained for our dam break simulation seen in Figure 2.8.

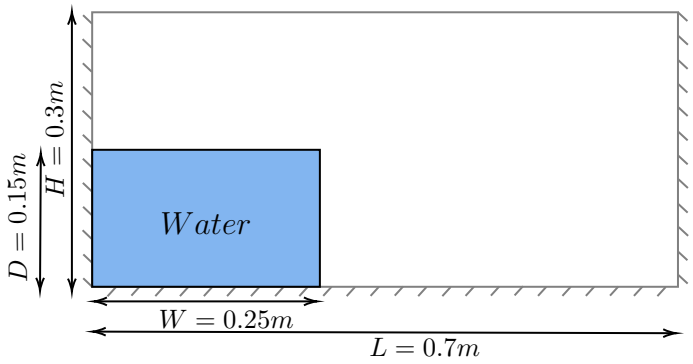


Figure 2.6: Schematic of the dam break setup.

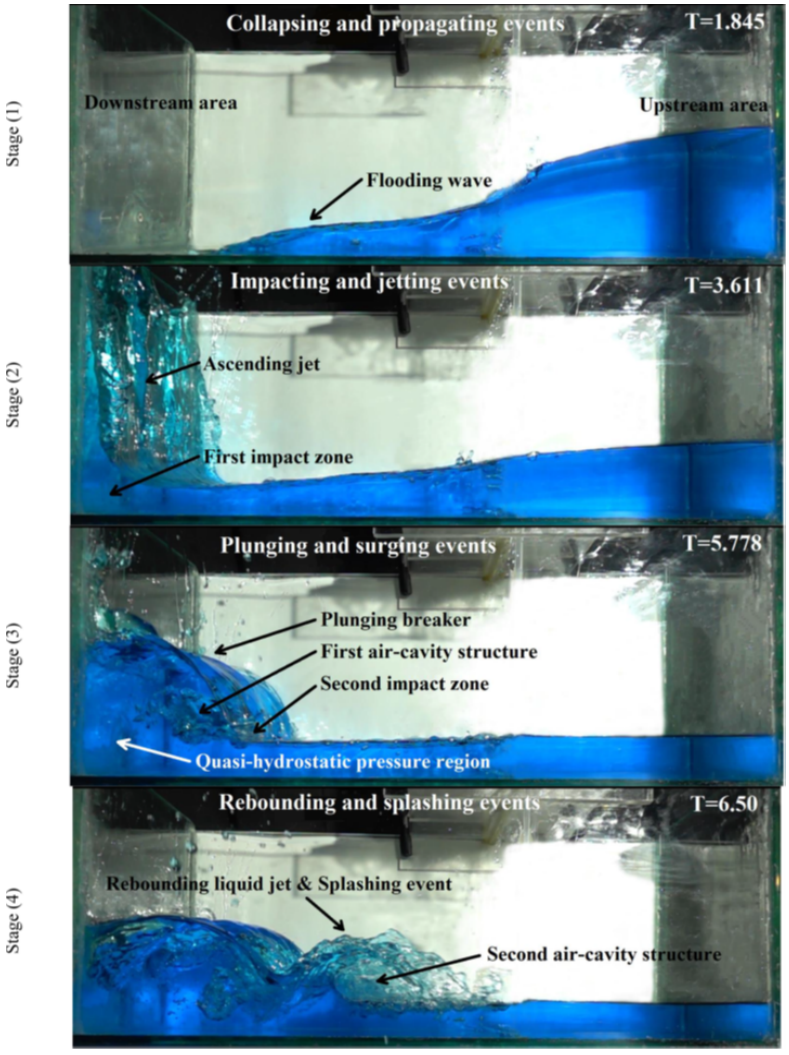


Figure 2.7: Characteristics of dam-break flow. Adopted from [24].

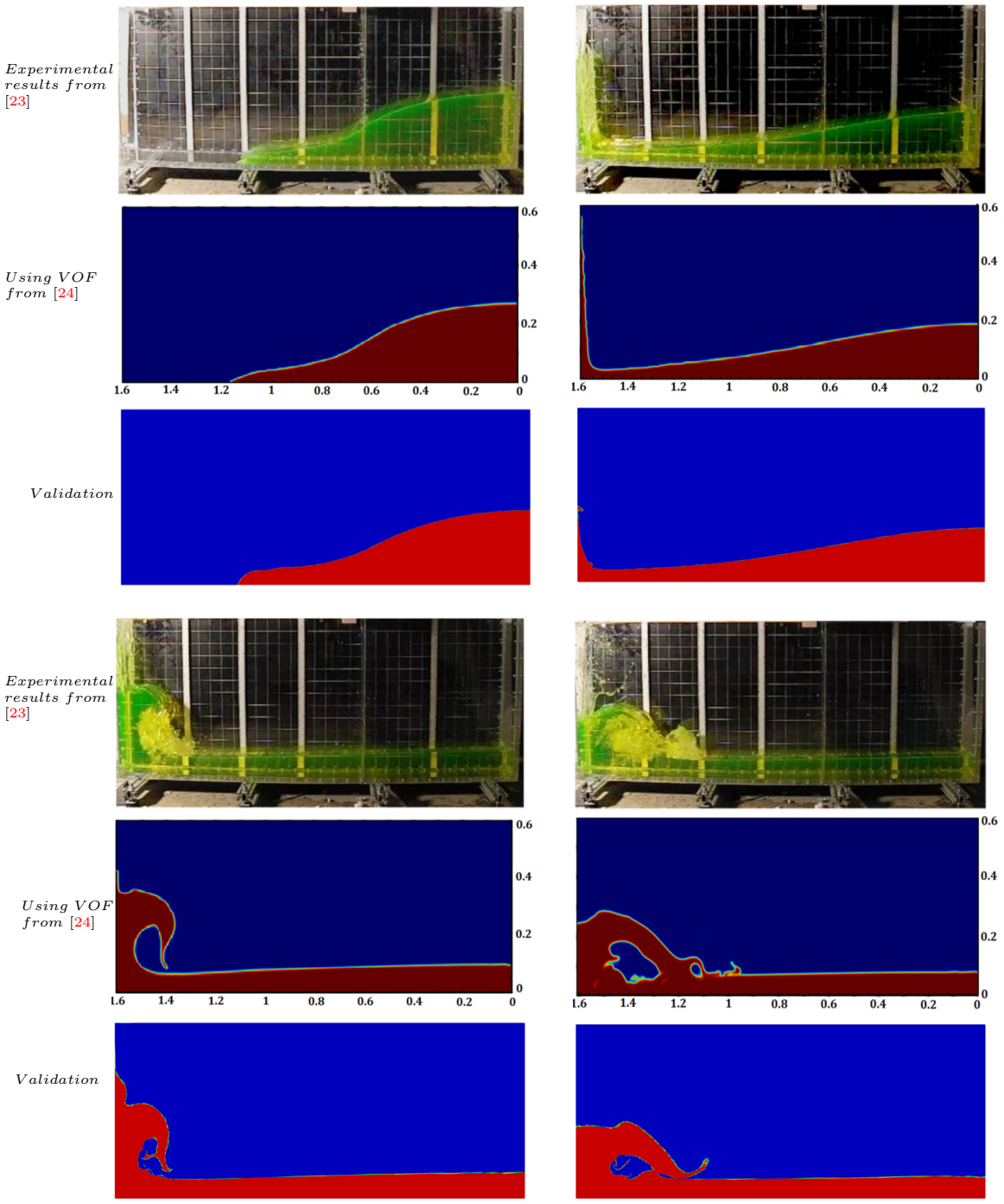


Figure 2.8: Comparison of the dam break flow evolution.

2.7 Conclusion

In this chapter, the numerical framework has been presented for fluid flow and heat transfer applications applied to multi-components or multi-phase problems. The convected level-set method detects the interface of an immersed part or different phases. The Navier-Stokes equations are solved using the Variational Multiscale method to ensure stabilization and the resolution of Convection-Diffusion-Reaction equations has also been presented. Numerical test cases have been presented and solved to assess the validity of the proposed solvers and framework.

However, in order to solve dynamic simulations and complex problems, the physical domain must be discretized to allow solving of partial differential equations. In the next chapter, mesh generation, i.e. the discretization, and mesh adaptation are presented and described in detail.

Bibliography

[1] S. Osher, J. A. Sethian, Fronts propagating with curvature-dependent speed: Algorithms based on hamilton-jacobi formulations, Journal of Computational Physics 79 (1) (1988) 12–49. 11

[2] Y.-C. Chang, T. Hou, B. Merriman, S. Osher, A level set formulation of eulerian interface capturing methods for incompressible fluid flows, Journal of Computational Physics 124 (2) (1996) 449–464. 11

[3] M. Sussman, P. Smereka, S. Osher, A level set approach for computing solutions to incompressible two-phase flow, Journal of Computational Physics 114 (1) (1994) 146–159. 12

[4] F. Gibou, R. Fedkiw, R. Caflisch, S. Osher, A level set approach for the numerical simulation of dendritic growth, Journal of Scientific Computing 19 (1) (2003) 183–199. 12

[5] O. Desjardins, V. Moureau, H. Pitsch, An accurate conservative level set/ghost fluid method for simulating turbulent atomization, Journal of Computational Physics 227 (18) (2008) 8395–8416.

[6] M. Sanchez, O. Fryazinov, P.-A. Fayolle, A. Pasko, Convolution filtering of continuous signed distance fields for polygonal meshes, in: Computer Graphics Forum, Vol. 34, Wiley Online Library, 2015, pp. 277–288.

[7] C. Bahbah, M. Khalloufi, A. Larcher, Y. Mesri, T. Coupez, R. Valette, E. Hachem, Conservative and adaptive level-set method for the simulation of two-fluid flows, Computers & Fluids 191 (2019) 104223. 12, 13

[8] A. Bonito, J.-L. Guermond, S. Lee, Numerical simulations of bouncing jets, International Journal for Numerical Methods in Fluids 80 (1) (2016) 53–75.

[9] T. Coupez, L. Silva, E. Hachem, Implicit boundary and adaptive anisotropic meshing, in: New challenges in grid generation and adaptivity for scientific computing, Springer, 2015, pp. 1–18. 13

[10] S. Patankar, Numerical heat transfer and fluid flow: Crc press (1980). 14

[11] M. Cervera, M. Chiumenti, R. Codina, Mixed stabilized finite element methods in nonlinear solid mechanics: Part i: Formulation, Computer Methods in Applied Mechanics and Engineering 199 (37-40) (2010) 2559–2570. 15

[12] F. Brezzi, L. Franca, T. Hughes, A. Russo, b = $\propto$ g, Computer Methods in Applied Mechanics and Engineering 145 (3) (1997) 329–339.

[13] E. Hachem, B. Rivaux, T. Kloczko, H. Digonnet, T. Coupez, Stabilized finite element method for incompressible flows with high reynolds number, Journal of Computational Physics 229 (23) (2010) 8643–8665. 15, 17

[14] A. Ern, J.-L. Guermond, Theory and practice of finite elements, Vol. 159, Springer Science & Business Media, 2013. 15

[15] T. J. Hughes, Multiscale phenomena: Green's functions, the dirichlet-to-neumann formulation, subgrid scale models, bubbles and the origins of stabilized methods, Computer Methods in Applied Mechanics and Engineering 127 (1-4) (1995) 387–401. 15

[16] R. Codina, Stabilization of incompressibility and convection through orthogonal sub-scales in finite element methods, Computer Methods in Applied Mechanics and Engineering 190 (13-14) (2000) 1579–1599. 17

[17] D. Forti, L. Dedè, Semi-implicit bdf time discretization of the navier–stokes equations with vms-les modeling in a high performance computing framework, Computers & Fluids 117 (2015) 168–182. 17

[18] P. Meliga, E. Hachem, Time-accurate calculation and bifurcation analysis of the incompressible flow over a square cavity using variational multiscale modeling, Journal of Computational Physics 376 (2019) 952–972. 17

[19] R. Codina, Comparison of some finite element methods for solving the diffusion-convection-reaction equation, Computer Methods in Applied Mechanics and Engineering 156 (1-4) (1998) 185–210. 17

[20] S. Badia, R. Codina, Analysis of a stabilized finite element approximation of the transient convection-diffusion equation using an ale framework, SIAM Journal on Numerical Analysis 44 (5) (2006) 2159–2197.

[21] R. Codina, J. M. González-Ondina, G. Díaz-Hernández, J. Principe, Finite element approximation of the modified boussinesq equations using a stabilized formulation, International Journal for Numerical Methods in Fluids 57 (9) (2008) 1249–1268. 17

[22] T. J. Hughes, G. R. Feijóo, L. Mazzei, J.-B. Quincy, The variational multiscale method—a paradigm for computational mechanics, Computer Methods in Applied Mechanics and Engineering 166 (1-2) (1998) 3–24. 19

[23] L. Lobovskỳ, E. Botia-Vera, F. Castellana, J. Mas-Soler, A. Souto-Iglesias, Experimental investigation of dynamic pressure loads during dam break, Journal of Fluids and Structures 48 (2014) 407–434. 21, 22, 24

[24] F. Garoosi, A. N. Mellado-Cusicahua, M. Shademani, A. Shakibaeinia, Experimental and numerical investigations of dam break flow over dry and wet beds, International Journal of Mechanical Sciences 215 (2022) 106946. v, 21, 22, 23, 24

Chapter 3

Anisotropic Mesh Adaptation

Contents

Résumé en Français

Avec le besoin continu de comprendre les phénomènes physiques et grâce à l'augmentation de la puissance de calcul, la modélisation numérique est devenue un outil très important. Cependant, la modélisation de ces phénomènes à l'aide de la simulation de la dynamique des fluides et de l'analyse thermique peut devenir très complexe. Ainsi, afin de concevoir des configurations réalistes et d'augmenter la précision de la solution et l'efficacité du calcul, des techniques d'adaptation du maillage ont été développées.
En effet, dans le chapitre précédent, la résolution des équations de Navier-Stokes et de Convection-Diffusion-Réaction ont été présentées et définies. Ces équations sont en fonction du temps et de l'espace et ne peuvent pas être résolues analytiquement sur un domaine entier. Par conséquent, le domaine physique doit être divisé en un nombre fini d'éléments géométriques: c'est le maillage ou la discrétisation de l'espace. Cette discrétisation va permettre de résoudre les équations aux dérivées partielles.
Avec un maillage uniforme simple et des degrés de liberté limités, la capture de caractéristiques physiques avec une échelle fine peut s'avérer assez coûteuse et peut même générer des résultats incorrects, en particulier lorsqu'il s'agit d'un problème transitoire comportant des caractéristiques à plusieurs échelles. Une résolution plus élevée de la solution est alors nécessaire : c'est pourquoi des techniques de raffinement de maillage local ont été développées. L'adaptation du maillage est donc considérée comme un ingrédient important du calcul numérique.
Dans ce chapitre, un aperçu des techniques de génération de maillage est présenté. Ensuite, l'adaptation de maillage anisotrope utilisée dans ce travail est décrite en détail. Enfin, le calcul de la fonction level-set comme une fonction de distance signée sur chaque sommet du maillage est expliqué.

3.1 Introduction

With the continuous need to understand physical phenomena and the increase in computational power, numerical modeling has become a very important tool. However, the modeling of such phenomena using fluid dynamic simulation and thermal analysis can become very complex. So, in order to devise realistic configurations and increase the solution's accuracy and computational efficiency, mesh adaptation techniques have been developed.

Indeed, in the previous chapter, the resolution of Navier-Stokes and Convection-Diffusion-Reaction equations were presented and defined. These equations are in function of time and space and can't be solved analytically over an entire domain. Therefore, the physical domain is to be divided into a finite number of geometrical elements: this is mesh generation or space discretization. This discretization will allow solving the partial differential equations.

With a simple uniform mesh and limited degrees of freedom, capturing fine scale physical features can be quite expensive and can even generate incorrect results, especially when dealing with a transient problem having multi-scale features. A higher resolution of the solution is then needed: that is why local mesh refinement techniques have been developed. Hence, mesh adaptation is regarded as an important ingredient in numerical computation.

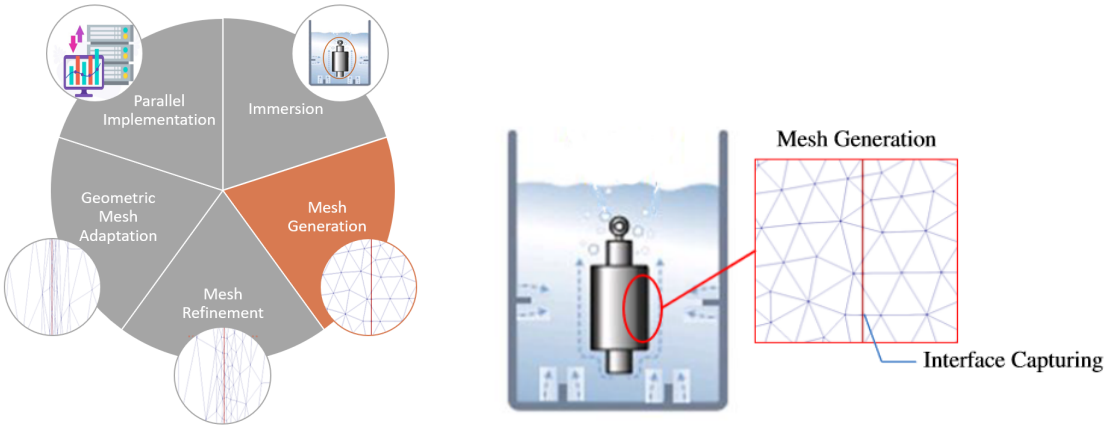
In this chapter, an overview of the mesh generation techniques is presented. Then the anisotropic mesh adaptation used in this work is described in detail. And finally, the computation of the level-set function as a signed distance function on each vertex of the mesh is explained.

3.2 Mesh Generation

Finite element methods are effective techniques to compute approximate solutions of partial differential equations. These methods use a discrete domain instead of the continuous one to approximate the solution on its vertices. Mesh generation is a key step in the numerical solution of physical problems using the finite element method and can be challenging and time-consuming. Hence, several methods and algorithms have been developed to generate automatically an appropriate mesh such as:

1. the advancing front method
2. the Delaunay method
3. the tree-based method
4. the topological method.

These methods will be briefly described in the following sections, focusing on the last one, the topological method, that is developed by the CFL (Computing and Fluids) team at CEMEF and used in this work.



(a) (b) Isotropic mesh generation & interface detection.

Figure 3.1: Mesh generation framework.

3.2.1 Advancing Front Method

The advancing front method [1, 2] constructs the mesh element by element starting from a defined boundary. In the beginning, the front consists of the boundary edges and it progresses forward as each new element is created. No area outside of the defined boundary is included in the triangulation process. The creation of elements starts at the boundary and progresses inward. After defining edges on the domain's boundary, for each edge, a new point is created and connected to the edge such as the new point closest to the edge. Once the element is created, the front is moved forward and the process is repeated iteratively until all the area enclosed by the initial boundary defined is meshed. Figure 3.2 illustrates the advancing front method. At the start, the front, represented in red, is the entire exterior boundary divided into several edges. Starting at a first edge (v_7, v_1), the closest point to it is point i. Connecting the three points, a new element, represented in blue, is created and the front progresses. On the updated front, moving counterclockwise, another edge is treated by repeating the same steps until all the bounded domain is meshed. However, during the creation process the quality of elements might deteriorate as the front advances causing convergence problems. The correction of a degenerate element is done as a second step where vertices should be removed and the process restarted. Hence, guaranteeing the elements' quality and the convergence of the method can be troublesome, particularly when dealing with complex three-dimensional geometries and domains.

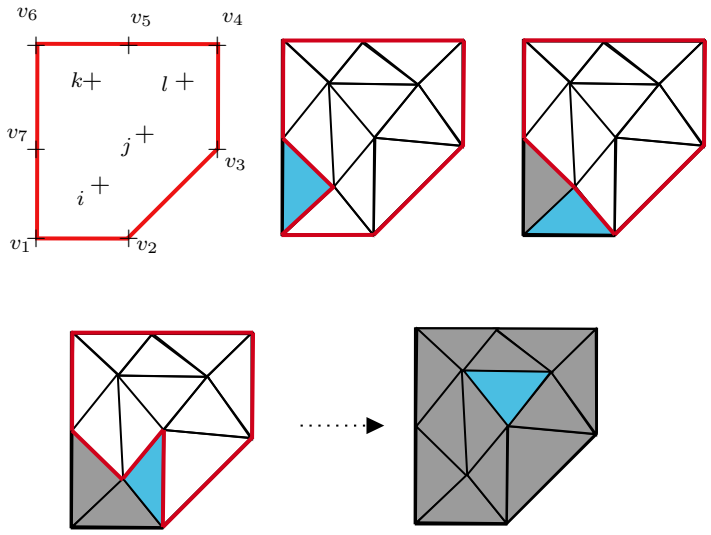


Figure 3.2: Representation of the advancing front method

3.2.2 Delaunay Method

The Delaunay method [3] is a classical mesh generation method that has been used by many researchers [4, 5] trying to deal with the properties of this method. The general concept is that the triangulation must satisfy the "empty sphere" criterion, known as the Delaunay criterion. This criterion states that the open ball (i.e. a disk for two-dimensional problems and a sphere for three-dimensional geometries) circumscribing each element shouldn't contain any vertex from the mesh. The Delaunay triangulation consists in first triangulating a box that includes all the vertices, then introducing the vertices one after the other in the triangulation by the Delaunay kernel. Finally, the elements outside the domain are removed. The obtained triangulation can be refined by inserting new points inside the domain. The elements whose circumscribed circles contain these vertices will be removed and new elements will be created respecting the above-stated criterion. The main disadvantage of the Delaunay method is related to the fact that the Delaunay criterion doesn't take into account the quality of the element created. The method may thus generate elements of very poor quality, especially near the borders. An optimization phase is therefore essential to the generation of a mesh. The Delaunay triangulation method is very technical and requires great expertise for the implementation of a robust and fast version. More information and details on Delaunay algorithms might be found in [3, 5–7].

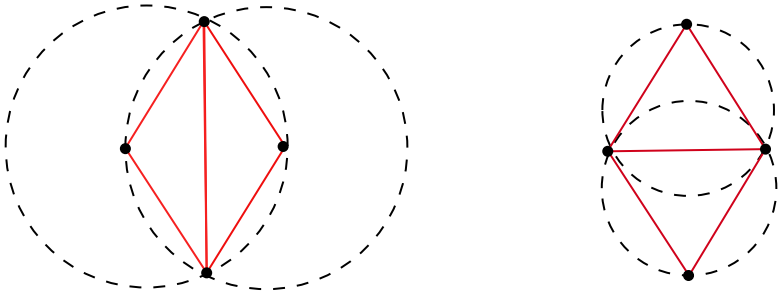


Figure 3.3: A Delaunay triangulation (right) vs a non-Delaunay triangulation (left) for a set of four points

3.2.3 Tree-based Method

The main idea of this method is to create an envelope that covers the targeted domain using a tree-like structure (quadtree for two-dimensional problems or octree for three-dimensional problems) and then use cells or bins of the tree to derive elements of the mesh. The method consists in defining a parent cell encompassing the object to mesh, and refining the latter recursively. In practice, the parent cell (or root) is divided into four identical cells (or eight in 3D), and each cell is marked according to its position relative to the object to be meshed: inside, outside, or if it intersects the boundary of the object. Only cells cut by the boundary are subdivided again. The process is repeated until the desired level of resolution is reached [8–10]. Using the tree-based method, a solution is always built, and the problem of existence doesn't arise. The boundary of the result is built as the mesh generation progresses. The tree-based method can be used to build simplices or quadrilateral elements. However, this method might generate poor quality elements, especially near the boundary.

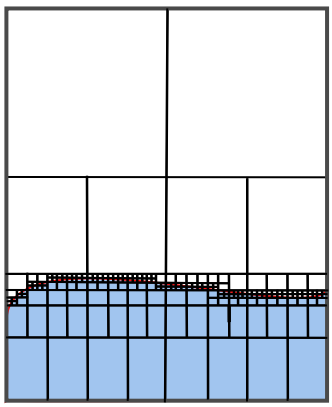


Figure 3.4: Tree based method using quadrilateral elements.

3.2.4 Topological Method

The topological method has been introduced in our laboratory CEMEF by T. Coupez [11]. The topological mesher MTC proceeds by iterative local optimization of the mesh topology, starting from an initial random mesh of the domain. This method presents many advantages such as its robustness and its easy implementation. It's based on two aspects: the minimal volume criterion and geometrical mesh quality. For more information about the method, the reader is referred to [11, 12]. First, some useful definitions will be recalled.

Let $\mathcal{N} \subset \mathbb{N}$ a set of finite number of vertices. Let $\mathcal{P}_{\mathcal{D}}(\mathcal{N})$ the set of parts in $\mathcal{N}$ made up of D different vertices and forms the set of all possible elements that can constitute $\mathcal{N}$.

$\mathcal{T}$ is a mesh topology meaning it is made of the set of elements verifying the properties defined in Definition 2, and the subset of vertices forming its elements is:

$$\mathcal{N}(\mathcal{T}) = \bigcup_{K \in \mathcal{T}} K \tag{3.1}$$

Definition 1. *A d-simplex is the convex hull of its (d + 1) affinely independent points, called vertices.*
For instance, a triangle is a 2-simplex and a tetrahedron is a 3-simplex.

Definition 2. *Consider a domain $\Omega \in \mathbb{R}^d$. Denote by $\mathcal{N}$ a finite set of vertices in Ω and $\mathcal{T}$ a set of d-simplices generated from the $\mathcal{N}$ vertices. Let $\mathcal{F}(\mathcal{T})$ be the set of $\mathcal{T}$'s elements' faces.*
We say that $\mathcal{T}$ is a ***mesh topology*** *of Ω if and only if:*

(i) each face F of $\mathcal{F}(\mathcal{T})$ shares at least one and at most two elements of $\mathcal{T}$.

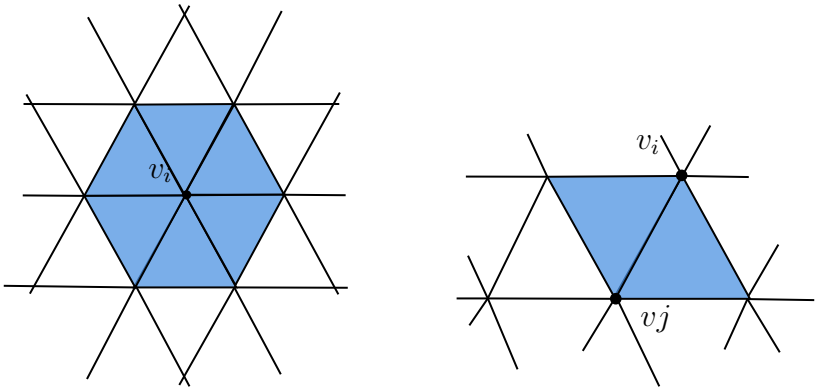
(ii) $(\mathcal{N}, \partial\mathcal{T})$ is a mesh of the boundary $\partial\Omega$.

Hence, the mesh topology consists of the mesh connectivity: the element-vertex relation. It can be described independently from any mesh vertex coordinates.

The topological mesh generator solves an optimization problem under the constraint of the volume of Ω. A local mesh modification of a mesh topology is done via a cut/paste operation where a subset of elements to be deleted $\mathcal{A}$ is replaced by another subset $\mathcal{B}$:

$$\mathcal{T} \leftarrow \mathcal{T} - \mathcal{A} + \mathcal{B} \tag{3.2}$$

Note that, $\mathcal{A}$ and $\mathcal{B}$ have the same boundary $\partial\mathcal{A} = \partial\mathcal{B}$ only if the result is still a mesh topology. The mesh topology is optimized in the neighborhood of a vertex or an edge (Figure 3.5).



(a) $\mathcal{A} = \mathcal{T}(v_i)$ (b) $\mathcal{A} = \mathcal{T}(v_i, v_j)$

Figure 3.5: Element subsets.

The generation of new mesh topologies, preserving the boundary, is achieved using the "starring" operator: a vertex $\mathcal{S}$ is connected to the given boundary faces that don't contain it:

$$\mathcal{T}^*(\mathcal{S}, \partial\mathcal{A}) = \{K | K = \{\{\mathcal{S}\} \cup F\}, F \in \partial\mathcal{A}, \mathcal{S} \notin F.\} \tag{3.3}$$

As already mentioned, the new subset created should be a mesh topology. That's why the closure $\bar{\mathcal{A}}$ is introduced:

$$\bar{\mathcal{A}} = \{K \in \mathcal{T}, K \subset \mathcal{N}(\mathcal{A})\} \tag{3.4}$$

And finally, the cut/paste operation becomes [12]:

$$\mathcal{T} \leftarrow \mathcal{T} - \bar{\mathcal{A}} + \mathcal{B} \tag{3.5}$$

3.2.4.1 Selection criteria for optimal local mesh generation

The generation of the new local topologies is based on:

- **The minimum volume principle:**
  This principle ensures the conformity of the elements, with no element overlaps. Let $\mathcal{A} \subset \mathcal{T}$ be a mesh topology. According to the minimum volume principle, the optimal re-triangulation, generated by the local mesh modification that is a **starring** operation improving the quality of $\mathcal{A}$, is given by:

$$\begin{gathered} \mathcal{B} = \arg\min_{\mathscr{T}} \sum_{K \in \mathscr{T}} |K| \\ with \quad \mathscr{T} = \{\mathscr{T}^*(S, \partial\mathcal{A}), s.t \ S \in \mathcal{N}(\mathcal{A}) \cup \{\mathcal{C}(\mathcal{A})\}\} \end{gathered} \tag{3.6}$$

where $|K|$ denotes the volume of the element K and $\mathcal{C}(\mathcal{A})$ the centroid of the vertices on $\partial\mathcal{A}$. The minimization is done over the set of all possible mesh topologies, obtained for all the vertices in $\mathcal{A}$, connecting the vertices on the border of $\mathcal{A}$, or other vertices like $\mathcal{C}(\mathcal{A})$, with all boundary faces.

When there are several options for a certain subset, the selection is made following mesh quality. The shape factor of the simplex is defined as the ratio between the volume of the element K and the length of its edges h_K.

- **The geometric quality criterion:**
  The quality of elements in a mesh topology is an important requirement for the assessment of finite element meshes generated automatically or meshes that have undergone adaptation. A minimal requirement for mesh quality is for all elements to be non-inverted, and that no element has a volume equal to zero. It is also important to avoid nearly degenerate elements such as a *needle* or *flattened* elements that have one angle close to zero or π, respectively.

Some quality measures

In the literature, several element quality measures have been proposed based on different geometric parameters [13], some of which are listed below.

Considering a triangular element K (Figure 3.6) with an area $\mathcal{A}$, a half parameter p, and edges lengths: $a = AB, b = AC, c = BC$. Let r and R be the inscribed and circumscribed radii, respectively, and α, β and γ the angles at vertex A, B and C, respectively. L_{max} (resp. L_{min}) is the largest (resp. smallest) length edge.

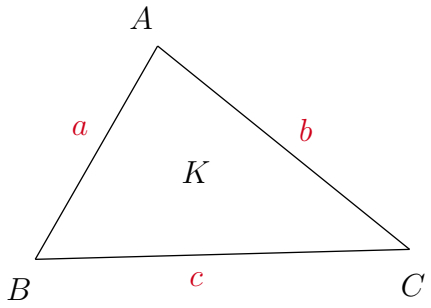


Figure 3.6: A triangular element K

Let's recall some elementary geometry results:

The area $\mathcal{A}$ is one half the magnitude of the cross product of any pair of adjacent edge vector or it can be computed as:

$$\mathcal{A} = rp \tag{3.7}$$

or using the well-known Heron's formula:

$$\mathcal{A} = \sqrt{p(p-a)(p-b)(p-c)} \tag{3.8}$$

▶ Aspect Ratio:
The quality of an element $Q(K)$ can be defined by computing its aspect ratio AR and compared to the reference element $\hat{K}_0$. When using the aspect ratio as a criterion, $\hat{K}_0$ is chosen as the equilateral unit triangle with $AR = 1$. The aspect ratio is then computed as:

$$Q(K) = AR = \frac{L_{max}}{2\sqrt{3}r}. \tag{3.9}$$

▶ Aspect Frobenius:
The aspect Frobenius is the sum of the edge lengths squared divided by the area and normalized so that a unit equilateral triangle has a value of 1.

$$Q(K) = \frac{a^2 + b^2 + c^2}{4\mathcal{A}\sqrt{3}}. \tag{3.10}$$

▶ Edge Ratio:
The edge ratio of a triangle is:

$$Q(K) = \frac{L_{max}}{L_{min}} \tag{3.11}$$

▶ Radius Ratio:
The radius ratio is expressed as:

$$Q(K) = \frac{R}{2r} \tag{3.12}$$

The reference quality for an equilateral unit triangle $\hat{K}_0$ is equal to 1.

▶ Maximum and Minimum Angles:
The maximum included angle of the triangle, measured in degrees, is

$$Q(K) = \max_{n\in\{0,1,2\}} \left\{ \arccos\left(\frac{\vec{L}_n \cdot \vec{L}_{n+1}}{\left\|\vec{L}_n\right\| \left\|\vec{L}_{n+1}\right\|} \right) \left(\frac{180^\circ}{\pi} \right) \right\} \tag{3.13}$$

And the minimum included angle of the triangle, measured in degrees, is

$$Q(K) = \min_{n\in\{0,1,2\}} \left\{ \arccos\left(\frac{\vec{L}_n \cdot \vec{L}_{n+1}}{\left\|\vec{L}_n\right\| \left\|\vec{L}_{n+1}\right\|} \right) \left(\frac{180^\circ}{\pi} \right) \right\} \tag{3.14}$$

Chosen geometric criterion

Here, to choose the optimal mesh, the quality of each element in the candidate mesh topologies is evaluated by:

$$Q(K) = \frac{|K|}{h_K^d} \tag{3.15}$$

where d is the space dimension and h_K the mean of the lengths of the edges. For each candidate mesh topology, the worst elements are compared, then the geometrical quality principle selects the mesh topology having the best quality among these elements, therefore handling the optimization of elements' shapes.

3.2.4.2 Algorithm of the topological mesh generator

Based on the local mesh optimization already defined, the final mesh can then be generated through an iterative procedure until the optimal mesh with minimum volume is achieved. Algorithm 1 explains in detail the iterative procedure.

Algorithm 1 Optimization of a Mesh Topology

Input $(\mathcal{N}, \partial\mathcal{T})$ a mesh topology of Ω

Output $(\mathcal{N}_o, \partial\mathcal{T}_o)$ the optimal mesh topology obtained

1: **while** $\sum_{K\in\mathcal{T}} |K| \geq |\Omega|$, the volume isn't yet optimal, **do**

2: **for** each vertex and edge in $\mathcal{T}$ **do**

3: Remove the local topology $\mathcal{T}_\mathcal{A}$ associated with the specific vertex or edge

4: Replace $\mathcal{T}_\mathcal{A}$ with new mesh topology $\mathcal{T}_\mathcal{B} = \mathcal{T}^*(S_\mathcal{A}, \partial\mathcal{T}_\mathcal{A})$ that minimizes the volume and maximizes the local mesh quality

5: Update $\mathcal{T} = \bigcup_\mathcal{B} \mathcal{T}_\mathcal{B}$

A local topology $\mathcal{T}_\mathcal{A}$ associated to a vertex $\mathcal{S}$ is composed of the elements of $\mathcal{T}$ whose all vertices belong to $\{\{\mathcal{S}\} \cup \mathcal{N}(\mathcal{S})\}$, the set of vertices neighbor to $\mathcal{S}$. In the configuration represented in Figure 3.7 the candidate topologies to achieve a star formation are:

- Eliminating both vertices $\mathcal{S}_0$ and $\mathcal{S}_6$ and creating a mesh topology around the centroid $\mathcal{C}$
- Keeping point $\mathcal{S}_0$ and removing vertex $\mathcal{S}_6$
- Eliminating both vertices $\mathcal{S}_0$ and $\mathcal{S}_6$ and creating a mesh topology around any of the remaining vertices $\mathcal{S}_1$ to $\mathcal{S}_5$ (5 possible configurations)

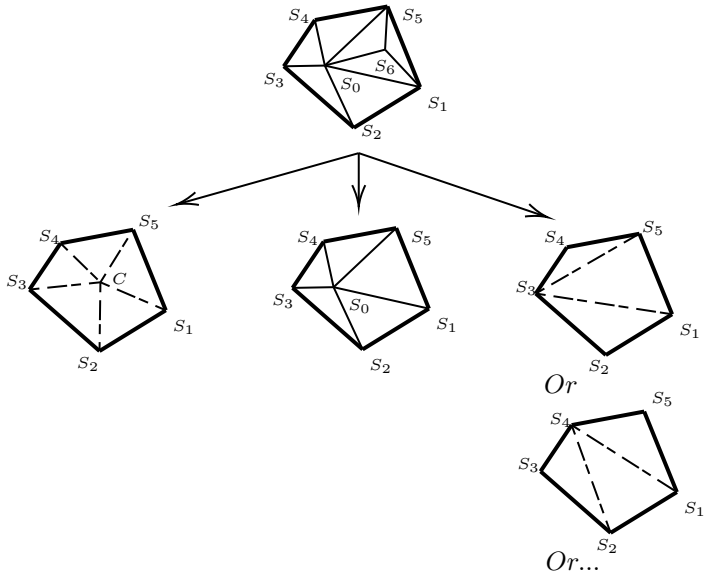


Figure 3.7: Example of the local optimization process applied on a vertex. Adopted from [11]

The same steps can be applied to an initial topology associated with an edge.

Note that, the boundary patches are given special treatment. Indeed, in order to eliminate or remove a boundary vertex $\mathcal{S} \in \partial\mathcal{T}_{\mathcal{A}}$, each one of the boundary faces is connected to a virtual vertex creating new virtual elements that do not contribute to the mesh without affecting the overall volume of the mesh (Figure 3.8).

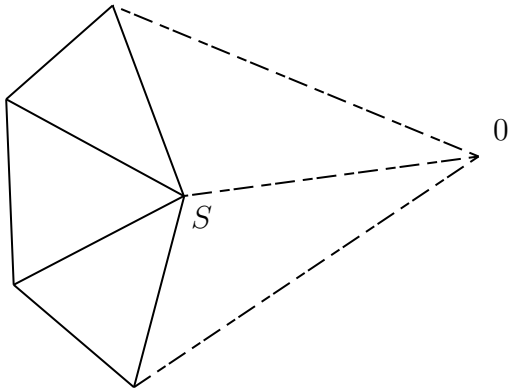
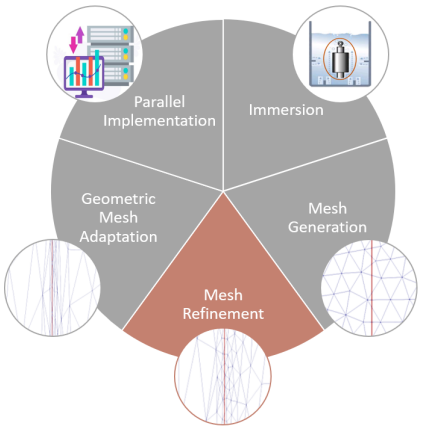


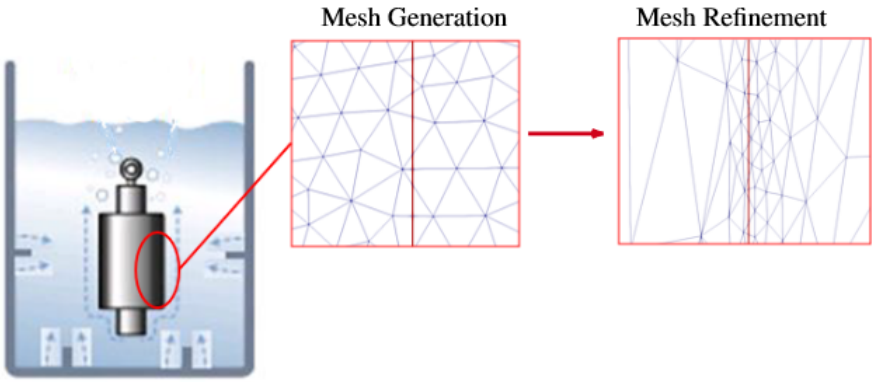
Figure 3.8: Mesh topology with virtual elements.

3.3 Adaptive Mesh Refinement

Dealing with multi-component problems and turbulent flows may become very complex due to the development of discontinuities or vortices. That's why a mesh allowing good precision and accuracy during the entire simulation is needed. These types of meshes aren't cheap: they require a large number of fine isotropic elements especially in the regions of discontinuity making the resolution of the partial differential equations very expensive. In order to sustain the high resolution and accuracy of the solution during the entire simulation, methods have been developed to dynamically adapt the mesh keeping a minimum of mesh elements.



(a)



(b) From isotropic to anisotropic mesh.

Figure 3.9: Mesh refinement framework.

3.3.1 Metric-based Anisotropic Mesh Adaptation

In this work, anisotropic mesh adaptation, based on [14], is used and described in what follows.

Anisotropic mesh adaptation is a powerful method that not only increases the quality of the mesh and the accuracy of the numerical solution but also reduces significantly their CPU time. The domain discretization is accomplished following the size and directional constraints, hence, mesh elements are concentrated in regions with high gradients and large discontinuities allowing their capture with higher accuracy. The anisotropic mesh algorithm allows also to control the number of vertices and to get the smallest error possible, therefore the algorithm is designed to build the mesh, compute the numerical solution and evaluate an estimation of the interpolation error. A local geometrical transformation is applied to each element before quality measurements. To obtain the optimal metric value, an edge based error estimator and a gradient recovery procedure are defined. The mesh is then generated according to the new metric field [15, 16].

3.3.2 Edge-based Error Estimation

Let u_h be a $\mathbb{P}_1$ finite element approximation obtained by applying the Lagrange interpolation operator $\mathbf{\Pi}$ to a function $u \in C^2(\Omega)$ and $x = \left\{x_i \in \mathbb{R}^d, i = 1, ...N\right\}$ the set of coordinates of vertices v_i in the mesh. For each vertex v_i of the mesh, let $U_i = u(x_i) = u_h(x_i)$ and $\Gamma(i)$ the set of vertices or "patch" connected to x_i by a common edge x_{ij} such in Figure 3.10, such that $x_{ij} = x_j - x_i$ and $U_{ij} = U_j - U_i$.

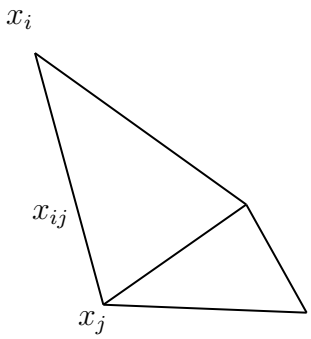


Figure 3.10: Representation of $\Gamma(i)$ and of the edge x_{ij} connecting vertices v_i ands v_j.

The gradient $\nabla u_h \cdot x_{ij}$ on the edge x_{ij} is continuous, therefore we can write:

$$U_j = U_i + \nabla u_h \cdot x_{ij} \tag{3.16}$$

which leads to

$$\nabla u_h \cdot x_{ij} = U_j - U_i \tag{3.17}$$

From [16], the error estimator can be defined as:

$$\| \nabla u_h \cdot x_{ij} - \nabla u(x_i) \cdot x_{ij} \| \leq \max_{y \in |x_i, x_j|} | x_{ij} \cdot H_u(y) \cdot x_{ij} | \tag{3.18}$$

with H_u the Hessian of u. At x_i, the gradient g_i of u_h can be written as:

$$\nabla g_h \cdot x_{ij} = g_j - g_i \tag{3.19}$$

Hence, the projection of the Hessian based on the gradient at the extremities of the edge is:

$$(\nabla g_h \cdot x_{ij}) \cdot x_{ij} = (g_j - g_i) \cdot x_{ij} \tag{3.20}$$

$$(H_u \cdot x_{ij}) \cdot x_{ij} = g_{ij} \cdot x_{ij} \tag{3.21}$$

with $g_{ij} = g_j - g_i$. From [16], it can be shown that $| g_{ij} \cdot x_{ij} |$ gives a second order accurate approximation of the second derivative of u along the edge x_{ij}. The error err_{ij} along the edges can then be defined as:

$$err_{ij} = | g_{ij} \cdot x_{ij} | \tag{3.22}$$

Equation 3.22 is the exact interpolation error along the edge and allows the evaluation of the global L_2 error. However, the interpolation error can only be evaluated when the gradient at the vertices is known and continuous on the vertices, thus a recovery method needs to be considered.

3.3.3 Gradient recovery procedure

The gradient recovery operator is defined by a local optimization problem:

$$G_i = arg \min_{G} \left(\sum_{j \in \Gamma(i)} | (G - \nabla u_h) \cdot x_{ij} |^2 \right) \tag{3.23}$$

where G_i is the recovered gradient. Let X_i be the length distribution tensor at vertex v_i defined using the vector product $\otimes$ as:

$$X_i = \frac{1}{| \Gamma(i) |} \left(\sum_{j \in \Gamma(i)} x_{ij} \otimes x_{ij} \right) \tag{3.24}$$

The recovered gradient can then be expressed in terms of the length distribution tensor as follows:

$$G_i = (X_i)^{-1} \sum_{j \in \Gamma(i)} U_{ij} x_{ij} \tag{3.25}$$

and finally, the estimated error err_{ij} is written as:

$$err_{ij} = G_{ij}.x_{ij} \tag{3.26}$$

3.3.4 Metric Construction

In order to link the error variation to the changes in the length of the edges, a stretching factor s_{ij} is introduced. However, it's insufficient to only enlarge the edge if the error is too small or to reduce it or break it if it is too large, the neighborhood of the vertex must be taken into account: a metric is thus the best averaging representation. It is defined as follows:

$$\widetilde{M}_i = (\widetilde{X}_i)^{-1} \tag{3.27}$$

where

$$\widetilde{X}_i = \frac{1}{\mid \Gamma(i) \mid} \left(\sum_{j \in \Gamma(i)} s_{ij} \otimes s_{ij} \right) \tag{3.28}$$

The stretching factor s_{ij} of the edge x_{ij} is chosen so that the total number of vertices in the mesh is kept fixed and is defined as

$$s_{ij} = \left(\frac{err_{ij}}{err(N)} \right) \tag{3.29}$$

where $err(N)$ the total error.

3.3.5 Mesh Adaptation

In a multi-component application, the interface between the solid and the liquid need to be modeled accurately. Hence, the mesh should be adapted to several variables, like the velocity and the level-set function. The most common way is to define a metric for each variable and combine them using metric intersection operation. However, in this work, this operation is simplified and one metric is used to account for the changes in all variables. Equation 3.22 is extended to consider all sources of error defined in vector $\mathbf{v}(x_i)$:

$$\mathbf{v}(x_i) = \left\{ \frac{V_i}{\mid V^i \mid}, \frac{\mid V^i \mid}{\max_j \mid V^j \mid}, \frac{\phi}{\max(\phi)} \right\} \tag{3.30}$$

Note that since all fields are normalized, the variations of all the variables are taken into account.

Figure 3.11 compares the interface captured by an isotropic and anisotropic mesh for the same immersed geometry. Applying the anisotropic mesh adaptation on the level-set function, the elements surrounding the interface are stretched along its direction and hence define it more precisely. Figures 3.12 and 3.13 show the anisotropic mesh adaptation near the interface of the 2D and 3D immersed geometries. Comparing the initial and final mesh, the geometries are better defined, however, the interface of the immersed object cuts the elements of the mesh. Hence, the objective of this work is to accurately capture the interface and create an anisotropic fitted mesh: a new geometrical adaptation method is proposed.

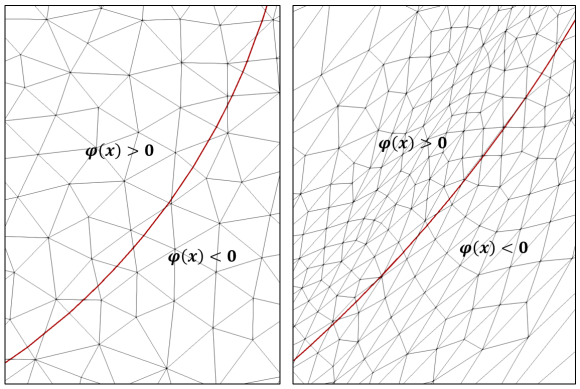
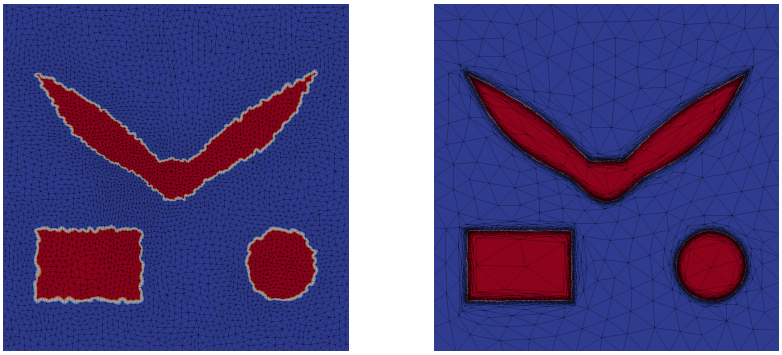


Figure 3.11: A level-set function computed on an isotropic (left) and anisotropic (right) meshes.

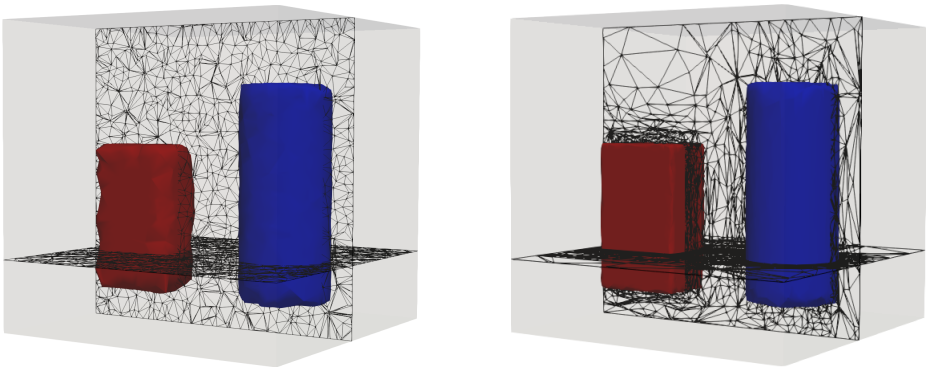


(a) Initial mesh (b) Final mesh

Figure 3.12: Mesh adaptation Algorithm applied on 2D immersed bodies.

3.4 Numerical Computation of the Level-set function

As seen in 2.1, the level-set or signed distance function ϕ of an interface Γ is used to determine the position of the interface of the immersed body. For any point x_i in the domain Ω, the level-set function corresponds to the shortest distance to the interface Γ. This is achieved by determining the elements with the minimum surface distance, which is the shortest distance value from the surface to x_i with $|\mathbf{d}| = min_{e=1}(|\mathbf{d_e}|)$.
In order to find the minimum distance $|\mathbf{d_e}|$ in each element a bounding box check is done. It determines if their exists another candidate element closer to fx_i based on the current $|\mathbf{d}|$. If x_i is outside of the bounding box then the current minimum $|\mathbf{d}|$ remains however if it lies inside it, a projected volume is done.
The general idea of projected volume check consists of determining whether x_i lies inside or outside the projected volume of an element:



(a) Initial mesh (b) Final mesh

Figure 3.13: Mesh adaptation algorithm applied on 3D immersed bodies.

– if x_i is inside, then $|\mathbf{d_e}|$ is taken as the shortest distance between x_i and the plane defined by the element face normal $|\mathbf{n_e}|$. The condition where this is satisfied is: $(x_{1i} \cdot \mathbf{n_3} \leqslant 0) \cap (x_{2i} \cdot \mathbf{n_1} \leqslant 0) \cap (x_{3i} \cdot \mathbf{n_2} \leqslant 0)$, where $x_{ij} = x_j - x_i$ and $\mathbf{n_1}, \mathbf{n_2}$ and $\mathbf{n_3}$ are the edge normals.

– If x_i lies outside the projected volume of the element, each edge of the triangle is examined to get the shortest distance to any of the line segments $|\mathbf{d_e}|$.

Finally, the level-set function $\phi = |\mathbf{d}| \frac{x_{ic} \cdot \mathbf{n_e}}{|x_{ic} \cdot \mathbf{n_e}|}$, where $x_{ic} \cdot \mathbf{n_e}$ will be positive inside the immersed geometry Ω_s and negative in the rest of domain Ω_f [17]:

$$\phi(x) = \begin{cases} dist(x, \Gamma) & \text{if} \quad x \in \Omega_s \\ 0 & \text{if} \quad x \in \Gamma \\ -dist(x, \Gamma) & \text{if} \quad x \notin \Omega_s \end{cases} \tag{3.31}$$

The interface of the immersed object is determined by the zero-value of ϕ (Figure 3.14). Further details on how to compute the signed distance function can be found in [18–21].

In order to decrease the computational time cost needed to compute all $|\mathbf{d_e}|$, a hierarchical representation of the surface mesh is done [18, 22, 23]. A box tree is created by dividing the domain into levels of small boxes where the lowest level contains the entire mesh. Children boxes are recursively created by packing the elements of the mesh and determining the parent box that contains elements of the immersed interface or surface elements. Hence, the distance between x_i and a box $\mathcal{C}_i$ is evaluated before computing the distance between x_i and the children of $\mathcal{C}_i$ or its elements.

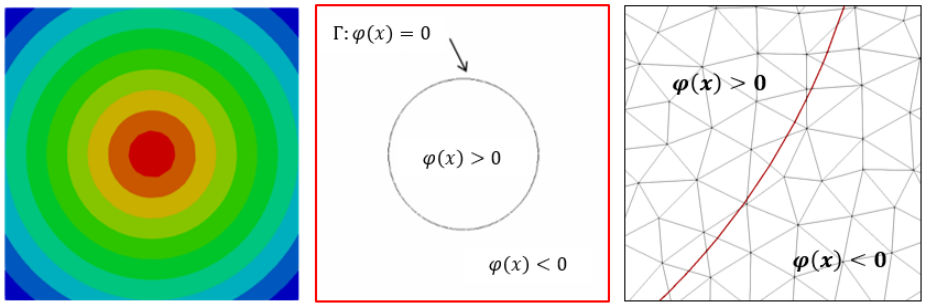


Figure 3.14: A level-set function (in red) for a circle separating two domains.

3.5 Conclusion

As seen in this chapter, mesh generation and mesh adaptation are very important tools for numerical simulations and modeling, especially when dealing with the solution's accuracy. The basic notions for the study and generation of meshes have been recalled and the operations of the MTC mesher, used in this thesis, have been described [11, 12]. The dynamic mesh adaptation based on edge-based error estimation constructing metric-based anisotropic meshes has also been explained. The numerical computation and detection of the level-set function have been detailed, the level-set function tracks and detects the interface of the immersed geometry. This property allows then the fixation of the boundary conditions and resolution of physical phenomena.

However, with the methods described so far, the mesh generated doesn't necessarily conforms to the immersed geometry. This non-conformity has been a challenge to researchers, especially when dealing with complex problems with high gradients and sharp discontinuities. The next chapter describes the existing methods that have attempted to solve this problem, and our proposed method to generate an anisotropic fitted mesh is presented.

Bibliography

[1] R. Löhner, P. Parikh, Generation of three-dimensional unstructured grids by the advancing-front method, International Journal for Numerical Methods in Fluids 8 (10) (1988) 1135–1149. 32

[2] S. Lo, A new mesh generation scheme for arbitrary planar domains, International Journal for Numerical Methods in Engineering 21 (8) (1985) 1403–1426. 32

[3] B. Delaunay, et al., Sur la sphere vide, Izv. Akad. Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk 7 (793-800) (1934) 1–2. 33

[4] F. P. Preparata, M. I. Shamos, Computational geometry: an introduction, Springer Science & Business Media, 2012. 33

[5] J.-D. Boissonnat, M. Yvinec, Algorithmic geometry, Cambridge university press, 1998. 33

[6] N. P. Weatherill, O. Hassan, Efficient three-dimensional delaunay triangulation with automatic point creation and imposed boundary constraints, International Journal for Numerical Methods in Engineering 37 (12) (1994) 2005–2039.

[7] H. Borouchaki, P. L. George, Meshing, Geometric Modeling and Numerical Simulation 1: Form Functions, Triangulations and Geometric Modeling, John Wiley & Sons, 2017. 33

[8] H. Borouchaki, F. Alauzet, P. Laug, P. L. George, A. Loseille, L. Maréchal, Meshing, Geometric Modeling and Numerical Simulation, Volume 2: Metrics, Meshes and Mesh Adaptation, John Wiley & Sons, 2019. 34

[9] M. A. Yerry, M. S. Shephard, Automatic three-dimensional mesh generation by the modified-octree technique, International Journal for Numerical Methods in Engineering 20 (11) (1984) 1965–1990.

[10] M. Yerry, M. Shephard, A modified quadtree approach to finite element mesh generation, IEEE Computer Graphics and Applications 3 (01) (1983) 39–46. 34

[11] T. Coupez, Grandes transformations et remaillage automatique, Ph.D. thesis, Paris, ENMP (1991). v, 35, 40, 47

[12] T. Coupez, Génération de maillage et adaptation de maillage par optimisation locale, Revue européenne des éléments finis 9 (4) (2000) 403–423. 35, 36, 47

[13] P. Pébay, T. Baker, Analysis of triangle quality measures, Mathematics of Computation 72 (244) (2003) 1817–1839. 37

[14] T. Coupez, Metric construction by length distribution tensor and edge based error for anisotropic adaptive meshing, Journal of Computational Physics 230 (7) (2011) 2391–2405. 42

[15] C. Gruau, T. Coupez, 3d tetrahedral, unstructured and anisotropic mesh generation with adaptation to natural and multidomain metric, Computer Methods in Applied Mechanics and Engineering 194 (48-49) (2005) 4951–4976. 42

[16] T. Coupez, E. Hachem, Solution of high-reynolds incompressible flow with stabilized finite element and adaptive anisotropic meshing, Computer Methods in Applied Mechanics and Engineering 267 (2013) 65–85. 42, 43

[17] J. A. Sethian, A fast marching level set method for monotonically advancing fronts, Proceedings of the National Academy of Sciences 93 (4) (1996) 1591–1595. 46

[18] J. Bruchon, H. Digonnet, T. Coupez, Using a signed distance function for the simulation of metal forming processes: Formulation of the contact condition and mesh adaptation. from a lagrangian approach to an eulerian approach, International Journal for Numerical Methods in Engineering 78 (8) (2009) 980–1008. 46

[19] P. E. DesJardin, B. T. Bojko, M. T. McGurn, Initialization of high-order accuracy immersed interface cfd solvers using complex cad geometry, International Journal for Numerical Methods in Engineering 109 (4) (2017) 487–513.

[20] R. N. Elias, M. A. Martins, A. L. Coutinho, Simple finite element-based computation of distance functions in unstructured grids, International Journal for Numerical Methods in Engineering 72 (9) (2007) 1095–1110.

[21] Q. Zhu, J. Yan, A mixed interface-capturing/interface-tracking formulation for thermal multi-phase flows with emphasis on metal additive manufacturing processes, Computer Methods in Applied Mechanics and Engineering 383 (2021) 113910. 46

[22] K. Wang, J. Grétarsson, A. Main, C. Farhat, Computational algorithms for tracking dynamic fluid–structure interfaces in embedded boundary methods, International Journal for Numerical Methods in Fluids 70 (4) (2012) 515–535. 46

[23] K. Wang, A. Rallu, J.-F. Gerbeau, C. Farhat, Algorithms for interface treatment and load computation in embedded boundary methods for fluid and fluid–structure interaction problems, International Journal for Numerical Methods in Fluids 67 (9) (2011) 1175–1206. 46

Chapter 4

Anisotropic Fitted Algorithm for Immersed Geometries

Contents

Résumé en Français

Le développement de méthodes efficaces pour simuler des systèmes multi-composants fait partie des défis d'ingénierie et reste un besoin pour une variété d'applications industrielles, en particulier dans le cas de l'interaction fluide-structure ou du transfert de chaleur conjugué. Ces dernières années, un intérêt croissant a été porté à l'étude numérique d'une variété d'applications d'ingénierie qui impliquent de tels couplages entre des domaines fluides et solides.
Classiquement, les techniques de couplage consistent à diviser le domaine global en plusieurs sous-domaines locaux où un modèle local (équation à résoudre) peut être analysé indépendamment sur chaque sous-domaine. La solution globale peut alors être construite en assemblant de manière appropriée les solutions locales des sous-domaines modélisés individuellement [1, 2]. Un maillage adapté au corps immergé est généralement nécessaire pour de telles simulations et sa construction peut être limitée en raison des coûts de calcul nécessaires à cause du remaillage ou de la complexité des géométries traitées.
Alternativement, le développement de méthodes immergées ou incorporées devient un sujet de recherche intense principalement en raison de leur simplicité relative et de leur efficacité de calcul pour la simulation de géométries complexes [3–5]. En effet, il n'y a pas de contrainte pour la génération du maillage, différentes approches existes et sont décrites dans ce chapitre. Un défi principal commun lié au développement de ces méthodes est le fait que les éléments à l'interface sont coupés de telle manière qu'une fraction d'entre eux reste à l'intérieur du domaine solide, et l'autre dans le domaine fluide, et par conséquent la question de savoir comment imposer les conditions limites à l'interface.
L'objectif est de pouvoir bénéficier des avantages des deux méthodes : la méthode des corps ajustés et la méthode immergée. Dans ce chapitre, après une brève description des méthodes existantes, un aperçu général de la méthode proposée est décrite pour une adaptation géométrique permettant la création d'un maillage ajusté anisotrope pour les géométries immergées.

4.1 Introduction

As seen in the above chapters, the development of efficient methods to simulate multi-component systems is among engineering challenges and still a need for a variety of industrial applications, especially in the case of fluid-structure interaction or conjugate heat transfer. In recent years, there has been an increasing interest in studying numerically a variety of engineering applications that involve such couplings between fluid and solid domains.

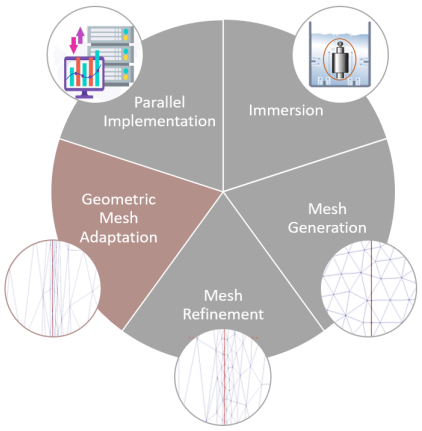
Classically, coupling techniques consist of dividing the global domain into several local subdomains over each of which a local model (equation to be solved) can be analyzed independently. The global solution can then be constructed by suitably piecing together local solutions from individually modeled subdomains [1, 2]. A body-fitted mesh is generally required for such simulations and its construction may be limited due either to the needed computational costs due to repeated remeshing or to the complexity of the treated geometries.

Alternatively, the development of immersed or embedded methods is becoming a subject of intense research mainly because of their relative simplicity and computational efficiency for simulating complex geometries [3–5]. Indeed, there is no constraint for the mesh generation, different approaches are proposed to enforce strongly or weakly the boundary conditions at the fluid-solid interface [6–10], and finally several clever techniques such as penalty or enrichment methods can be used to ensure continuity and to increase accuracy at the interface [11–16]. One common main challenge related to the development of these methods is in fact that elements at the interface are cut in such a way that a fraction of them remains inside the solid domain, and the other in the fluid domain, and consequently the question of how to impose a boundary condition at the interface. The objective is to be able to benefit from the advantages of both methods: body-fitted and immersed methods. In this chapter, after a brief description of the existing methods, a general overview of the proposed method is described for a geometric adaptation allowing the creation of an anisotropic fitted mesh for immersed geometries.

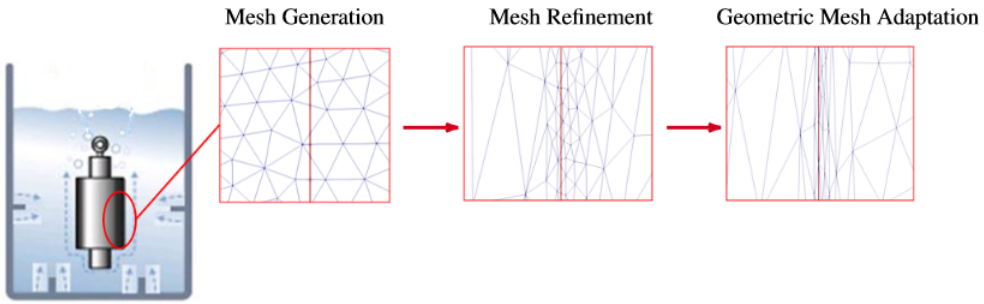
4.2 Existing Methods

In order to solve multi-component systems, the mesh generated should be adapted on the entire domain to accurately detect the boundary layer and capture the entire flow phenomena. To tackle this challenge, different methods have been developed and can be divided into three main categories:

- Modifying the numerical scheme
- Modifying the governing equations



(a)



(b) From cut cells to fitted mesh.

Figure 4.1: Geometric mesh adaptation framework.

- Altering the mesh

The diagram shown in figure 4.2 summarizes the existing embedded geometry techniques.

In this section, some of the existing methods used to solve multi-component and multi-phase flows problems will be briefly described.

4.2.1 Modifying the Numerical Scheme

Enriched finite element methods are based on the decomposition of the solution into a classical Galerkin finite element part and an enriched part. The enrichment can be *global or local*. The local enrichment method is known to be mostly used to approximate a solution with discontinuous characteristics and singularities in the computational domain.

Recently, the X-FEM gained a lot of attention and is continuously developed and

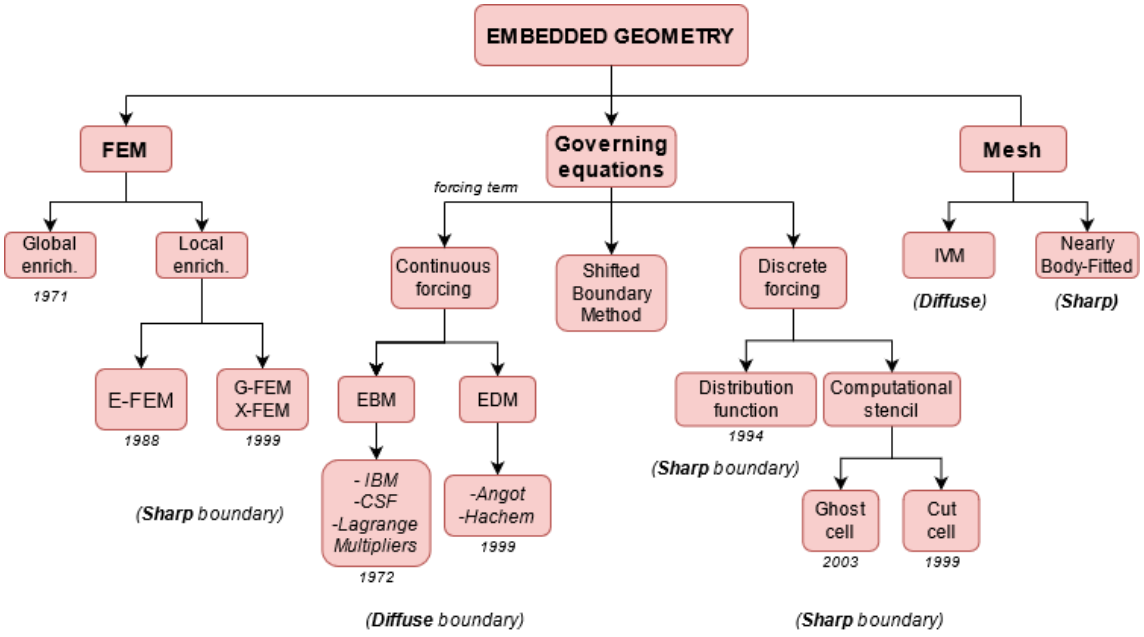


Figure 4.2: Diagram summarizing the existing embedded geometry techniques. Adapted from [17]

E-FEM: Enriched Finite Element Method, G-FEM: Generalilzed Finite Element Method, X-FEM: Extended Finite Element Method, IVM: Immersed Volume Method, EBM: Embedded Boundary Method, EDM: Embedded Domain Method.

used with structured and unstructured mesh. The embedded geometry is described using the level-set method. The cut elements are determined and the enriched vertices defined. These vertices will be used to handle the embedded boundary. Based on the type of discontinuity at hand, the enrichment function is chosen. The X-FEM formulation is applied near discontinuities and singularities as follows:

$$u^h(x) = \underbrace{\sum u_k \phi_k(x)}_{classic-FEM} + \underbrace{\sum u_{k*} \phi_{k*}(x) F(x)}_{local-enrichment} \tag{4.1}$$

where:
x = the spatial position (x,y,z)
u_k = the degree of freedom at node n_k
ϕ_k = the shape function
ϕ_{k*} = the partition of unity function
u_{k*}= the unknown at the enriched node
$F(x)$ = the enrichment function

Details of the extended finite element method can be found in [6, 18].

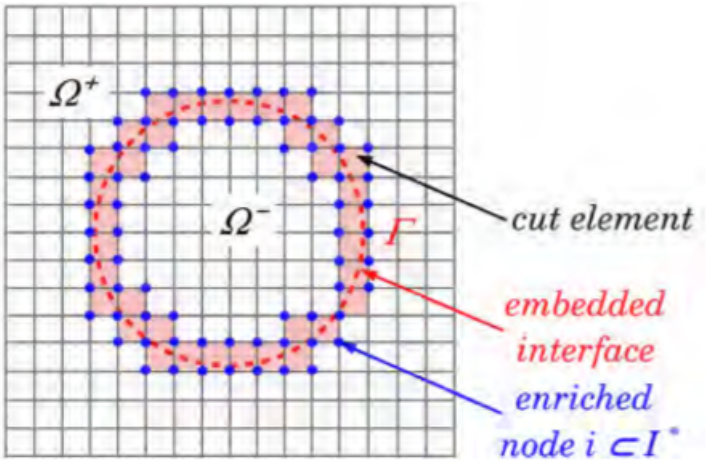


Figure 4.3: Extended finite element approach [17].

4.2.2 Modifying the Governing Equations

Other methods add a forcing term to the governing equations near the boundary layer. Depending on the approach, the forcing term can be introduced before or after discretization on the whole domain or only on the boundary of the embedded geometry.

The forcing term introduced before discretization reproduces the effect on the fluid of the embedded boundary. Two categories in continuous forcing can be distinguished:

- The Embedded Boundary Methods (EBM): here, the forcing term represents the effect of the embedded boundary on the fluid. The well known Immersed Boundary Method (IBM) [3, 4] belongs to this category.
- The Embedded Domain Methods (EDM): the forcing term penalizes some quantity over the entire domain such in the Immersed Stress Model [2].

4.2.2.1 The Ghost Cell Method

Introducing the forcing term into the governing equations after discretization results in extracting it directly from the numerical solution rather than introducing it in the continuous momentum equations. The governing equations are discretized on a computational grid, resulting in a set of discretized equations. The forcing term is then introduced only for cell points close to the immersed boundary to account for it. The Ghost Cell Method [11] falls in this category.

The Ghost cell method is widely used to solve two-phase flows and obtaining an accurate description of the interface. Ghost cells are generally defined as cells in the solid domain and neighboring at least one fluid cell. The general idea of this approach is to impose the boundary condition on the interface to a ghost vertex inside a body by interpolation or extrapolation. Below is a brief description of the procedure of the Ghost Cell Method:

Once the immersed boundary is identified, the computational domain is divided into two: the physical domain and the ghost-cell domain. Hence, the ghost-cells are determined. Each ghost-cell is associated with a boundary point located inside the cell, and interpolation points that are usually nearest fluid vertices or image vertices. Depending on the application, an interpolation scheme is defined and accordingly the boundary condition is imposed.

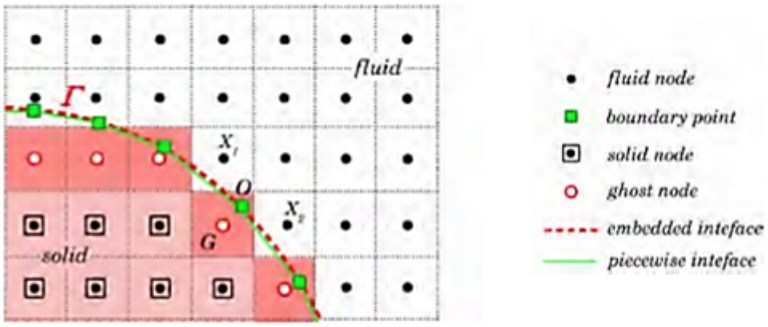


Figure 4.4: Ghost cell Method. Figure taken from [17].

In [11], Tseng et al. calculate the ghost-cell value by extrapolation from the nearest fluid vertices. Referring to figure 4.4, G is the ghost vertex and X_1, X_2 are the fluid vertices. The point O is the boundary point at which the boundary condition needs to be satisfied. To determine the positions of O, the authors propose two approaches: O can be chosen as the midpoint of the boundary segment and should be within the ghost-cell or O is computed such that $\overrightarrow{GO}$ is normal to the boundary. Flow variables are expressed in terms of linear or quadratic polynomials in order to satisfy and reconstruct the boundary conditions. More details on how to impose the boundary conditions can be found in [11].

4.2.2.2 The Shifted Boundary Method

The Shifted Boundary Method (SB), presented in 2018 by Main et al. [9, 10], shifts the location where the boundary conditions are applied to a surrogate boundary and then weakly enforces the shifted boundary conditions (Figure 4.5).

The Shifted boundary method proposes the following Taylor expansion in order to impose the Dirichlet boundary condition u_D on the surrogate boundary $\tilde{\Gamma}$:

$$u(\tilde{x}) + \nabla u(\tilde{x}) \cdot \mathbf{d}(\tilde{x}) - u_D(\tilde{x}) + O(||\mathbf{d}(\tilde{x})||^2) = 0 \tag{4.2}$$

where the distance vector $\mathbf{d}$ between the boundary and its surrogate is defined as: $\mathbf{d} = ||d||\mathbf{n}$ with $\mathbf{n}$ being the normal to the original boundary.

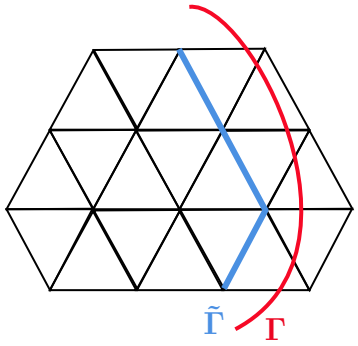


Figure 4.5: Shifted Boundary Method - The true and surrogate boundary.

Considering the Poisson problem with Dirichlet boundary condition:

$$\begin{aligned} -\Delta u &= f, \quad \text{on}\Omega \\ u &= u_D, \quad \text{on}\Gamma \end{aligned} \tag{4.3}$$

Nitsche's method introduces a penalty term to equation 4.3 in order to weakly enforce the Dirichlet boundary condition on Γ as well as the partial differential equation on Ω. The details of this method can be found in [9].

4.2.3 Altering the Mesh

The methods mentioned above require the addition of a forcing term. This category includes two methods that tackle the multi-component system through the use of a **Monolithic formulation**. A Monolithic resolution consists in considering the computational domain and the immersed object as one whole domain, and then solve one set of equations on this domain. Two models fall in this category:

- The *Nearly* Body Fitted mesh [17] introduced by Quan et al. in 2014.
- The Immersed Volume Method (IVM) [1, 19]

4.2.3.1 Nearly Body Fitted

The Nearly body fitted (NBF) mesh is a recent study inspired from the IVM. The difference between the two methods is that the virtual boundary of anisotropic mesh created in IVM is replaced by a real interface approximation in the NBF method.

The basic idea of this method is to generate a refined mesh around the embedded geometry and to impose the boundary conditions on the vertices closest to the interface. These vertices form an approximation Γ^* of the interface (Figure 4.6). The novelty here is that the mesh adaptation is done to both the geometry and the flow using a mesh intersection technique.

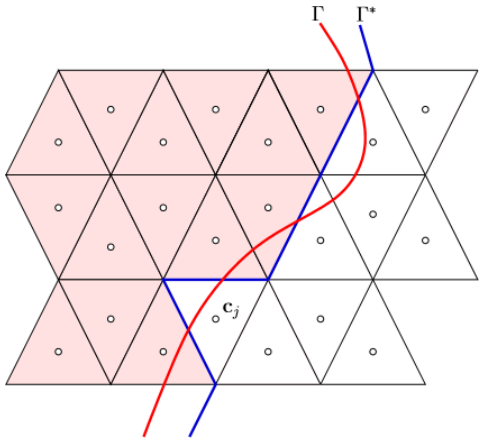


Figure 4.6: A discrete approximation Γ^* of the level-set Γ. Figure taken from [17].

The first metric is a level-set based metric. At each vertex, a metric is built based on the gradient $\nabla\phi(x)$ and the Hessian $\mathcal{H}(\phi(x))$ of the distance function. The mesh metric $\mathcal{M}_{LS}$ obtained is given by:

$$\mathcal{M}_{LS} = \mathcal{R}^T \begin{pmatrix} \lambda_n & 0 & 0 \\ 0 & \lambda_{t1} & 0 \\ 0 & 0 & \lambda_{t2} \end{pmatrix} \mathcal{R} \tag{4.4}$$

where $\mathcal{R} = (n, t_1, t_2)^T$ and the eigenvalues λ_n,λ_{t1} and λ_{t2} are computed from the mesh sizes h_n and h_{ti}.
Equation (4.4) is applied in a narrow band region in the vicinity of the interface. Outside this band, an isotropic element mesh size h_b is defined such as:

$$\mathcal{M}_{LS} = \begin{pmatrix} \frac{1}{h_b^2} & 0 & 0 \\ 0 & \frac{1}{h_b^2} & 0 \\ 0 & 0 & \frac{1}{h_b^2} \end{pmatrix} \tag{4.5}$$

The second mesh metric is based on the *Hessian* of the flow. It is computed as follows:

$$\mathcal{M}_{\mathcal{H}} = \alpha \mathcal{H}(|\mathbf{u}(x)|) \tag{4.6}$$

with α a factor of proportionality. Details on how to compute α can be found in [17].

Finally, a metric intersection is done in order to keep the smallest mesh size and conserve the orientation of the most anisotropic metric.

4.2.3.2 The Immersed Volume Method

The Immersed Volume Method (IVM) is an interesting approach for computational problems [1, 20]. It treats and solves the different subdomains as one singular domain with variable material properties. The IVM consists of three main components.

1. The geometry of the immersed solid is treated, and implicitly represented using the **sign distance function** described in section 3.4: the level-set function. Hence, the interface position of the immersed object is detected and the different subdomains defined.

2. **Anisotropic mesh adaptation**, described in detail in Chapter 3, is employed to provide an accurate configuration of the problem. The adaptation can be applied to the level-set function, providing a flexibility to capture details and with high fidelity any complex geometry, and/or to the solution of the set of equations to be solved. This will yield to the capture of the fluid-solid interface at low computational cost with well oriented stretched elements allowing a good capture of discontinuities and high gradients.

3. Finally, the **Mixing law** (referred to in section 2.3) is used to distribute the different components' properties on the different subdomains.

The IVM can be applied without the modification of any geometry or any physical property. It can be very easily implemented and applied with stabilized finite element methods.

4.3 Objective

The objective of each method listed above is to accurately track and define the interface. Our goal is to extend the IVM (section 4.2.3.2) to benefit from the properties of an anisotropic mesh that smoothly follows the interface and from there develop a system that enables us to get a robust and sharp interface by creating new elements or adapting some of the existing elements to intersect with the interface.

The anisotropic mesh elements stretched in the right direction in the Immersed Volume Method describe well the immersed geometry with all its angles and curvatures. It offers a high fidelity resolution at the interface creating a well defined boundary layer between the two domains (solid and fluid). This method guarantees a precise mesh on the fluid-solid interface in order to capture more precisely the thermal gradients and the strong discontinuity of the physical properties. It therefore offers great flexibility in the setting of FSI problems.

In a Body Fitted (BF) approach, isotropic mesh elements accurately define the geometry at the interface where there is no longer a need for a narrow band region. However, this method cannot support high thermal gradients and strong discontinuities of the physical properties.

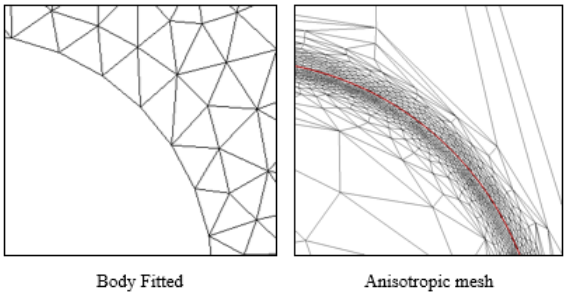


Figure 4.7: A close up comparison between an anisotropic and a body fitted mesh.

The aim is to define a technique that will allow to accurately capture the interface of immersed objects creating a body fitted mesh. The result will combine both the flexibility of the immersed method and the accuracy of a body fitted mesh.

The order of implementation of the two methods is of high importance: anisotropic mesh adaptation is applied first to create the refined region near the interface, hence stretching the elements and moving the vertices closer to it. Then, a BF approach is implemented in order to accurately capture the interface.

4.4 Anisotropic Adaptive Fitted Mesh

This section describes the general principles of the new fitting technique that can capture the boundary between the fluid and the solid. It's based on capturing the interface following a geometrical adaptation. Geometry based adaptation consists of modifying the elements of the mesh:

- by relocating mesh vertices to some regions of interest to better capture the desired physical and mechanical properties - this is known as **R-adaptation** (Section 4.4.1),

- by inserting new vertices in the region of interest and creating new elements that would need to be remeshed. The introduced vertices are the intersection points between two vertices v_i and v_j, and whose coordinates are computed by linear interpolation using the following equation:

$$x_{ij} = x_i - (x_i - x_j)\frac{\phi(x_i)}{\phi(x_i) - \phi(x_j)} \tag{4.7}$$

  with ϕ the level-function computed at each vertex.

- by inducing a change in the overall topology of the mesh like in **edge-swapping** or edge-flipping [21] while conserving the number of elements (Section 4.4.1).

4.4.1 Geometric Adaptation for a Fitted Mesh

The steps of the proposed technique are illustrated in the diagram of figure 4.8. Algorithm 2 summarizes the overall technique and the details are explained below (Algorithms 3 to 5).

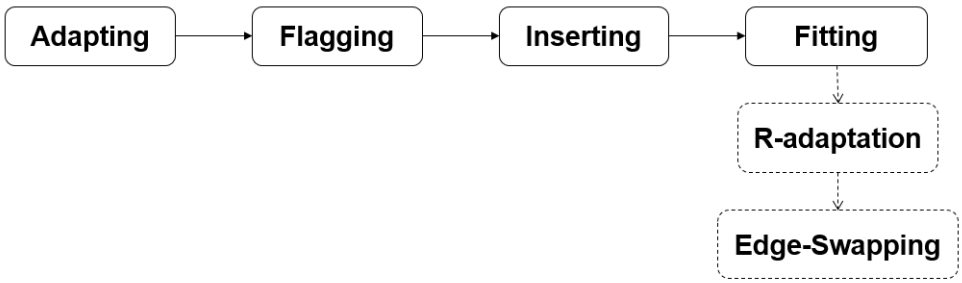


Figure 4.8: Scheme for applying the Geometric Adaptation.

Algorithm 2 Geometric Adaptation for a fitted mesh

1: Apply anisotropic adaptation on the level-set.
2: Flag the cut elements where $\phi(x_i)\phi(x_j) < 0$
3: **for** each flagged element **do**
4: Move the vertex with the minimum distance
5: Detect two adjacent elements with each element having one fitted vertex on the interface in order to define a pair
6: **for** each couple **do**
7: Apply Edge Swapping keeping in mind the orientation of the elements

- **Adapting**

After generating an isotropic mesh for the immersed geometry, an anisotropic mesh adaptation is applied to the level-set using the metric map described in equation 3.27. On a narrow band region in the vicinity of the interface, extremely anisotropic elements stretched along the boundary are formed. The anisotropic refinement allows us to follow the geometric shape of the immersed geometry hence describing its curvatures, angles, direction, etc. (Figures 4.9).

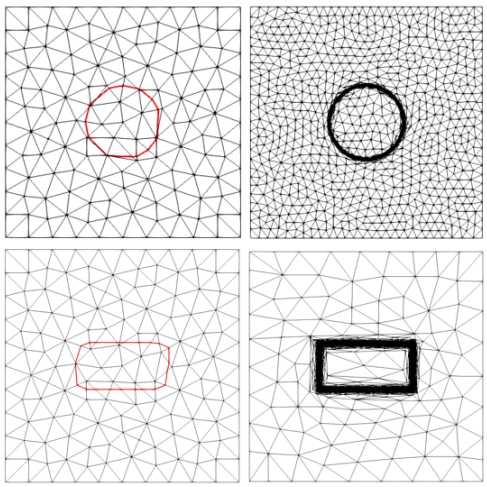


Figure 4.9: Anisotropic adaptation for 2D geometries.

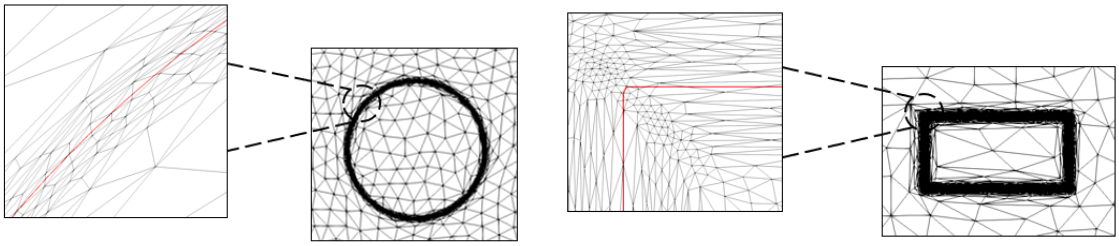


Figure 4.10: Zoom on the interfaces of a cylinder and a rectangle highlighting the cut elements.

- **Flagging**

Since the immersed geometry is described using the level-set method, the interface Γ separates the domain Ω into two subdomains: a negative one Ω_f and a positive one Ω_s [14]. Therefore, the interface elements or the *cut elements* are detected by looping over their edges. The product of the signed distance value computed at the

coordinates of each vertex (v_i, v_j) of a cut edge results in a negative value allowing the detection of the cut elements (Algorithm 3 and figure 4.12). A loop over all the level-set values is performed, the cells that are fully contained in Ω_f or in Ω_s are flagged with 0, and all the cells that are cut by the zero-value of the level-set are marked as 1 (Figure 4.11).

Algorithm 3 Flagging

1: **for** each edge (e_{ij}) **do**
2: Compute $\phi(x_i)$ and $\phi(x_j)$
3: **if** $\phi(x_i)\phi(x_j) < 0$ **then**
4: flag $= 1$
5: **else**
6: flag $= 0$
7: flag(v_i), $i \in \{1, \ldots, N\}$ N: # of vertices

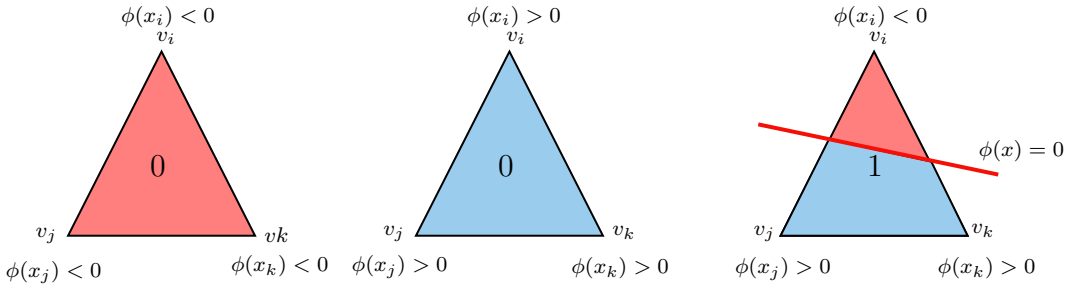


Figure 4.11: Illustration of Algorithm 3

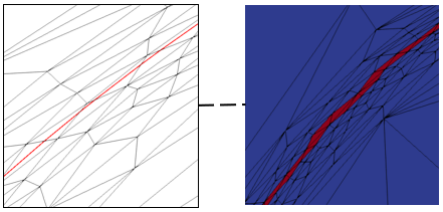


Figure 4.12: The flagged elements cut by the interface.

- **Inserting & Fitting**

▷ **R-adaptation**

R-adaption is based on relocating the vertices by maintaining the same number

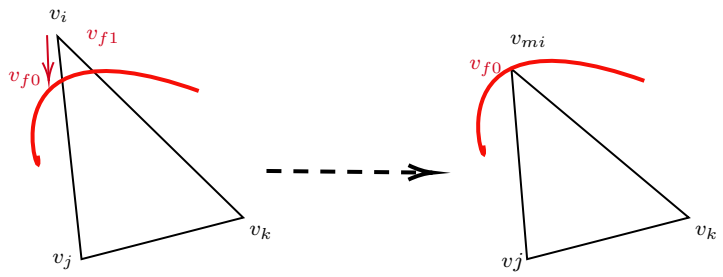


Figure 4.13: R-adaptation operation for a 2D cut element applied on vertex v_i.

of degrees of freedom and element connectivity. It's only applied on the flagged elements as a consequence, the computational cost remains low.

The general strategy consists of moving specific vertices of the mesh. For each cut element, the distance from each vertex to the interface is analyzed via the level-set function: the vertex having the minimum distance v_{m_i} is then marked. Since the interface cuts the facets of the mesh: the intersection points are then defined as virtual vertices v_f such that $f = 0, 1, ...n_f$ with n_f the number of cut facets.

The objective is to move the marked vertex v_{m_i} along its associated facet f with minimum effort. Since v_{m_i} can be associated to multiple virtual vertices, a set of associated virtual vertices is created and analyzed for each v_{m_i}. From this set, v_f is chosen as the virtual vertex having the smallest distance $\mathbf{d_{v_{m_i} v_f}}$, then the coordinates x_{m_i} of v_{m_i} are updated as follows:

$$x_{m_i} = x_{v_f} \tag{4.8}$$

with x_{v_f} the coordinates of the chosen virtual vertex associated to v_{m_f}.

Algorithm 4 R-adaptation

1: **for** the vertices of a cut element **do**
2: Mark the vertex having minimum distance
3: **for** each marked vertex **do**
4: Determine the needed virtual vertex v_f of the associated facet f and the interface
5: $x_{inew} = x_{v_f}$
6: $x(i), i \in \{1, \ldots, n\}$ n: # of marked vertices in cut elements

To prevent mesh degeneration or inverted elements, care must be taken. Hence, constraints on the number of marked vertices are applied: each cut element should have at least one vertex marked and no element has all of its vertices marked to

avoid generating a flat element. And each v_f can be associated to only one real marked vertex v_{m_0}.

▷ **Edge Swapping**

The R-adaptation algorithm allows the movement of the vertices' cut elements to capture the interface. To ensure a fitted mesh, algorithm 4 is coupled with a local mesh optimization algorithm. This algorithm is based on a simple topology change method: Edge swapping. Edge swapping not only avoids expensive remeshing but preserves the quality and integrity of the mesh. This operation conserves the number of elements and the number of edges. In 2D, this local optimization is a simple topological operation consisting of swapping or flipping a common edge shared by two elements (forming a pair or a couple) (Figure 4.14).

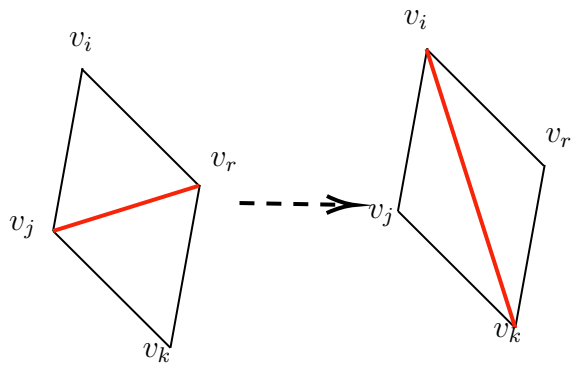


Figure 4.14: Edge swapping operation in 2D.

Since edge-swapping alters the topology of the mesh, a careful evaluation of the global connectivity of the mesh needs to be done to ensure that the orientation of the elements is preserved [22]. Therefore, the edge swapping algorithm can be decomposed into three main parts:

1. Pairing the elements
2. Swapping
3. Fixing the orientation

The geometric adaptation can be used to solve multi-component systems with complex geometries, CFD problems and Fluid-Solid Interaction applications.

4.5 Numerical Illustrations

The geometric adaptation algorithm is applied first on a set of geometries to show the accurate and precise capture of the interface and then benchmark problems are

Algorithm 5 Edge swapping

```
Loop over the cut elements
Flag the elements with only one vertex on the interface
Loop over the newly flagged elements
if two elements have a common edge then
    Form a couple                                   ▷ Pairing the elements
for each couple do
    Determine the common edge
    Swap the edge                                   ▷ Edge-Swapping
    for each element do
        Calculate the signed volume 𝒱
        while 𝒱 < 0 do
            Reorder the vertices indiex             ▷ Fixing the orientation
```

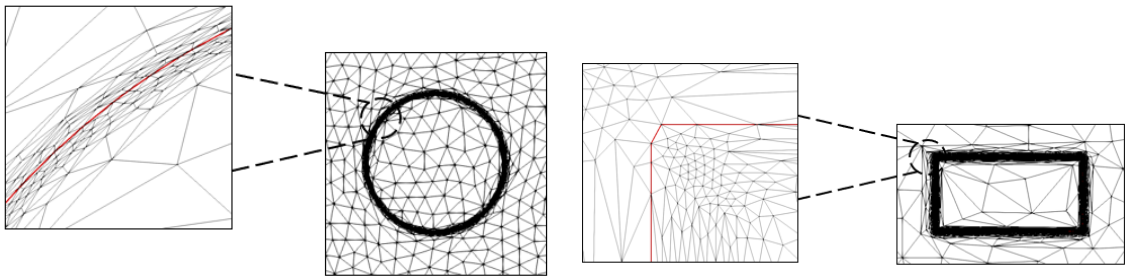


Figure 4.15: Zoom on the fitted interface for a cylinder and a rectangle.

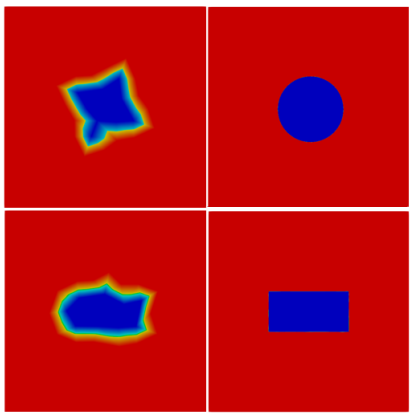


Figure 4.16: Comparison between the initial and final fitted geometry.

explored. The objective of exploring these cases is to understand the impact and importance of an immersed fitted interface.

4.5.1 Anisotropic Fitted Mesh for Immersed 2D Objects

In order to confirm the results, the global adaptive framework was implemented successfully on simple geometries (Figures 4.9 - 4.16) as well as the geometry of a rabbit (Figures 4.17-4.19) with different curvatures. First, an anisotropic adaptation, implemented on the level-set function of the immersed geometry, is applied. The use of anisotropic elements is an essential component to accurately trace the different changes in the geometry due to the flexibility of such elements (Figure 4.17).

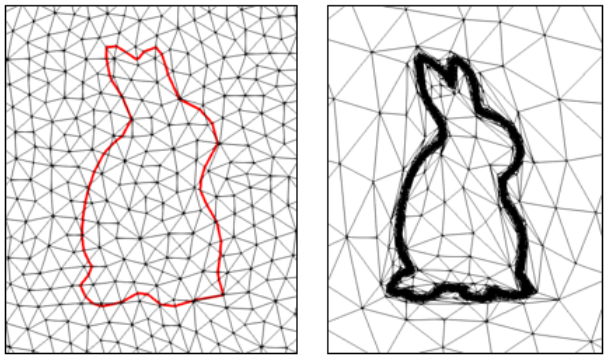


Figure 4.17: Anisotropic adaptation on the geometry of a 2D rabbit.

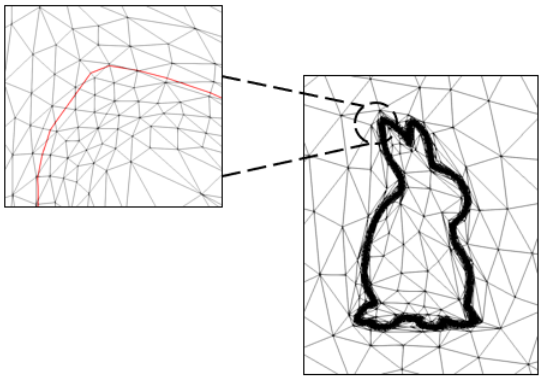


Figure 4.18: Zoom on the interface highlighting the cut elements

Then the immersed-fitting adaptation was applied to the elements in the vicinity of the level-set, represented in red in Figures 4.18 and 4.19. The R-adaptation relocates the marked vertices to get them closer to the interface, and then edge-swapping is applied to ensure a body-fitted mesh.

Figure 4.20 shows the flexibility and versatility of the fitting algorithm. Moreover, the flow over four circular cylinders at various Reynolds numbers (Figure 4.21)

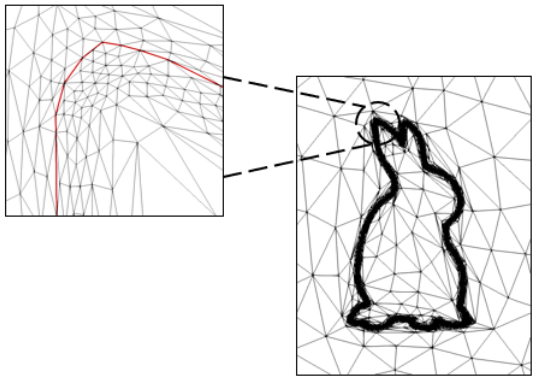


Figure 4.19: Zoom on the fitted interface.

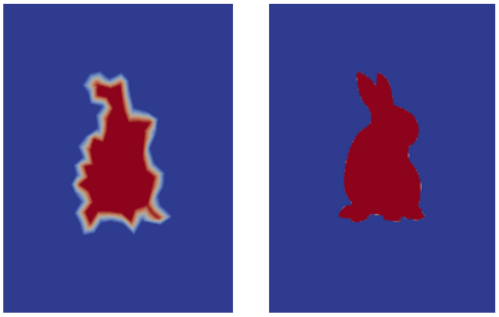


Figure 4.20: Comparison between the initial and the final fitted mesh of a 2D rabbit.

was studied. Using the method presented, an anisotropic fitted mesh was successfully created on all four cylinders independent on the flow's turbulence.

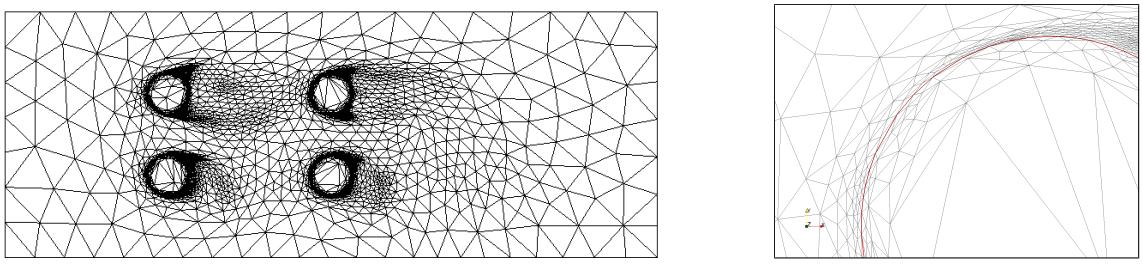


Figure 4.21: Immersed Fitted mesh for 4 circular cylinders and zoom on the interface.

4.5.2 Flow over a Circular Cylinder

We consider the flow over a circular cylinder, also studied by Main et al. in [10], at Reynolds number $Re = 100$ over which the Navier-Stokes equations for incompress-

ible flow are solved using the Variational Multiscale Method.

The velocity inlet of the flow is set to $U = 1$ and the top and bottom of the domain are set to be traction free. On the outlet, a free Newman boundary condition is imposed. The Drag coefficient is calculated using the following equation:

$$C_D = \frac{2F_d}{D\rho U^2} \tag{4.9}$$

where F_d is the drag force.

Table 4.1 compares the drag coefficient obtained for three types of meshes:

1. a non fitted isotropic initial mesh
2. a non fitted anisotropic mesh
3. an immersed-fitted mesh.

The immersed-fitted mesh was obtained by implementing the algorithm defined in Section 3.3.1 by applying an anisotropic adaptation near the interface and using the Immersed Method described in section 2.

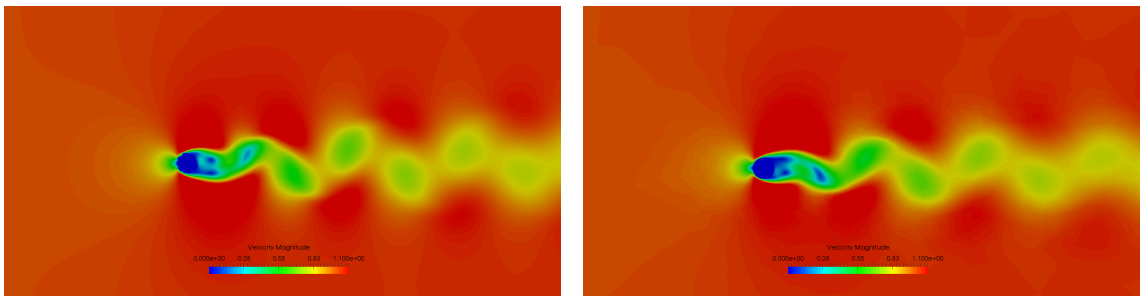


Figure 4.22: Solution for flow past a cylinder for $Re = 100$ using an isotropic mesh $(left)$ and an immersed-fitted anisotropic mesh $(right)$ at $t = 100s$.

Table 4.1: Drag Coefficient C_D computed for the 3 cases for Re=100 with N being the number of elements used.

	N	C_D	% Error
Reference	352 590	1.34	-
Isotropic mesh	156 163	1.413	5.45%
Anisotropic mesh	10 000	1.316	1.77%
Immersed-Fitted mesh	10 000	1.334	0.44%

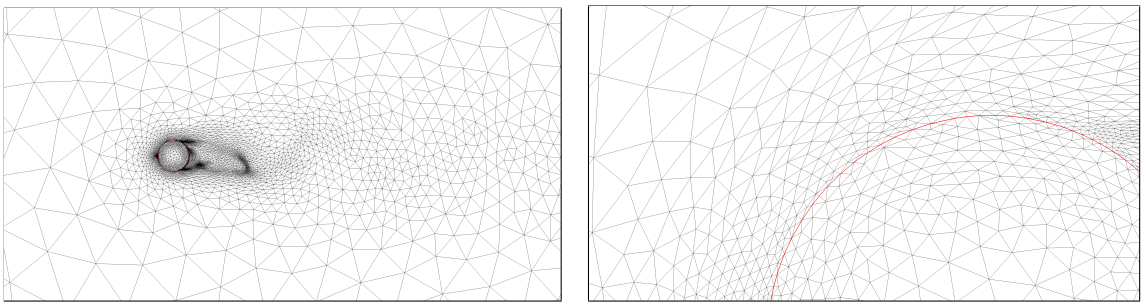


Figure 4.23: Anisotropic mesh adaptation for both the velocity and the level-set fields (left), zoom on the sharp immersed-fitted interface (right).

The results obtained (table 4.1 and plot 4.24) confirm that using the Immersed Fitted mesh adaptation yields to better results with a minimum error in C_D equal to 0.44%. Figure 4.24 shows a time history of the drag coefficient for $Re = 100$, for the three cases studied.

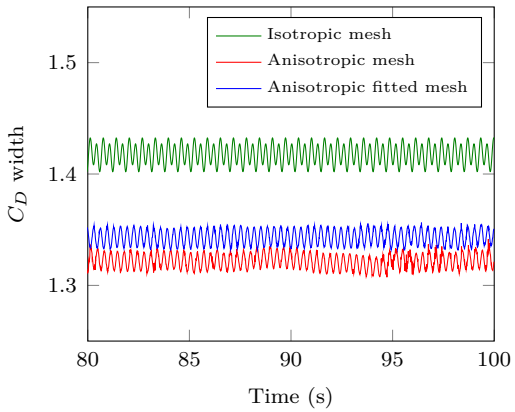


Figure 4.24: Time evolution of the drag coefficient for three different meshes.

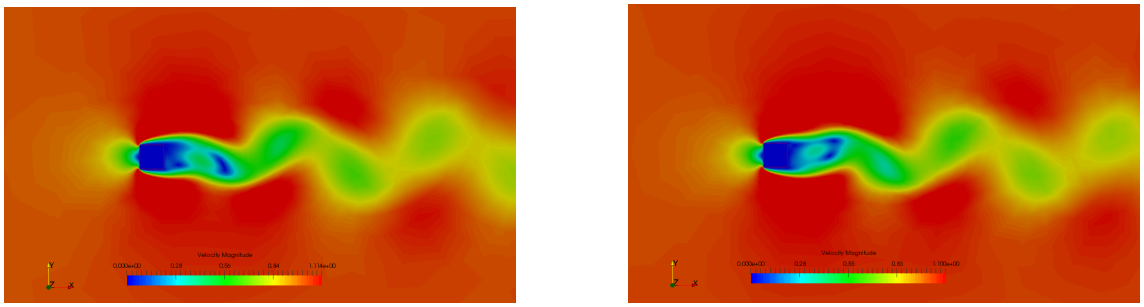
It is important to note from table 4.1 that the anisotropic adaptation allows one to significantly reduce the number of elements used in order to solve the Navier-Stokes equations, compared to the use of isotropic mesh: only 10 000 elements were needed compared to 156 163 elements in the case of isotropic refinement and more than 350 000 elements in [10]. Also, the computational time to calculate the results using the anisotropic or the immersed-fitted mesh is significantly lower than the time needed to solve the problem using an isotropic mesh.

4.5.3 Flow over a Square Cylinder

We consider next a flow over a square cylinder for Re=100 presented in [23]. The computational domain consists of a square cylinder placed normal to a free stream in an infinite domain. A uniform velocity is set on the inlet $U = 1$ and no slip conditions are applied to the boundary of the immersed square.
Such a geometry was tested because its corners and sharp angles can be a challenge. The algorithm of section 3.3.1 was successfully applied to the geometry without any modifications and was able to capture the sharp edges. Table 4.2 compares the drag coefficient obtained for the three types of meshes: the Immersed fitted mesh shows the best results.

Table 4.2: Drag Coefficient C_D computed for the 3 cases for a square cylinder at Re=100.

	C_D	% Error
Reference [23]	1.461	-
Isotropic mesh	1.524	4.312%
Anisotropic mesh	1.429	2.173%
Immersed-Fitted mesh	1.433	1.926%



(a) $t = 25s$ (b) $t = 100s$

Figure 4.25: Solution for flow past a square cylinder for $Re = 100$ for an immersed-fitted anisotropic mesh (bottom) at $t = 25s$ an $t = 100s$.

4.5.4 Complex Geometry with High Reynolds Number Flow

To go further, we consider the flow over a complex geometry, a 2D representation of a F1 car. The objective of this example is to show the performance of immersed-fitted meshing. Indeed, combined with flow solvers it allows to easily and accurately deal with complex fluid-structure interaction problems. Taking a closer look at the mesh

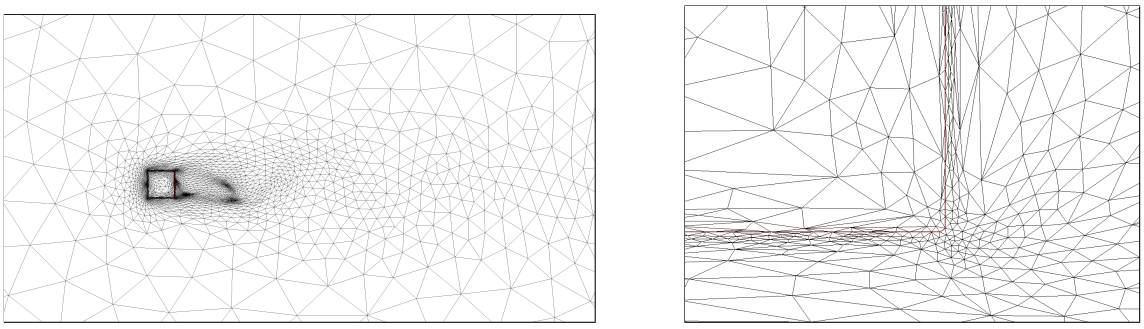


Figure 4.26: Anisotropic mesh adaptation for both the velocity and the level-set fields (left), zoom on the sharp immersed-fitted interface (right).

near the interfaces, we can detect the good orientation of the anisotropic elements as well as the accurate precision with the conformal interface. The velocity field and the mesh obtained are shown in Figures 4.27 and 4.28. Taking a closer look at the mesh in Figure 4.29, we can see how the interface not only follows the curvatures of the immersed geometry but also captures accurately its interface.

4.6 Advantage of the Proposed Method

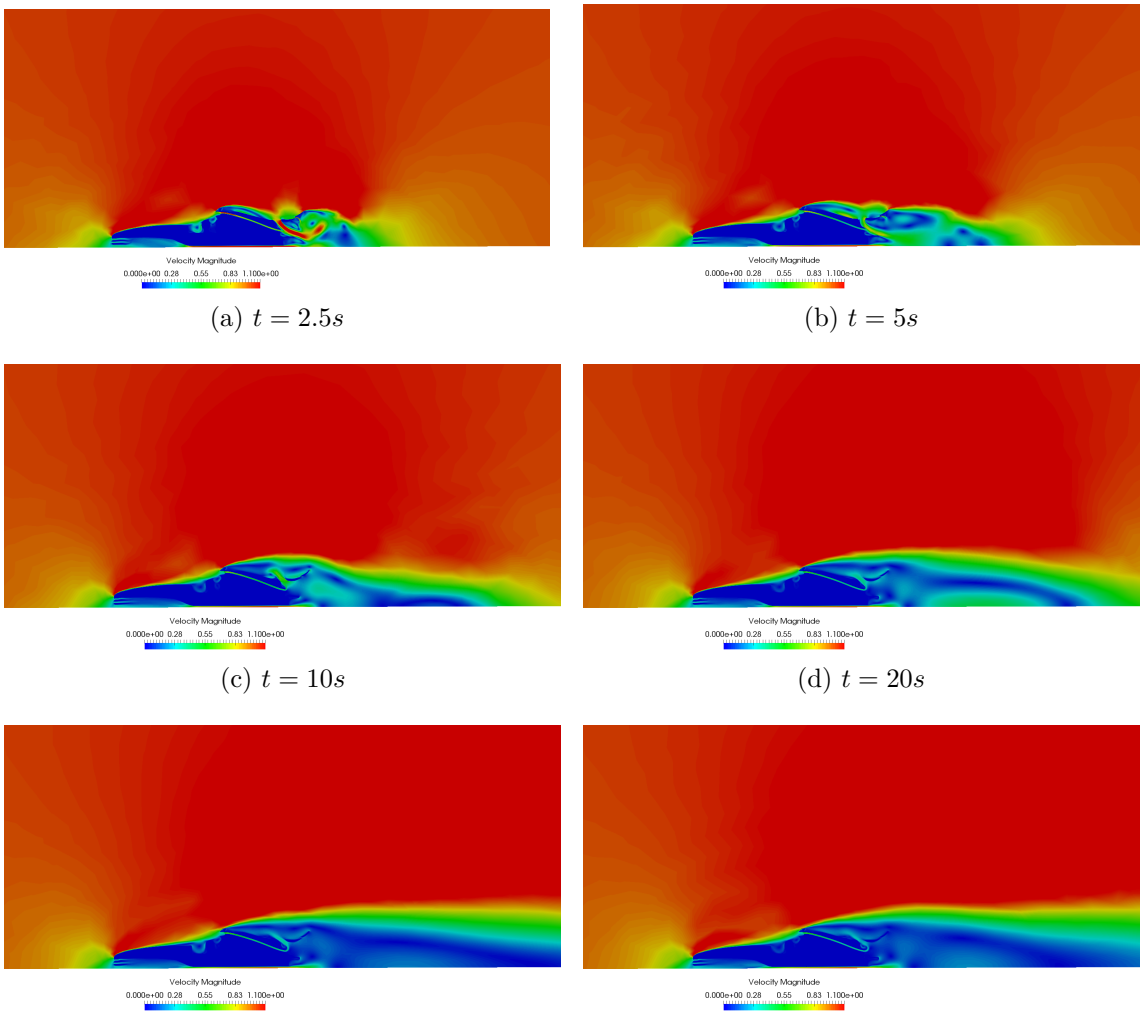
Since CFD and fluid flow problems use complex geometries, high resolution CAD models (or STL) require to be treated. The use of immersed geometries and how they are expressed on the interface depends on the end use and the level of geometric information needed at the interface. Many accurate interface description using high-order polynomial representations [24] or level-set functions [25, 26] have been used. The mathematical structure of the governing equations being solved frequently influences the augmentation of discrete operators employing immersed interfaces, which are divided into incompressible [27] or compressible formulations [28, 29]. Some methods, like the ghost fluid method, described in section 4.2.2.1, often requires one or more of the following steps:

1. inserting local vertices at the interface

2. increasing the size of the discrete stencils near the interface

3. adding vertices in the solid region in the vicinity of the interface to enforce local boundary conditions.

Other methods rely on transforming the background mesh to conform to the boundary by using a closest point projection to parameterize the immersed boundary over a collection of nearby edges [30], or on augmenting the level-set function via



(a) $t = 2.5s$ (b) $t = 5s$

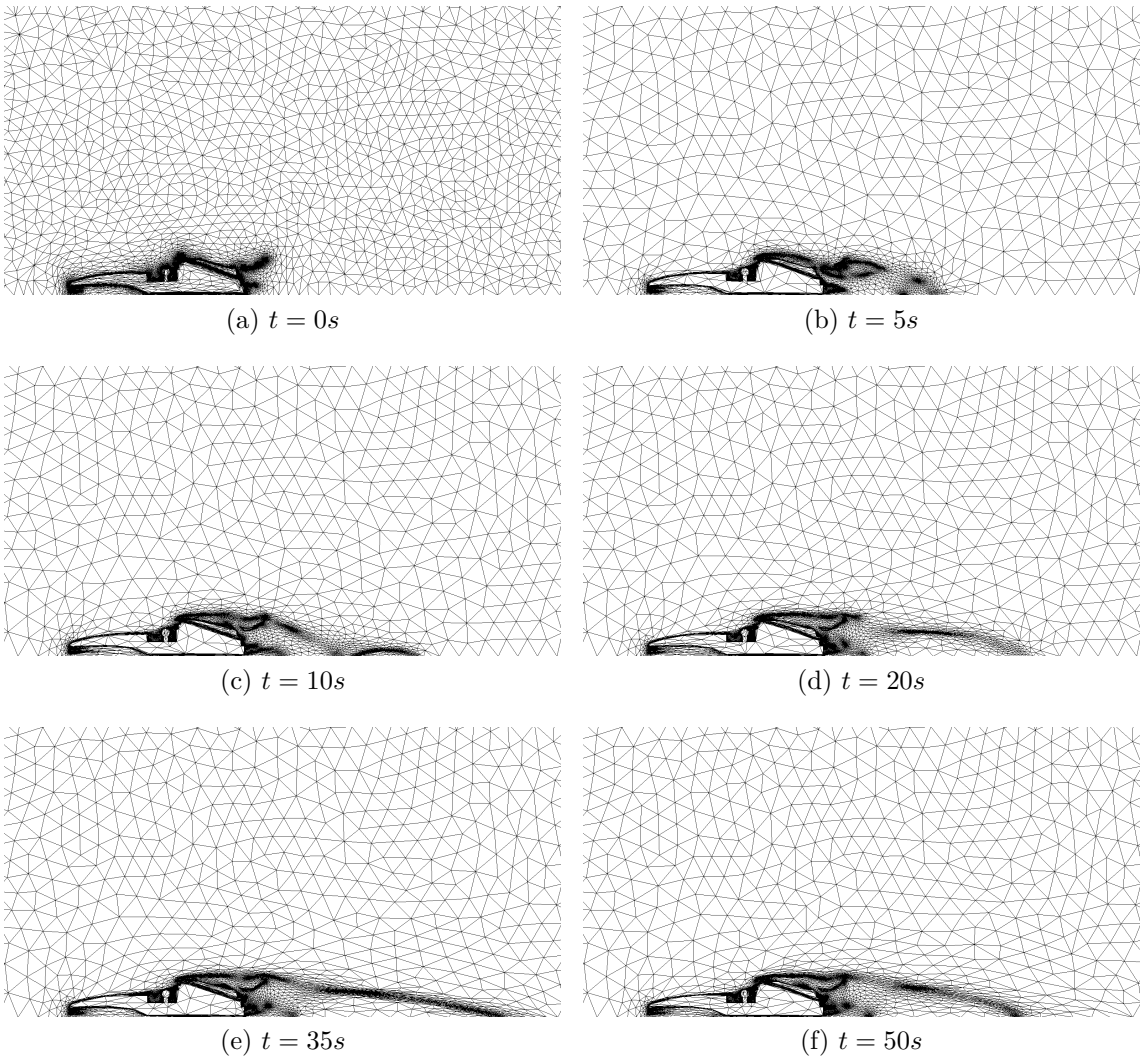
(c) $t = 10s$ (d) $t = 20s$

(e) $t = 35s$ (f) $t = 50s$

Figure 4.27: Velocity profile past an immersed 2D car.

an iterative procedure to then reconstruct the cut cell elements using Lagrangian polynomials [31].

Since the first step of the proposed method is to apply an anisotropic mesh adaptation, the mesh is locally refined according to the level-set function that describes the geometry without resorting to the reconstruction or correction of the interface from a CAD model. The anisotropic mesh adaptation gives the flexibility to adapt to the immersed geometry and to construct the stretched elements in the direction of the interface allowing to smoothly track the curve of the geometry. Unlike isotropic elements, the smoothness of the immersed interface and its detection can be done



(a) $t = 0s$ (b) $t = 5s$

(c) $t = 10s$ (d) $t = 20s$

(e) $t = 35s$ (f) $t = 50s$

Figure 4.28: Anisotropic mesh adaptation for both the velocity and the level-set of the immersed 2D car fields.

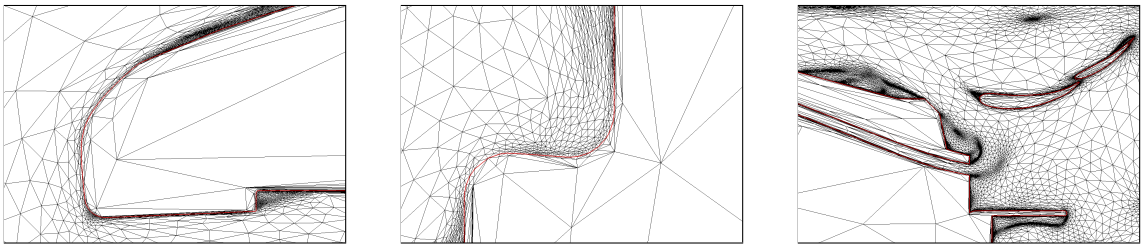


Figure 4.29: Zoom on the sharp immersed-fitted interface of the 2D car.

with a relatively low number of elements as can be seen in Table 4.1.

The metric map allows an adaptation on the level-set function as well as the solution of the problem as mentioned in section 3.3.5, the boundary layer is then well constructed taking onto account the flow physics. The Dirichlet boundary conditions can then be strongly imposed on the interface just like classical body-fitted meshes.

4.7 Conclusion

After a brief review and description of the existing method that treat multi-component systems and immersed geometries, in this chapter, the proposed anisotropic fitted algorithm is presented. It's an ordered two-step implementation allowing the creation of a fitted mesh while benefiting from the advantages of an anisotropic one, hence flow phenomena can be treated more accurately especially near the interface and zones with high gradients and discontinuity. First, the metric based anisotropic mesh adaptation is applied on a coarse initial mesh. This refinement allows one to detect the immersed object via the level-set function and hence better describes its geometric shape. Then a geometric mesh adaptation is applied to the flagged and isolated cut cell to create a conformal mesh on the interface of the immersed geometry with a series of topological adaptation: R-adaptation followed by swapping. Numerical illustrations in 2D have been presented.

However, since the objective of this work is to be applied to more real and complex problems, especially to model the quenching process, the proposed algorithm can be extended to 3D. This extension, of course, comes with some challenges. Furthermore, generating and refining a mesh adapted to three-dimensional complex industrial problems, and then solving the partial differential equations at hand become very expensive. In the next chapter, the new implementation of the parallel mesh adaptation used in our code CIMLIB-CFD is described and the proposed algorithm in parallel and 3D is explained in detail.

Bibliography

[1] E. Hachem, H. Digonnet, E. Massoni, T. Coupez, Immersed volume method for solving natural convection, conduction and radiation of a hat-shaped disk inside a 3d enclosure, International Journal of Numerical Methods for Heat & Fluid Flow (2012). 52, 53, 58, 60

[2] E. Hachem, S. Feghali, R. Codina, T. Coupez, Immersed stress method for fluid–structure interaction using anisotropic mesh adaptation, International Journal for Numerical Methods in Engineering 94 (9) (2013) 805–825. 52, 53, 56

[3] C. S. Peskin, Numerical analysis of blood flow in the heart, Journal of Computational Physics 25 (3) (1977) 220–252. 52, 53, 56

[4] C. S. Peskin, The immersed boundary method, Acta numerica 11 (2002) 479–517. 56

[5] D. Boffi, L. Gastaldi, A finite element approach for the immersed boundary method, Computers & Structures 81 (8-11) (2003) 491–501. 52, 53

[6] N. Moës, J. Dolbow, T. Belytschko, A finite element method for crack growth without remeshing, International Journal for Numerical Methods in Engineering 46 (1) (1999) 131–150. 53, 55

[7] J. Dolbow, N. Moës, T. Belytschko, An extended finite element method for modeling crack growth with frictional contact, Computer Methods in Applied Mechanics and Engineering 190 (51-52) (2001) 6825–6846.

[8] T. Belytschko, C. Parimi, N. Moës, N. Sukumar, S. Usui, Structured extended finite element methods for solids defined by implicit surfaces, International Journal for Numerical Methods in Engineering 56 (4) (2003) 609–635.

[9] A. Main, G. Scovazzi, The shifted boundary method for embedded domain computations. part i: Poisson and stokes problems, Journal of Computational Physics 372 (2018) 972–995. 57, 58

[10] A. Main, G. Scovazzi, The shifted boundary method for embedded domain computations. part ii: Linear advection–diffusion and incompressible navier–stokes equations, Journal of Computational Physics 372 (2018) 996–1026. 53, 57, 69, 71

[11] Y.-H. Tseng, J. H. Ferziger, A ghost-cell immersed boundary method for flow in complex geometry, Journal of computational Physics 192 (2) (2003) 593–623. 53, 56, 57

[12] E. Burman, Ghost penalty, Comptes Rendus Mathematique 348 (21-22) (2010) 1217–1220.

[13] H. J. Barbosa, T. J. Hughes, The finite element method with lagrange multipliers on the boundary: circumventing the babuška-brezzi condition, Computer Methods in Applied Mechanics and Engineering 85 (1) (1991) 109–128.

[14] A. Legay, J. Chessa, T. Belytschko, An eulerian–lagrangian method for fluid–structure interaction based on level sets, Computer Methods in Applied Mechanics and Engineering 195 (17-18) (2006) 2070–2087. 63

[15] J. Dolbow, I. Harari, An efficient finite element method for embedded interface problems, International Journal for Numerical Methods in Engineering 78 (2) (2009) 229–252.

[16] G. Vigueras, F. Sket, C. Samaniego, L. Wu, L. Noels, D. Tjahjanto, E. Casoni, G. Houzeaux, A. Makradi, J. M. Molina-Aldareguia, et al., An xfem/czm implementation for massively parallel simulations of composites fracture, Composite Structures 125 (2015) 542–557. 53

[17] D. L. Quan, A mesh-based technique for solving cfd problems with embedded geometries: Anisotropic adaptive" nearly" body-fitted mesh approach, Ph.D. thesis, PhD thesis, Université catholique de Louvain (2014). vi, 55, 56, 57, 58, 59, 60

[18] N. Moës, T. Belytschko, Extended finite element method for cohesive crack growth, Engineering Fracture Mechanics 69 (7) (2002) 813–833. 55

[19] E. Hachem, Stabilized finite element method for heat transfer and turbulent flows inside industrial furnaces, Ph.D. thesis (2009). 58

[20] T. Coupez, E. Hachem, Solution of high-reynolds incompressible flow with stabilized finite element and adaptive anisotropic meshing, Computer Methods in Applied Mechanics and Engineering 267 (2013) 65–85. 60

[21] F. Alauzet, Efficient moving mesh technique using generalized swapping, in: Proceedings of the 21st International Meshing Roundtable, Springer, 2013, pp. 17–37. 62

[22] P. L. George, H. Borouchaki, F. Alauzet, P. Laug, A. Loseille, L. Marechal, Meshing, Geometric Modeling and Numerical Simulation, Volume 2: Metrics, Meshes and Mesh Adaptation, John Wiley & Sons, 2019. 66

[23] B. Gera, P. K. Sharma, Cfd analysis of 2d unsteady flow around a square cylinder, International Journal of Applied Engineering Research Dindigul (2010). 72

[24] H. Udaykumar, R. Mittal, P. Rampunggoon, A. Khanna, A sharp interface cartesian grid method for simulating flows with complex moving boundaries, Journal of Computational Physics 174 (1) (2001) 345–380. 73

[25] S. Osher, R. Fedkiw, K. Piechor, Level set methods and dynamic implicit surfaces, Applied Mechanics Reviews 57 (3) (2004) B15–B15. 73

[26] J. A. Sethian, Level set methods and fast marching methods: evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science, Vol. 3, Cambridge University Press, 1999. 73

[27] D. Lacanette, S. Vincent, A. Sarthou, P. Malaurent, J.-P. Caltagirone, An eulerian/lagrangian method for the numerical simulation of incompressible convection flows interacting with complex obstacles: Application to the natural convection in the lascaux cave, International Journal of Heat and Mass Transfer 52 (11-12) (2009) 2528–2542. 73

[28] C. Farhat, A. Rallu, K. Wang, T. Belytschko, Robust and provably second-order explicit–explicit and implicit–explicit staggered time-integrators for highly nonlinear compressible fluid–structure interaction problems, International Journal for Numerical Methods in Engineering 84 (1) (2010) 73–107. 73

[29] C. Farhat, A. Rallu, S. Shankaran, A higher-order generalized ghost fluid method for the poor for the three-dimensional two-phase flow computation of underwater implosions, Journal of Computational Physics 227 (16) (2008) 7674–7700. 73

[30] R. Rangarajan, A. J. Lew, Universal meshes: A method for triangulating planar curved domains immersed in nonconforming meshes, International Journal for Numerical Methods in Engineering 98 (4) (2014) 236–264. 73

[31] P. E. DesJardin, B. T. Bojko, M. T. McGurn, Initialization of high-order accuracy immersed interface cfd solvers using complex cad geometry, International Journal for Numerical Methods in Engineering 109 (4) (2017) 487–513. 74

Chapter 5

Numerical and Parallel Modeling of the Anisotropic Adaptive Fitted Mesh

Contents

Résumé en Français

Comme les phénomènes d'écoulement et les problèmes de CFD visent à résoudre des problèmes complexes et réels, ils doivent s'appuyer sur des systèmes de calcul haute performance (HPC). Dans le contexte des applications industrielles, le temps de calcul devient un problème. L'utilisation de la programmation parallèle dans le calcul scientifique peut réduire considérablement le temps de calcul par rapport au temps séquentiel nécessaire pour réaliser la même tâche. Par conséquent, les problèmes complexes et à grande échelle traitant de systèmes multiphases et multi-composants deviennent plus abordables à traiter. Par exemple, le nombre de degrés de liberté peut être augmenté pour obtenir une solution plus précise sans causer une augmentation drastique du temps de calcul.
Cependant, pour bénéficier des avantages du calcul parallèle, il est nécessaire de prendre certaines précautions lors de la parallélisation de cet algorithme. L'efficacité de la programmation parallèle dépend fortement d'aspects logiciels tels que l'équilibrage des charges et l'architecture de l'ordinateur. Par conséquent, l'étape finale de cette nouvelle mise en œuvre est la parallélisation de l'algorithme.
Ce chapitre donne un aperçu rapide du parallélisme et se concentre sur l'algorithme parallèle d'adaptation du maillage utilisé dans CIMLIB-CFD. La modélisation parallèle de l'algorithme proposé est également expliquée en détail en mettant l'accent sur les défis rencontrés et son extension aux applications tridimensionnelles.

5.1 Introduction

Since flow phenomena and CFD problems aim to solve more complex and real problems, they must rely on high performance computing (HPC) systems. In the context of industrial applications, computational time becomes an issue. Using parallel programming in scientific computing can significantly reduce the computational time compared to the sequential time needed to achieve the same task. Therefore, large scale and complex problems dealing with multi-phase and multi-component systems become more affordable to handle. For example, the number of degrees of freedom can be increased to obtain a more accurate solution without a drastic increase of the computing time.

However, to benefit from the advantages of parallelism, the algorithm needs to be treated with care. The efficiency of the parallel programming strongly depends on software aspects such as load balancing and computer architecture. Therefore, the final step of this new implementation is the parallel modeling of the algorithm.

This chapter provides a quick overview of parallelism and focuses on the parallel mesh adaptation algorithm used in CIMLIB-CFD. The proposed algorithm's parallel modeling is also explained in details focusing on the challenges faced and its extension to three-dimensional applications.

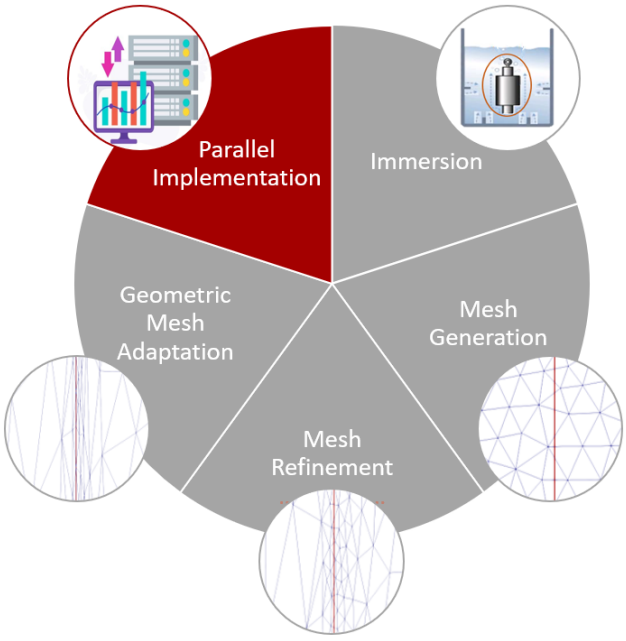


Figure 5.1: Parallel implementation framework.

5.2 Parallel Terminology & Memory Classification

As already mentioned, to have an efficient parallel algorithm, one has to know and understand the computer architecture and software. For computer architectures, several classifications exist. In this section, Flynn's taxonomy will be presented followed by the most common memory classifications.

5.2.1 Flynn's Taxonomy

Flynn's classification scheme [1] is a classification of the existing computing systems. It depends on the notions of instruction and data. Hence, based on the number of instruction and data streams that can be processed simultaneously, four major categories can be found:

1. **Single-Instruction, Single-Data (SISD) systems:**
   this system is a single-processor machine capable of executing one instruction and operating on one single data stream. All the instructions and data are stored in a primary memory. It is based on a sequential processor. The speed of processing is dependent on the rate at which the computer can transfer and communicate the information internally.

2. **Single-Instruction, Multiple-Data (SIMD) systems:**
   this system uses a multi-processor machine to execute one single instruction but operates on several data streams. Each processing unit obtains the same information, but loads a separate and different data set. Machines based on a SIMD model are well suited to scientific computing.

3. **Multiple-Instruction, Single-Data (MISD) systems:**
   this system uses a multi-processor machine to execute several instructions but accesses the same data stream. Even though, it can perform many operations on the same data set, it remains restrictive and tricky to operate hence, it isn't common to find machines built using this model.

4. **Multiple-Instruction, Multiple-Data (MIMD) systems:**
   this system uses a multi-processor machine to execute several instructions on multiple data sets. Each processor executes a different instruction to a different data memory and stores the results into the corresponding storage. The MIMD model is capable for any type of applications and is mostly used in multi-processors systems and clusters.

5.2.2 Memory Classification

As for memory access and classification, two major types exist:

1. **Shared memory system:**
   In this system, the memory data is stored in one single storage (global memory) to which all the processors are connected and can access all the data. Hence, the communication between the different processors takes place through the shared memory. Indeed, a modification of the data by one processor is visible and seen to all others. Parallel computing using this model is generally associated with threads (OpenMP).

2. **Distributed memory system:**
   This type of system refers to a multiprocessor computer model in which each processor has its own private memory. This pairing (processor-memory) is called a node. The interconnection between all this private memory processors is accomplished via a network. In term of parallel computing, a distributed memory machine is associated with a process using a message passing interface (MPI) to communicate between nodes.

Even-though the shared memory model seem easier to program since a minimum communication between the processors needs to be done, this system is less tolerant to failure, where one failure can affect the entire system contrary to distributed memory systems. However, the choice between these two systems is also dependent on the equipment used.

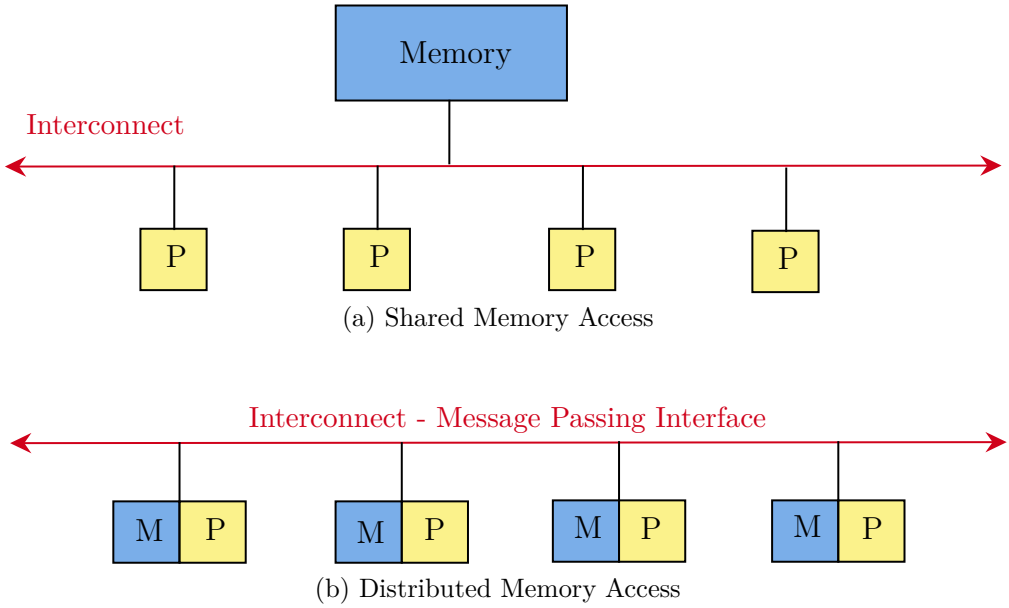
In our work, distributed memory MIMD system with Message Passing Interface (MPI) is used for parallel computing and implementation.

5.2.3 Load Balancing

In scientific computing, load balancing is the process of distributing the workload as equally as possible to all the resources, i.e. the processors in a parallel environment. There exists two type load balancing algorithms: static and dynamic ones.

Static load balancing algorithms have a prior knowledge about the existing servers of the distributed network, and divide the incoming load in the beginning evenly on all participating processors. During the simulation, the allocated load cannot be transferred to other processors. This can affect the scalability since the current state isn't taken into account. However, with dynamic load balancing algorithms the load is identified and evaluated during run-time, and distributed to the processors involved accordingly.

When dealing with finite element problems that require mesh generation and refinement during the computation, as seen in Chapter 3, there is a local change in the number of vertices, and elements involved in the mesh, so load imbalance is bound to happen. A dynamic reallocation of the vertices can overcome this issue by transferring some of the workload from a processor to another. An overview and more details about dynamic load balancing can be found in [2, 3].



(a) Shared Memory Access

(b) Distributed Memory Access

Figure 5.2: The memory classification of parallel computers (M stands for Memory and P for processor)

5.3 Parallel Implementation of the Anisotropic Fitted Adaptation

In this section, the parallel implementation of the proposed anisotropic fitted adaptation will be explained in details.

5.3.1 Parallel Mesh Adaptation

Our library Cimlib-CFD is based on Message Passing Interface (MPI) for efficient and parallel implementation and communication between processors (procs). Details of the parallel mesh adaptation algorithm used in the library can be found in [4–6].

At the beginning of the simulation, the initial mesh $\mathcal{M} = (\mathcal{N}, T)$, with $\mathcal{N}$ and T the set of vertices and simplices forming the mesh, respectively, is partitioned over the allocated resources p into multiple subdomains creating sub-meshes $\mathcal{M}_i = (\mathcal{N}_i, T_i)$ such as:

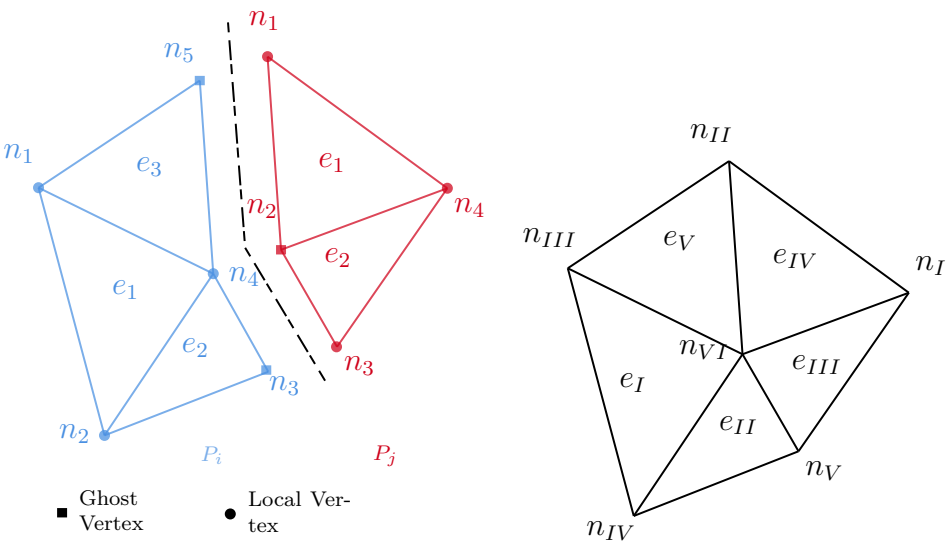
$$\mathcal{M} = \bigcup_{i=0}^{card(p)} \mathcal{M}_i \tag{5.1}$$

An initial load balancing is done based on the mesh topology and the geometry of the domain. Each sub-mesh is associated with a particular processor p_i. Note that a simplex can only belong to one sub-domain: it can be owned by only one

processor. However, a vertex or a set of vertices can be shared between processors: for instance, a shared vertex v_i between two processors p_i and p_j can be owned by p_i, and also be a ghost vertex of p_j. The shared vertices form an interface between the different processors referred to as inter-process interface:

$$\Gamma_{ij} = \left\{ \mathcal{M}_i \cap \mathcal{M}_j | p_i \neq p_j \right\} \tag{5.2}$$

Figure 5.3 illustrates the partitioning and distribution on each processor, showing the local and ghost vertices. Once the partition is created, each p_i reads the data related to its associated sub-mesh, and has a local numbering of the vertices as well as simplices.



(a) Local numbering per processor (b) Global numbering

Figure 5.3: Illustration of mesh partitioning

Following the initial partitioning, on each $\mathcal{M}_i$, the different operations such as the resolution of the Navier-Stokes or CDR equations (see Chapter 2) are done. In case of anisotropic mesh adaptation or refinement, the local error is computed using the method described in Chapter 3.

The mesh is then adapted with respect to the solution field. For each sub-domain, a local error estimate is computed and the individual sub-meshes are adapted or refined to the error estimator using a sequential mesh adaptor. In order to keep the global mesh correct, the interfaces between the sub-meshes should also be adapted. To do so, the interfaces between processors are first blocked and kept unchanged during local remeshing inside each sub-domain, hence avoiding communication between the adjacent processors. To obtain a final adapted mesh, a repartitioning step is performed to move the interface inside a sub-domain to enable remeshing in the

next phase. A few iterations of remeshing and repartitioning are necessary in order to remesh the entire domain and build the optimal mesh. Note that the time spent doing so per iteration decreases drastically as there are less and less elements to move and remesh [6].

5.3.2 Parallel implementation: Anisotropic-Fitted Mesh

5.3.2.1 Flagging

After anisotropic mesh adaptation, the cut elements by the interface of the immersed geometry need to be detected. Since each simplex can only be owned by one processor, algorithm 3 can be used on each partition (sub-domain): with a loop over the local elements of each sub-domain, the local flagged elements K_{f_i} are detected and then isolated. The collection of all flagged simplices form a new sub-mesh or hall $\mathcal{H}$:

$$\mathcal{H} = \bigcup_{i=0}^{card(p)} K_{f_i} \tag{5.3}$$

A new global numbering or data structure applied on $\mathcal{H}$ is then needed to establish the communication between the processors. Figure 5.4 illustrates on a immersed 2D circle the reduction of the whole mesh to only the hall $\mathcal{H}$ and a segment of the new global numbering of the elements.

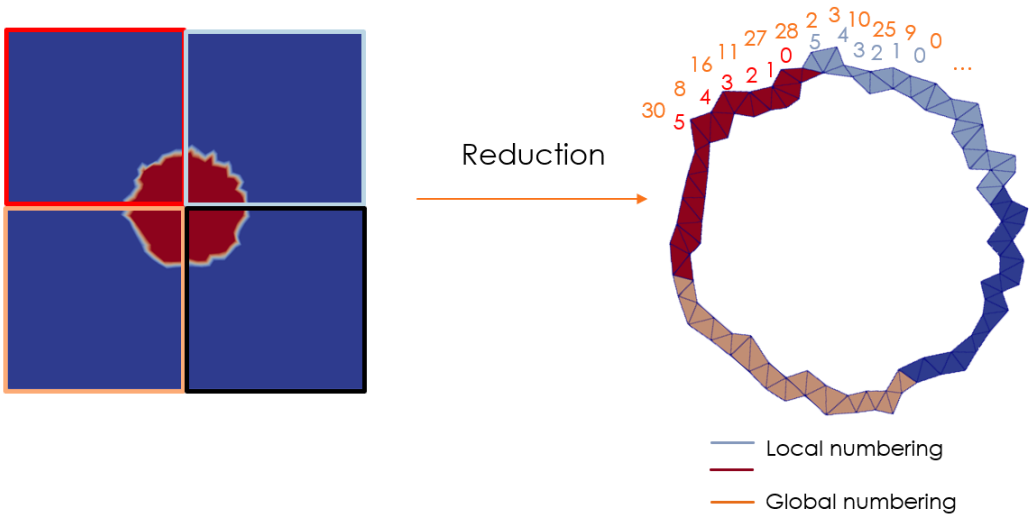


Figure 5.4: Illustration on an immersed circle of the creation of a hall $\mathcal{H}$ and a new global numbering of the vertices and simplices forming $\mathcal{H}$, with each color representing a different partition associated with a processor p_i

For the rest of this section, all the work and algorithm described will be applied on this new hall $\mathcal{H}$, unless otherwise specified.

5.3.2.2 R-adaptation

In this step, the properties of the level-set as a signed distance function are inserted to determine the vertex of each element having the minimum distance to the interface.

On each partition, a loop on the vertices of each element is done to mark the vertex having the minimum distance. However, special care is needed to deal with the shared vertices. To do so, a point-to-point communication is done as follows:

1. For each partition of $\mathcal{H}$, a new local numbering for the vertices is put in place.

2. Then, the shared vertices and their respective owner are determined.

3. For each ghost vertex, its local position on the local processor is send to its owner and its "global" index as seen by the owner is send back to the local processor: this is done by using twice a point-to-point communication, $MPI\_Sendrecv$.

With these three steps, a global numbering of the vertices forming $\mathcal{H}$ is accomplished using an all-to-all communication $MPI\_Allgatherv$. Hence, a mapping between local partition numbering and global numbering on $\mathcal{H}$ is created:

$$v_i^p \mapsto v_i^{\mathcal{H}} \tag{5.4}$$

This global numbering allows a last point-to-point communication between the adjacent processors to communicate all the marked vertices. Finally, a sequential bubble sort makes sure that no element has all its vertices marked.

Then following the same procedure explained in section 4.4.1 and algorithm 4, the coordinates of each marked vertex are replaced by the coordinates of its associated global virtual vertex v_f: this is achieved with a series of mappings and all-to-all communication:

1. First, for each marked vertex $v_{m_i}^p$, the set of virtual vertices v_{f_i} and their respective distances $d_{v_{m_i}}$ are determined on each partition:

$$v_{m_i}^p \mapsto \langle d_{v_{m_i}}, v_{f_i} \rangle \tag{5.5}$$

2. Then, using an all-to-all communication $MPI\_Allgatherv$ all the distances d_{v_m} are communicated to all processors.

3. Following this communication, a global sorting of each set of all the virtual vertices, for each global marked vertex $v_{m_i}^{\mathcal{H}}$, on all processors is applied to end up with the following mapping:

$$v_{m_i}^{\mathcal{H}} \mapsto \langle d_{glb_{v_{m_i}}}, p_{v_f} \rangle \tag{5.6}$$

with p_{v_f} the processor that owns the final virtual vertex v_f having the global minimum distance $d_{glb_{v_{m_i}}}$ associated with the marked vertex v_{m_i}.

4. Finally, the coordinates x_{m_i} of v_{m_i} are locally updated as follows:

$$x_{m_i} = x_{v_f} \tag{5.7}$$

with x_{v_f} the coordinates of the final virtual vertex associated to v_f.

5.3.2.3 Edge Swapping

As seen in section 4.4.1, R-adaptation isn't always enough, a local topology optimization: edge swapping is needed. However, with parallel programming, two configurations of algorithm 5 can be found affecting the sparsity pattern and allocation. In this section, a brief description of the sparsity pattern is presented followed by an explanation of the different configurations.

▷ **Sparsity Pattern**

The study of a mathematical properties of a partial differential equations is usually done on a general weak formulation, verifying the Lax-Milgram Theorem [7]:

$$\begin{cases} \text{Find } u \in V \subset H, \text{ such that:} \\ a(u,v) = L(v), \qquad\qquad \forall v \in V \end{cases} \tag{5.8}$$

with $a(.,.)$ a coercive continuous bilinear form on $V \times V$ and $L(.)$ a continuous linear form on V.

The approximate or discrete solution u_h to a problem can then be constructed on a finite dimensional space $v_h \subset V$, to have the approximate weak problem read:

$$\begin{cases} \text{Find } u \in V_h \subset H, \text{ such that:} \\ a(u_h,v_h) = L(v_h), \qquad\qquad \forall v_h \in V_h \end{cases} \tag{5.9}$$

with $a(.,.)$ a coercive continuous bilinear form on $V_h \times V_h$ and $L(.)$ a continuous linear form on V_h, while V_h is a finite dimensional approximation space characterized by a discretization parameter h.

To define the discrete space V_h, a basis $(\varphi_1, ..., \varphi_{N_{V_h}})$ of V_h needs to be constructed with $N_{V_h} = dim(V_h)$, on which the discrete solution is written as:

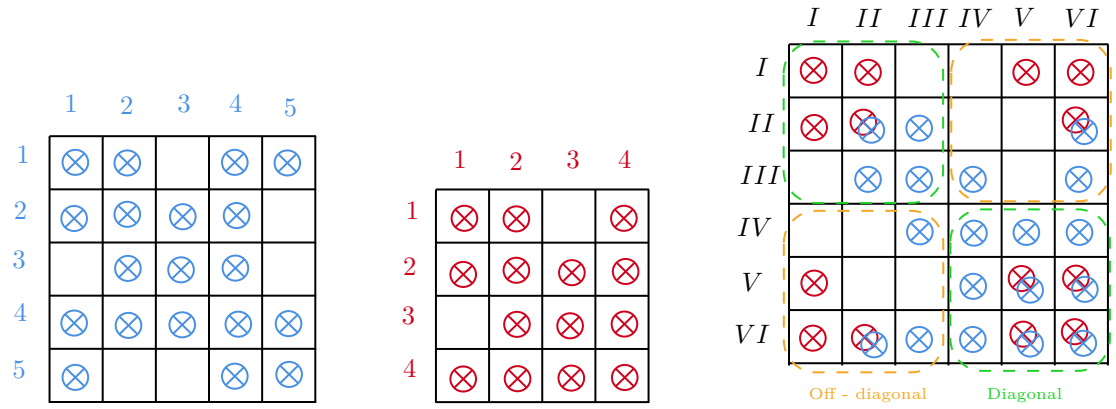
$$u_h = \sum_{j=1}^{N_{V_h}} u_j \varphi_j \tag{5.10}$$

with $\{u_j\}$ a family of N_{V_h} real numbers called global degrees of freedom and $\{\varphi_j\}$ a family of N_{V_h} elements of V_h called global shape functions.

Using the Galerkin method [8], the approximate problem can then be written as a linear system of algebraic equation:

$$\mathbf{Au} = \mathbf{b} \tag{5.11}$$

In parallel computation, the global matrix $\mathbf{A}$ is computed by assembling the local matrices $\mathbf{A}_i$ assembled on each partition p_i. Figure 5.5 shows the matrix assembly of the two sub-domains represented in Figure 5.3. The result of the global assembly of the entire domain leads to a sparse matrix.



(a) Local matrix associated with the blue sub-domain

(b) Local matrix associated with the red sub-domain

(c) Global matrix

Figure 5.5: Matrix assembly of two sub-domains

To store and manipulate such matrices, a special data structure taking advantage of the sparse structure is used: in our library, the non-zero elements are stored by rows, along with an array of corresponding column numbers and an array of pointers to the beginning of each row.

Note that, it's important to preallocate the memory needed for a sparse matrix since the dynamic process of allocating new memory and copying from an old to new storage is intrinsically expensive.

In this work, the Portable Extensible Toolkit for Scientific computation (PETSc) is used to to store and solve large systems in parallel. It is an open-source suite of data structures and routines to solve scientific applications modeled by PDEs [9].

▷ **Edge Swapping Configuration**
When dealing with edge swapping, two configurations can be found:

(a) **Basic Configuration**
With the basic configuration, represented in figure 5.6a, the shared edge to be swapped falls exactly on the inter-process interface, and when flipped the inter-process interface changes along with it. This means that the local number of vertices owned by each processor remains the same. However, this type of configuration entitles that on each partition p_i the connectivity, ownership and assembly of an existent element have to be completely deleted and a new element has to be constructed and added the list of elements. For instance, in figure 5.6a, the first element K_0, initially belonging to p_i, connected to the set of vertices: v_j owned by p_i and v_i, v_k shared, give the local mapping:

$$\begin{cases} K \to V : \\ K_0 \mapsto v_i, v_j, v_k \end{cases} \tag{5.12}$$

After swapping, a redistribution hence a renumbering of the vertices needs to be done to eventually reconstruct and add the new element K_0' mapping to v_i owned by p_i and v_j, v_r shared:

$$\begin{cases} K \to V : \\ K_0' \mapsto v_i, v_j, v_r \end{cases} \tag{5.13}$$

Similarly, the second element K_1 owned by p_j associated to v_r (owned) , v_i and v_k (shared) becomes K_1' mapping to v_k (owned) , v_j and v_r (shared). So eventually, for each processor, a new numbering taking into account the removal of a vertex, an element and the addition of a new vertex and element, and a new communication of the shared vertices need to be established.

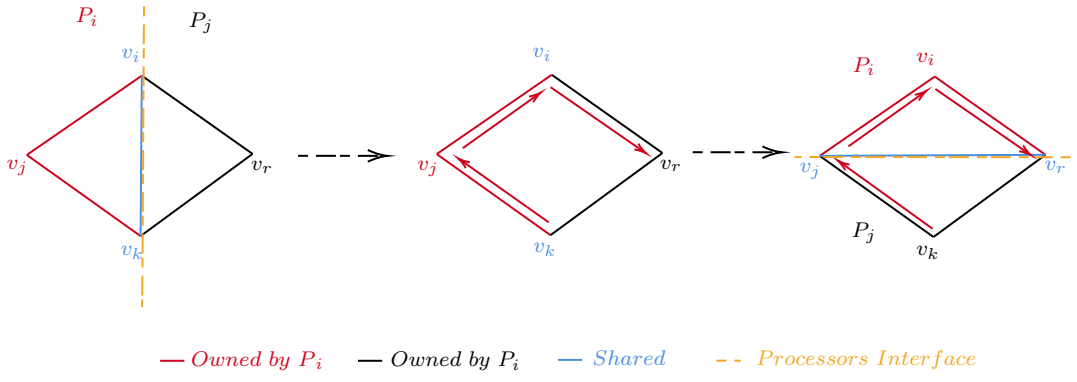
(b) **Z-Configuration**
The Z-Configuration, as it names states, increases the inter-process interface by only adding one extra shared vertices to the initial shared set on each partition, resulting in a larger inter-process interface having a Z shape as can be seen in Figure 5.6b.

Indeed, in figure 5.6b, v_j (or v_r) initially only owned by p_i (by p_j) and not seen by p_j (by p_i), is also communicated to p_j (p_i) and becomes shared by the

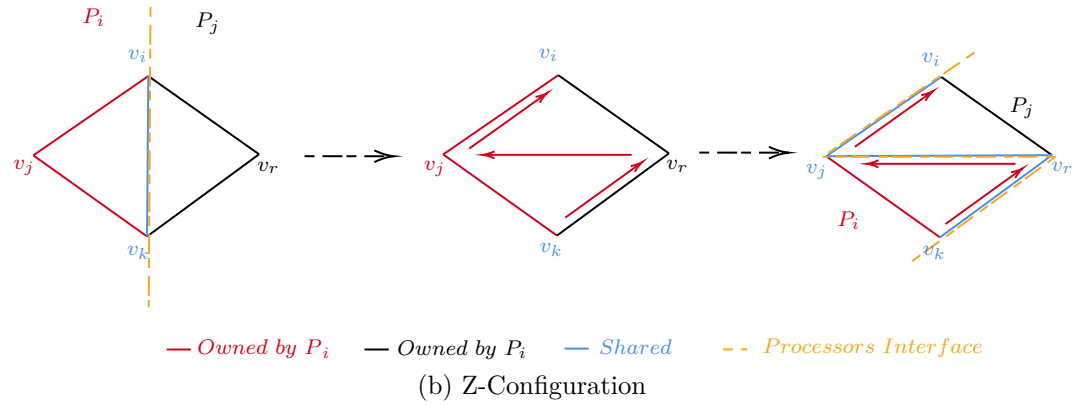
two processors. A simple point-to-point communication is done here to add the newly shared vertex to the adjacent partition without losing the initial ownership of the respective vertices.

In this work, the Z-configuration (Figure 5.6b) has been chosen since, for each pair of elements subject to swapping since only one vertex needs to be added to each partition.

To overcome the change in the sparsity pattern, an extra padding, dependent on the space dimension, is allocated on each off-diagonal entries to allow for more memory space when constructing the sparse matrix. Therefore, any new addition of a shared vertex is already accounted for and we avoid the destruction and reconstruction of the entire matrix each time a swap occurs.



(a) Basic Configuration



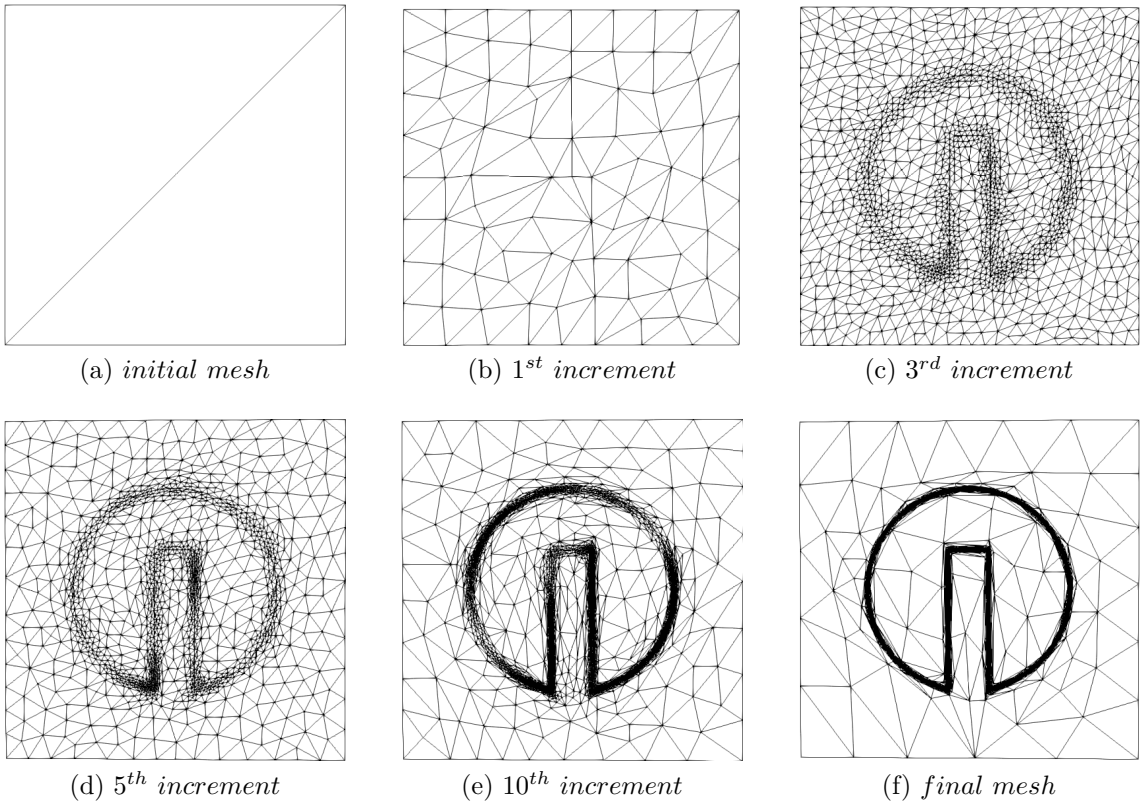
(b) Z-Configuration

Figure 5.6: Illustration of configurations of 2D parallel edge swapping

5.3.3 2D Illustration of the Parallel Implementation

We consider the well-known Rudman-Zalesak slotted disk immersed in a 0.5×0.5 square domain. Starting with a very coarse two-element mesh, the parallel

anisotropic fitted algorithm is applied on the geometry. As can be seen in Figure 5.7, the mesh evolves as the geometry is detected and the anisotropic fitted mesh is created.



(a) *initial mesh* (b) 1^{st} *increment* (c) 3^{rd} *increment*

(d) 5^{th} *increment* (e) 10^{th} *increment* (f) *final mesh*

Figure 5.7: Anisotropic Fitted mesh evolution of the Rudman-Zalesak slotted disk

5.4 3D Modeling

The 3D parallel modeling of the anisotropic fitted interface of an immersed geometry follow almost the same algorithm presented in section 5.3 for a parallel two-dimensional computation, except for the last step: edge swapping. Just as in 2D, the mesh is subject to an anisotropic adaptation, capturing the complex immersed geometry. Then, the cut cells, i.e, tetrahedral elements, are flagged and isolated to create the sub-mesh $\mathcal{H}$. The properties of the level-set as a signed distance function are inserted on the flagged elements in order to determine and mark the vertices that are goning to be subject to R-adaptation. Once the R-adaptation is done, edge swapping is replaced by a three-dimensional permutation as will be explained in this section.

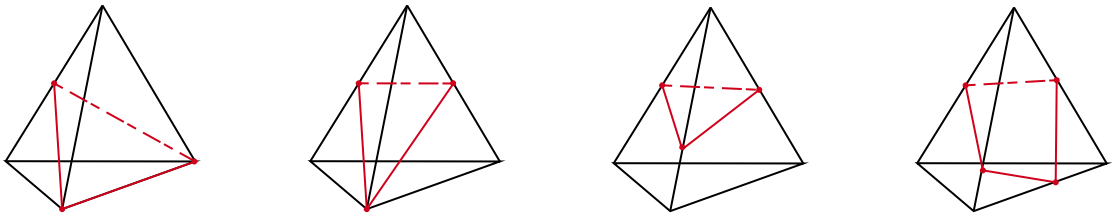
In 3D, the immersed geometry's interface can cut the tetrahedral elements in four different ways, creating an immersed "virtual" plane:

1. three intersection points, with two of them coincide with real vertices of the cell and one virtual vertex forming the intersection between an edge of the cell and the virtual plane (Figure 5.8a).

2. three intersection points, with only one of them coinciding with the real vertices of the cell, and two vertices forming the intersection between two edge of the cell and the virtual plane (Figure 5.8b).

3. three intersection points, with none of them coinciding with any real vertex, all the points intersect an edge of the cell (Figure 5.8c).

4. four intersection points that are all virtual (Figure 5.8d).



(a) Case 1 (b) Case 2 (c) Case 3 (d) Case 4

Figure 5.8: Illustration of the possible cases for the 3D cut cells

In order to ensure the creation of a fitted mesh, accurately defining and capturing the geometry, each one of these cases need to treated individually.

Case 1:

The treatment of this case is very straightforward. The geometry's interface cuts the cell at one virtual vertex e_0 and two real vertices. This configuration gives readily two new tetrahedra. Figure 5.9 shows how the initial cell $\{v_0, v_1, v_2, v_3\}$ is decomposed into two new cells adjacent and connected by the new facet $\{e_0, v_2, v_3\}$.

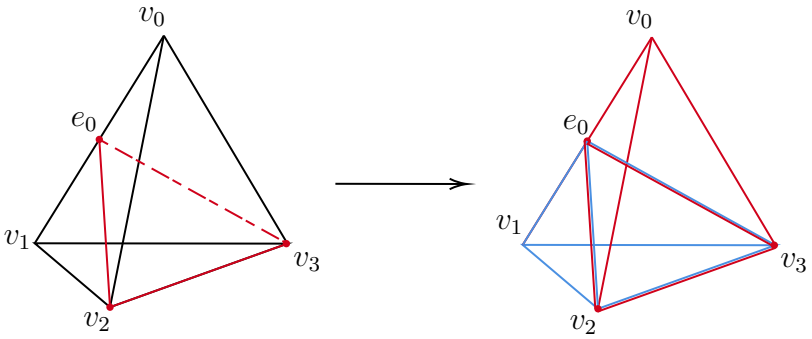


Figure 5.9: 3D cutting of case 1

Case 2:

In this case, the interface intersects the cell at two virtual points e_0 and e_1 and one real vertex. This configuration leads to decomposing the initial cell into a total of 4 tetrahedra.

The construction of the first one is pretty intrinsic: it's made of the common node of the two intersected edges of the cut cell and the cut plane. In Figure 5.10, the intersected edges are $\{v_0, v_1\}$ and $\{v_0, v_3\}$, so the common node is v_0 and the first tetrahedron K_1' is made of the vertices $v_0, e_0, v_2,$ and e_1.

After the construction of the first tetrahedra, the "base" of the initial cell is left with vertices $v_1, e_0, e_1, v_3,$ and v_2. In order to construct new tetrahedra, a vertex m needs to be introduced on the edge opposite to the edge formed by the virtual vertices. In Figure 5.10, the new vertex m is added on the edge $\{v_1, v_3\}$ such as the coordinates of m, x_m, are obtained from the coordinates x of vertices v_1 and v_3 as such:

$$x_m = \frac{x_{v_1} + x_{v_3}}{2} \tag{5.14}$$

Once introduced, the vertex m is linked to the vertices of the cut plane and the 3 new tetrahedra are constructed:

- $K_2' \mapsto \{v_1, v_2, m, e_0\}$
- $K_3' \mapsto \{v_2, v_3, m, e_1\}$
- $K_4' \mapsto \{v_2, e_0, e_1, m\}$.

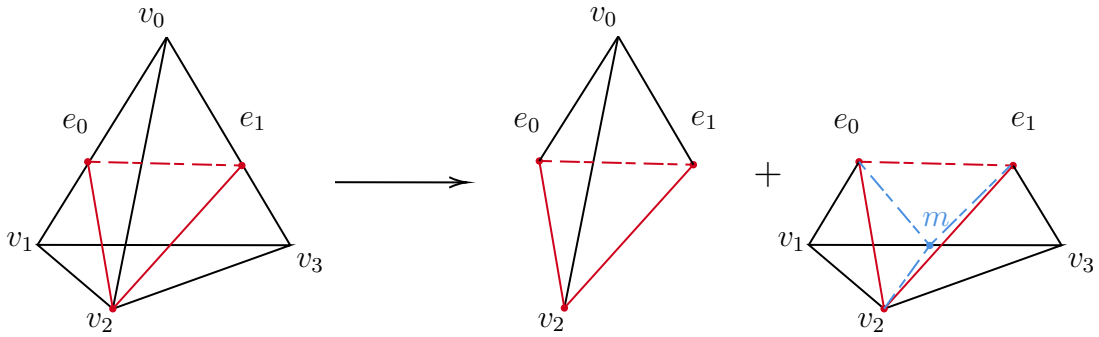


Figure 5.10: 3D cutting of case 2

Case 3:

In this case the cut plane or interface intersects the mesh cell in 3 virtual vertices: e_0, e_1 and e_2, leading to the decomposition of the initial cell into 5 tetrahedra.

Similarly to case 2, the construction of the first cell is straightforward made of the common node and the cut plane. For the rest of the tetrahedra, a vertex m is also introduced just like in the previous case; however in this case, it results in the creation of 4 new cells. Using the vertex opposite to the cut edge by m (vertex v_2 in Figure 5.11) as a reference node, the truncated base $(v_1, v_2, v_3, e_0, e_1)$ can be treated as in case 2, and then the last tetrahedron with the third intersection point can be added (e_0, e_2, e_1, v_2).

Figure 5.11 shows the how the cut plane decomposes the initial cell into 5 new tetrahedra:

- $K'_1 \mapsto \{e_0, e_2, e_1, v_0\}$
- $K'_2 \mapsto \{v_1, v_2, e_0, m\}$
- $K'_3 \mapsto \{v_2, v_3, e_1, m\}$
- $K'_4 \mapsto \{v_2, e_1, e_0, m\}$
- $K'_5 \mapsto \{e_0, e_2, e_1, v_2\}$.

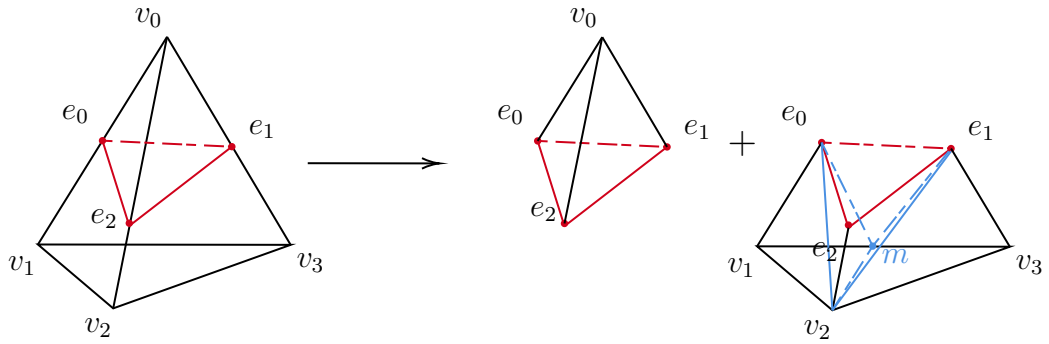


Figure 5.11: 3D cutting of case 3

Case 4:

In this case, the cut plane intersects the mesh cell into four virtual vertices. In order to create a fitted mesh with this configuration, two new vertices m_1 and m_2 are introduced as the mid-point of two adjacent edges of the initial cell's base, splitting the initial mesh cell into two. Each new created tetrahedra is treated as in case 2, resulting in total to the creation of $2 * 4$ tetrahedra.

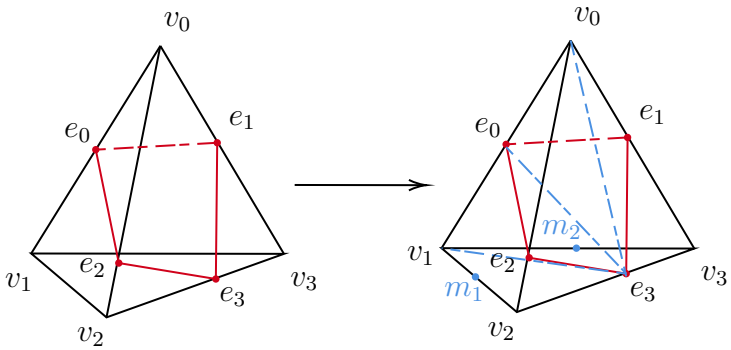


Figure 5.12: 3D cutting of case 4

Note that due to the R-adaptation, the most common cases to happen are cases 1 and 2 with at least one vertex of the cut plane coinciding with a real vertex of the cut cell.

5.4.1 3D Cutting Parallel Modeling Challenge

When partitioning the initial mesh, the mesh information and parameters are divided such that each partition has an equal load resulting in a balanced system. Each partition has a fixed number of elements and a known number of owned and shared vertices. Unlike in the 2D parallel implementation of edge swapping using Z-configuration, 3D cutting results in the construction of the new cells and vertices to the initial mesh, hence altering the initial number of elements. Since a cell cannot be shared by two or more processors, the new cells constructed will be added to the list of elements on the owner of the initial cell. This addition can affect the load balance between the processors involved. To overcome this, an improvement of the load balance through a dynamic re-distribution of the added elements must be done while minimizing the communication cost and trying not to increase the inter-process interface. Note that, the dynamic load balancing for the addition of new cells and the dynamic creation of a new mesh coupled with this parallel algorithm aren't yet supported by the Cimlib-CFD library, but developments are being made so they are integrated to the library making the parallelization of the entire 3D algorithm possible.

5.5 Conclusion

After a recall of the definitions and taxonomy, the parallel implementation of the proposed algorithm has been presented and described in details, in this chapter. A new data structure involving the creation of a sub-mesh and the renumbering locally (on each partition) and globally of the entities (elements and vertices) have been done. This numbering insures a more efficient communication between the processors especially during R-adaptation and Edge-Swapping involving vertices on the inter-process interface. The 3D extension was also described in details replacing the 2D edge-swapping with 3D cutting. Four different cases of cut cells have been studied and implemented, in order to ensure the creation of an anisotropic fitted mesh. Depending on the number of intersection points, the initial cut cell is divided into two or more new tetrahedra. One challenge however remains, the dynamic load balancing allowing us an equal re- distribution of the load on each process after 3D cutting.

Bibliography

[1] M. J. Flynn, K. W. Rudd, Parallel architectures, ACM Computing Surveys (CSUR) 28 (1) (1996) 67–70. 84

[2] P. Jimack, B. Topping, An overview of parallel dynamic loadbalancing for parallel adaptive computational mechanics codes, Parallel and Distributed Processing for Computational Mechanics: Systems and Tools (1999) 350–369. 85

[3] B. Hendrickson, K. Devine, Dynamic load balancing in computational mechanics, Computer Methods in Applied Mechanics and Engineering 184 (2-4) (2000) 485–500. 85

[4] T. Coupez, H. Digonnet, R. Ducloux, Parallel meshing and remeshing, Applied Mathematical Modelling 25 (2) (2000) 153–175. 86

[5] Y. Mesri, H. Digonnet, H. Guillard, Mesh partitioning for parallel computational fluid dynamics applications on a grid, Finite Volume for Complex Applications (2005) 631–642.

[6] Y. Mesri, H. Digonnet, T. Coupez, Advanced parallel computing in material forming with cimlib, European Journal of Computational Mechanics/Revue Européenne de Mécanique Numérique 18 (7-8) (2009) 669–694. 86, 88

[7] S. Bergman, Contributions to the theory of partial differential equations, no. 33, Princeton University Press, 1955. 90

[8] S. C. Brenner, L. R. Scott, L. R. Scott, The mathematical theory of finite element methods, Vol. 3, Springer, 2008. 91

[9] S. Balay, S. Abhyankar, M. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, A. Dener, V. Eijkhout, W. Gropp, et al., Petsc users manual (2019). 92

Chapter 6

Quenching Process

Contents

Résumé en Français

Dans ce chapitre, tous les outils et algorithmes numériques développés et décrits précédemment sont utilisés pour simuler le refroidissement d'un solide immergé. Nous montrons comment, en quelques étapes simples : définir les propriétés du "quenchant" et du solide immergé, puis fixer les conditions limites et initiales, la mise en place d'une simulation de refroidissement peut être réalisée, puis la solution calculée. Une comparaison détaillée de différents types de maillage montre l'importance d'un maillage bien défini sur la résolution de la solution.
Dans la première section, un couplage 2D de transfert de chaleur conjugué est décrit et expliqué. Dans cet exemple, les équations de Navier-Stokes stabilisées sont couplées à l'équation de convection-diffusion-réaction (CDR), expliquée au chapitre 2, et sont appliquées sur un maillage ajusté anisotrope obtenu à l'aide de l'algorithme présenté au chapitre 3. Ce couplage aboutit à la simulation du processus de refroidissement du cylindre rectangulaire immergé.
La deuxième section se concentre sur la présentation du travail de la chaire INFINITY, qui consiste à consolider un cadre multi-échelle unifié pour la représentation et la simulation d'une application industrielle : le processus de trempe. Dans ce qui suit, l'ensemble du dispositif de simulation de la trempe, développé par les contributions des différents candidats et chercheurs de la chaire et de l'équipe CFL, sera présenté et expliqué. Ensuite, des cas industriels fournis par nos partenaires AUBERT & DUVAL et SAFRAN seront présentés avec toutes les spécifications numériques et de simulation.

6.1 Introduction

In this chapter, all the numerical tools and algorithms developed and described previously are used to simulate the cooling of an immersed solid. We show how with a few simple steps: defining the properties of the quenchant and the immersed solid and then setting the boundary and initial conditions, the setup of a cooling simulation can be achieved and then the solution computed. A detailed comparison of different mesh types shows the importance of a well-defined mesh on the solution's resolution.

In the first section, a 2D conjugate heat transfer coupling is described and explained. In this example, the stabilized Navier-Stokes equations are coupled with the convection-diffusion-reaction equation (CDR), explained in chapter 2, and are applied on anisotropic fitted mesh obtained using the algorithm presented in chapter 3. This coupling results in the simulation of the cooling process of the immersed rectangular cylinder.

The second section focuses on the presentation of the work of the INFINITY chair, which is to consolidate a unified multi-scale framework for the representation and the simulation of the industrial application: the quenching process. In what follows, the entire quenching simulation set up, developed by the contributions of the different candidates and researchers of the chair and the CFL team, will be presented and explained. Then, industrial test cases provided by our partners AUBERT & DUVAL, and SAFRAN will be presented with all the simulation and numerical specifications.

6.2 Conjugate Heat Transfer Coupling

We now consider a heat transfer application where a heated solid is immersed in a cooling cavity (Figure 6.1). A Cartesian coordinate system is used with the origin at the center of mass of the solid that has a rectangular shape of height h and an aspect ratio of $2:1$, initially at temperature T_s. The solid is fixed at the center of a cavity of height H and aspect ratio 4:1, whose walls are isothermal and have a fixed temperature T_w. Cooled air at T_c is pumped into the enclosure from two inlets located at the top of the cavity, at a velocity V_i. Two outlets for the hot air are located at the sidewalls of the cavity. Table 6.1 gives the numerical parameters used all expressed in SI units and the temperature in Celsius. Using the immersed-fitted algorithm coupled with flow and heat transfer solvers, the level-set function, the velocity and the temperature are used as multi-components criteria in order to adapt the mesh on the interface of the immersed geometry as well as the solution of the equations solved. This can be illustrated in Figure 6.5a showing how the mesh conforms to the interface. The corresponding adapted mesh colored by the

temperature distribution in Figure 6.5b shows how the numerical framework can capture accurately all boundary layers and shear regions via stretched elements. Figures 6.2 to 6.5 are a clear example of how the immersed-fitted algorithm evolves dynamically with the problem at hand: starting with a coarse mesh, the mesh progresses towards an anisotropic refined one without any intervention from the user. The problem is then solved simultaneously in the domain capturing the vortices formed and in the solid tracking its cooling evolution.

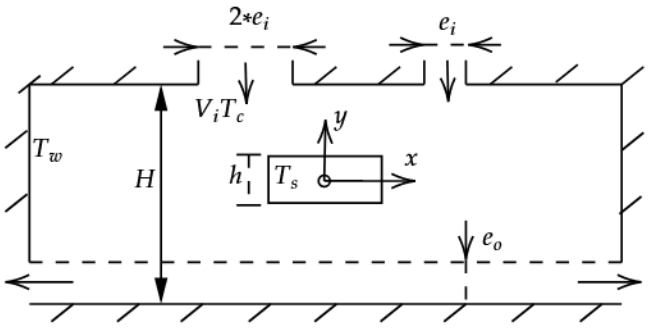
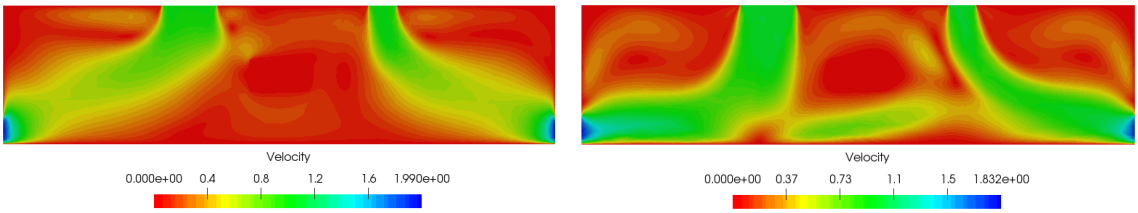


Figure 6.1: Schematic of the 2D forced convection set-up.

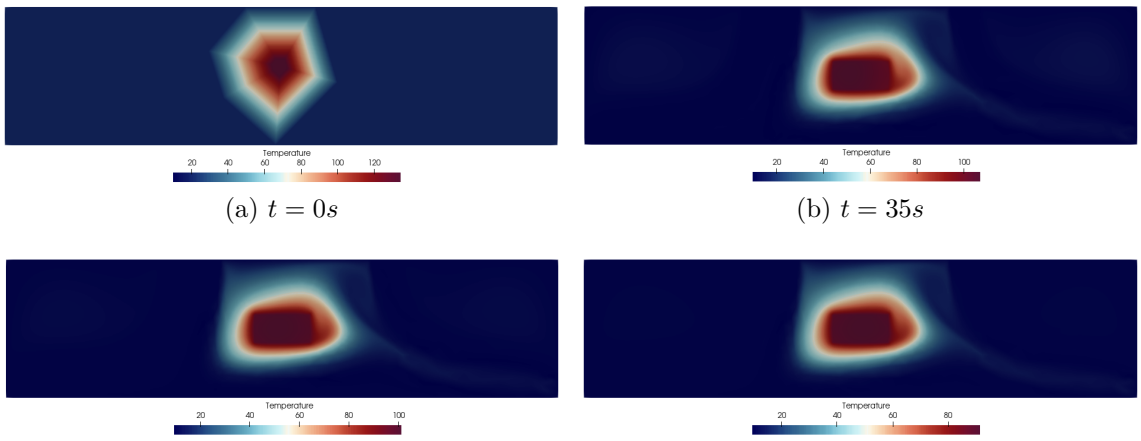
Table 6.1: Numerical parameters used in the 2D forced convection problem, with ρ the fluid density, μ the dynamic viscosity, λ the thermal conductivity, and c_p specific heat.

H	h	e_i	e_o	V_i	T_w	T_c	T_s	μ	ρ	λ	c_p	
1	0.2	0.2	0.2	1	10	10	150	0.0011		0.5	1000	Fluid
								1000	100	15	300	Solid



(a) $t = 1s$ (b) $t = 150s$

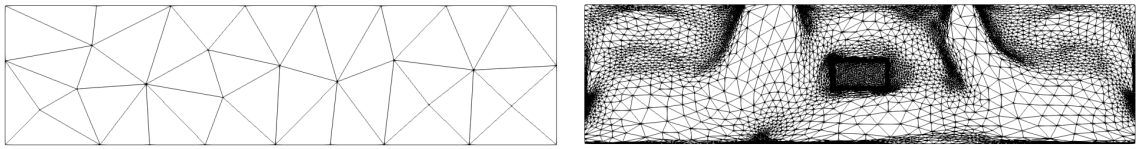
Figure 6.2: Velocity profiles inside the cavity.



(a) $t = 0s$ (b) $t = 35s$

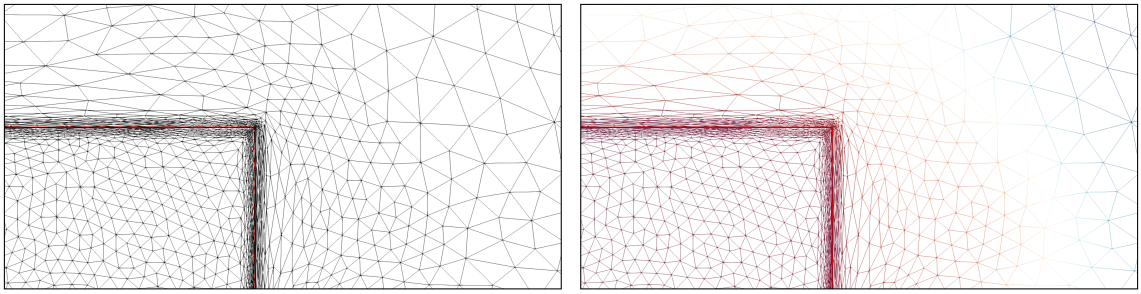
(c) $t = 70s$ (d) $t = 150s$

Figure 6.3: Temperature distribution inside the cavity.



(a) $t = 0s$ (b) $t = 150s$

Figure 6.4: Coarse initial mesh (left) and the obtained anisotropic adapted one on both the velocity, temperature and the solid level-set (right) at $t = 150s$.



(a) (b)

Figure 6.5: Zoom on the sharp immersed-fitted interface of the 2D solid.

6.3 Industrial Applications: Quenching of a Solid

The main objective of the INFINITY chair, hence of this work, is to be able to achieve a full model to simulate the fluid and solid part as well as their interaction during the quenching process. Previously developed solvers by the CFL team that simulate multi-phase flows and heat transfer are used. Note that in order to achieve a high accuracy and resolution of the solution, special attention to the mesh needs to be given: for instance, having the same element size in all the domain might cause the increase of the computational cost since the elements should be small enough to capture all the discontinuities. Therefore, the size of the elements near the fluid-solid interface should be smaller than the size in areas away from the interface. Furthermore, having anisotropic fitted elements in high gradient regions would not only allow us to achieve better precision but also to capture the jumps in different media without the need to increase the number of elements used.

In this section, industrial test cases will be presented, showing the importance of an appropriate mesh, and the simulation results compared to experimental data provided by our partners. For each test case, the quenching tank and the solid dimension with along with .STL files, if needed, were given. The initial conditions of each test case and the position of the solid in the tank were also provided. The thermal and physical properties of both the coolant and the solid were also given.

6.3.1 The Simulation Setup

As presented in Chapter 1, the cooling rate of a solid (often a metal alloy) is directly related to its microstructure, metallurgic, and mechanical properties. In this work, we consider pool quenching, where the heated solid is immersed in a liquid coolant. The cooling rate is greatly influenced by the behavior of the coolant that extracts the heat from the immersed solid. We are interested in liquid coolants that vaporize in contact with the heated components. These boiling phenomena have a direct effect on heat transfer. The cooling rate of a solid varies during the quenching process (Figure 6.6):

- During the calefaction phase, a vapor film surrounds the immersed solid. This insulation causes the cooling rate to be relatively moderate.

- A sudden increase in the rate is then observed due to nucleate boiling. During this phase, the coolant may come in direct contact with the solid surface but mainly boiling takes place as vapor bubbles are formed and then released on most of the surface of the immersed part. This phenomenon reduces the temperature of the solid drastically and is the most efficient cooling mode with heat transfer being very important.

- Finally, with boiling ceasing and the natural convection being dominant, the immersed part cools down slowly.

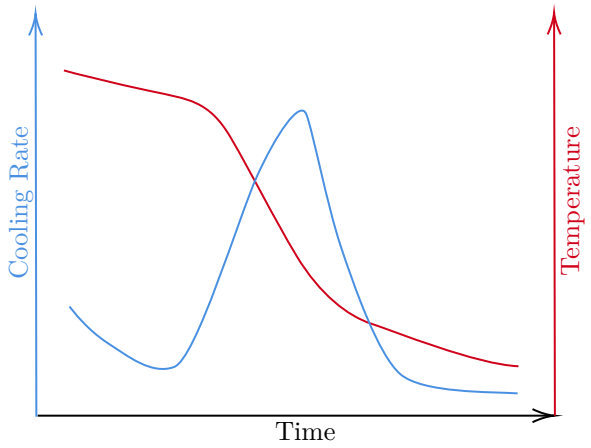


Figure 6.6: Typical cooling rate and temperature evolution during a quenching process.

Simulating the quenching process allows the manufacturers to control the phase changes taking place in the alloy, thus achieving good homogeneity and obtaining high-quality products for new designs. The direct simulation of the whole quenching process can be accomplished by using the Immersed Volume Method, described in 4.2.3.2. The IVM has the ability to account for all the features of the process in a unified way with an anisotropic mesh adaptation that provides a better description of the interfaces. However, the simulation of a complete quench is challenging and can be quite complex and costly. To overcome the computational costs, a strategy was adopted to model the full quenching process in the INFINITY chair: the desynchronization of the quenching process. This strategy consists in dividing the direct simulation into two parts: the boiling process i.e. the modeling and study of the thermo-hydrodynamic properties and the solid cooling (Figure 6.7).

- **The thermo-hydrodynamic simulation:**
  The thermo-hydrodynamic simulation represents the simulation of the boiling process at a fixed temperature over a few seconds. This process is modeled using a phase change solver where the Navier-Stokes equations and the thermal CDR equations are solved coupled with the advection of the level-set. The Navier-Stokes equations are modified to take into account the surface tension terms and mass transfer considered on the extended liquid-vapor interface via the Continuum Surface Force approach [1, 2]. Once a steady regime is reached, time averaged local heat fluxes on the solid surface, and film thicknesses are recovered in order to compute the heat transfer coefficient (HTC). Note that

h_{HTC} is a coefficient averaged over a given surface, and integrates the effect of a fluid temperature:

$$q_w = h_{HTC}\Delta T_w \tag{6.1}$$

where q_w represent the heat flux, $\Delta T_w = T_w - T_{sat}$ the temperature variation, T_w the surface temperature, and T_{sat} the saturation temperature of the coolant used. Details on its modeling can be found in [3].

- **Thermal Simulation:**
  This second step consists of the simulation of the solid part alone. The HTC, computed from the recovered heat fluxes from the thermo-hydrodynamic simulation, is set as boundary conditions and a thermal model is solved.

The two steps are computed on two separate domains that communicate together. The main advantage of this desynchronization is that it offers the precision of a direct numerical simulation at an affordable computational cost.

In what follows, the quenching framework has been applied to two test cases, provided by the industrial partners AUBERT & DUVAL, and SAFRAN.

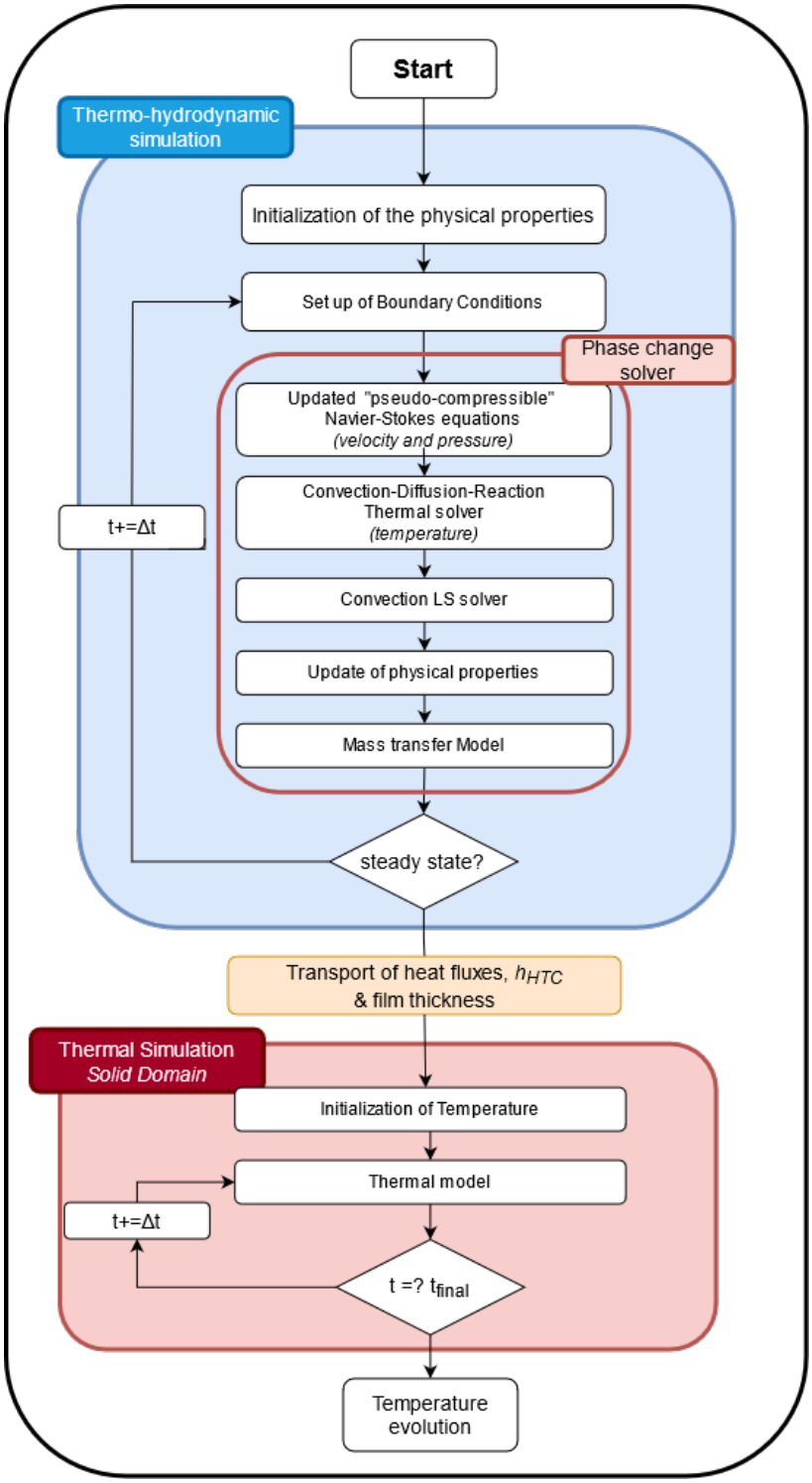


Figure 6.7: Solving procedure of the quenching process.

6.3.2 Test case 1: Quenching of an immersed cylinder

The first test case considers an immersed cylinder having a diameter of $150mm$ and a height of $150mm$ immersed in a $(1 \times 1 \times 1.2)m$ quenching tank (Figure 6.8). The physical properties of the coolant (see Table 6.2) and the solid part were provided by AUBERT & DUVAL. The cylinder is positioned horizontally at the center of the tank with its axis parallel to the tank's walls. The total cooling process is around 24 minutes. The initial conditions are :

- The initial solid temperature is: $T_s = 980°C$
- The initial water temperature is: $T_w = 30°C$

Table 6.2: Numerical parameters considered for water and vapor, with ρ the fluid density, μ the dynamic viscosity, k the thermal conductivity, and c_p specific heat.

	$\rho(kg/m^3)$	$\mu(Pa.s)$	$c_p(J/kg/°C)$	$k(W/m/K)$
Water	1000	$5 \cdot 10^{-3}$	4216	$6.79 \cdot 10^{-1}$
Vapor	$6 \cdot 10^{-1}$	$1 \cdot 10^{-3}$	2030	$2.7 \cdot 10^{-1}$

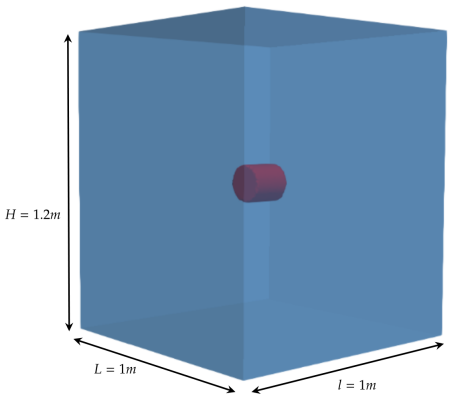


Figure 6.8: Test Case 1: Quenching tank and immersed cylinder at T_s and T_w.

Before starting the simulation, a mesh adapted to the problem has to be generated. Different types of meshes can be used:

- **Isotropic Fitted mesh:**
  A "filled" fitted (BF) mesh is tested. As can be seen in Figures 6.9 and 6.10 isotropic mesh elements accurately define the geometry at the interface. The mesh is refined near the interface where smaller isotropic elements can be found to deal with high gradients and temperature jumps, whereas in regions

away from the interface the size of the elements increase linearly. In order to accurately capture the discontinuities and the temperature variations at the interface, a high number of elements is required to ensure that the interface is well-defined, meaning a significant increase in the computational cost. The mesh shown in Figures 6.9 and 6.10 is made of more than 600,000 elements, and as can be seen, is not sufficiently refined at the interface.

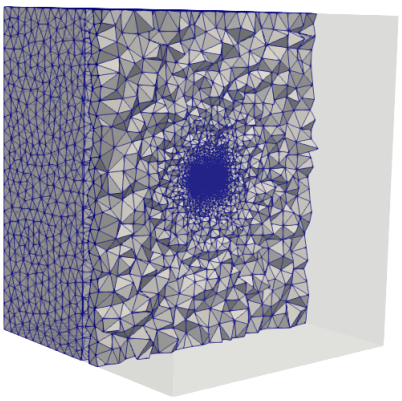
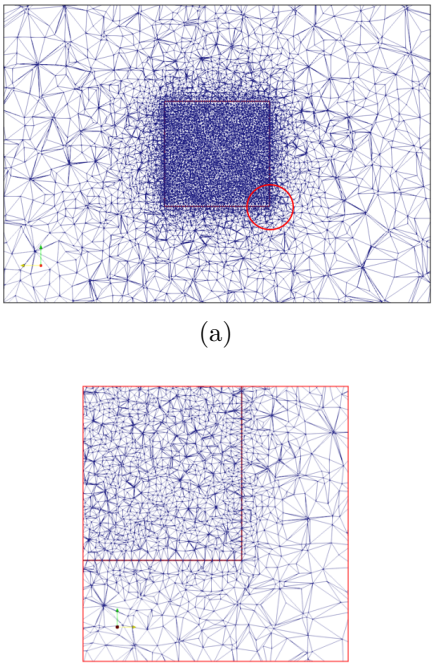


Figure 6.9: Test Case 1: Filled Body-Fitted mesh.



(a)

(b)

Figure 6.10: Test Case 1: Zoom on a cut of the filled BF mesh

- **Anisotropic Boundary layers mesh:**
  An anisotropic boundary layer mesh can also be considered. This mesh allows us to vary the element size in different regions of the mesh: the mesh size increases as we move away from the interface (Figure 6.11). First, an anisotropic mesh adaptation is applied near the interface generating very stretched elements and allowing a better resolution of the high gradients present in this region. Then a box surrounding the solid is defined with a specific element size. Within this box, the evolution of the bubbles formed during the cooling process can be tracked. However, with these predefined meshes, the anisotropic adaptation cannot be achieved dynamically as the thermo-hydrodynamic simulation is solved, and the advantages of fitted meshes, precisely capturing the

interface and setting the boundary conditions, aren't achieved.

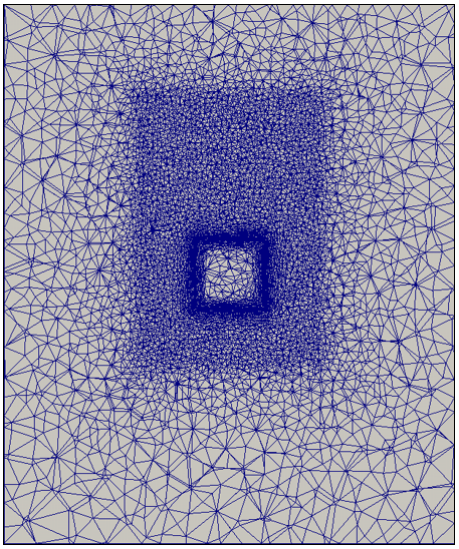


Figure 6.11: Test Case 1: Cut of anisotropic boundary layer mesh.

- **Cavity Anisotropic Body-Fitted mesh:**
  A cavity anisotropic body-fitted (BF) mesh is tested. Using this type of mesh, the solid is extracted from the fluid domain forming a cavity. The properties of the solid are taken into account via the boundary conditions applied to the cavity and then, using the mixing laws at the interface, the effect of the solid on the fluid is considered.

  Figure 6.12 shows an anisotropic cavity BF mesh where the interface is precisely defined by the nodes of the mesh, since the solid has been extracted from the fluid domain. Anisotropic mesh adaptation has been applied at the interface as per section 3.3.1. The stretched elements, which better define the geometry, form a thickness around the interface allowing a better resolution of the gradients and discontinuities near the solid. Because of the anisotropic properties, fewer elements near the interface are needed: for this simulation, a maximum of 400,000 elements is used. Using the multi-criteria adaptation (defined in section 3.3.5) the mesh is adapted following different variables and the creation of the vapor film as well as the bubbles evolution can be tracked.

- **Anisotropic Fitted mesh:**
  An anisotropic fitted mesh allows us to combine all the advantages of above mentioned meshes:

  ▷ **A "filled" mesh:** the resolution of the problem and equations is done simultaneously on the fluid and the solid. The cooling rate of the core of

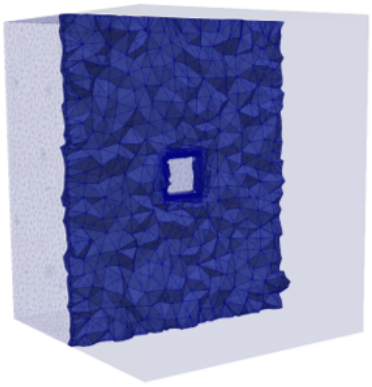
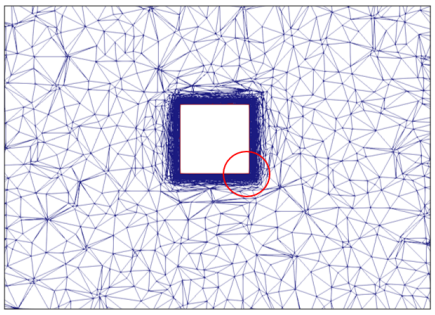
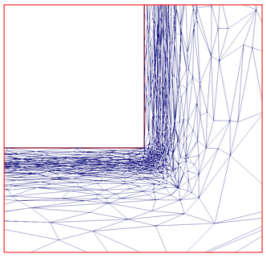


Figure 6.12: Test Case 1: Cavity-Anisotropic BF mesh.



(a)



(b)

Figure 6.13: Test Case 1: Zoom on a cut of the cavity-anisotropic BF mesh.

the solid can also be tracked during the thermo-hydrodynamic simulation, unlike when using a "cavity" BF mesh.

- ▷ **A Fitted mesh:** the boundary of the immersed geometry, i.e. the fluid-solid interface, is well captured with the vertices of the mesh falling directly on the interface. This allows to accurately impose the boundary conditions and track the evolution of the solution precisely on the interface.
- ▷ **An Anisotropic mesh:** the geometry of the immersed part is well-defined independently of its complexity. The stretched elements follow all its curvature and angles defining a narrow band region near the interface. This narrow band region allows us to deal with high gradients, jumps and discontinuities at the interface.
- ▷ **A Boundary layer mesh:** having different element sizes (Figure 6.15) while moving away from the interface allow to follow the evolution of the vapor bubbles forming and capture their effect on the cooling process without increasing the number of elements of the domain and hence increasing the computational cost.

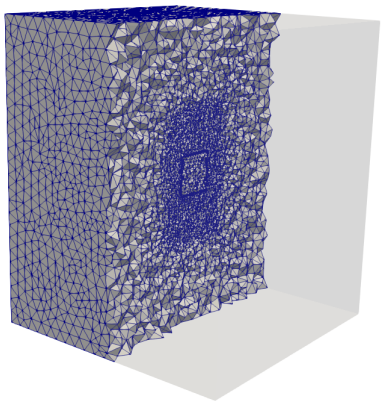


Figure 6.14: Test Case 1 :Anisotropic fitted mesh for the immersed cylinder.

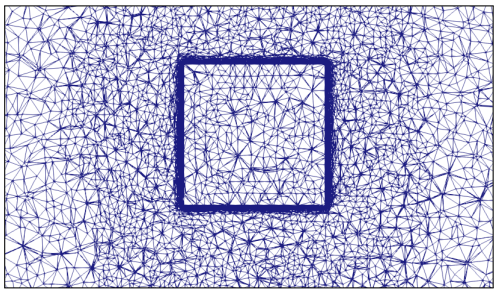
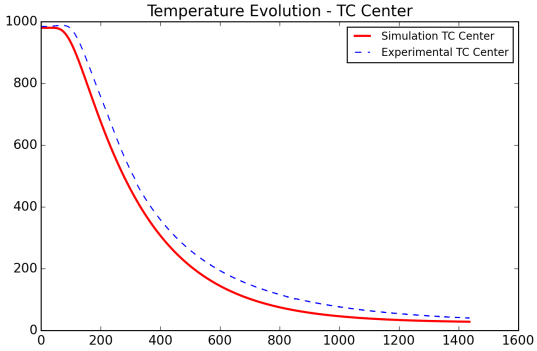


Figure 6.15: Test Case 1: Zoom on a cut of the anisotropic fitted mesh.

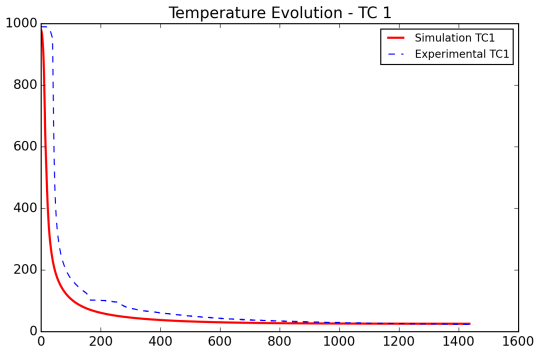
For the resolution of this test case, the anisotropic fitted mesh is used along with the mixing laws at the interface.

The strategy presented in the previous section 6.3 is used: first the thermo-hydrodynamic simulation is computed until a certain vaporization pattern is achieved reaching a steady state. Recall, that in this first part, the updated pseudo-compressible Navier-Stokes equations and the CDR equations are solved, representing the phase change solver. The heat fluxes are then computed and transported to the thermal simulation. Finally, the temperature evolution can be extracted and plotted.

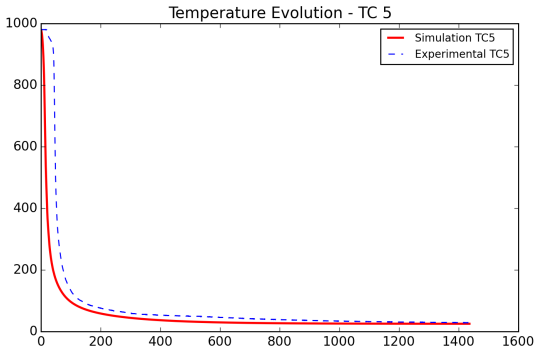
The graphs obtained show how the adopted strategy, with the use of an anisotropic fitted mesh gives in good accuracy and resolution since the results obtained are in coherence with the experimental results. Eight sensors were placed at different positions at the boundary of the solid, and one at its center. Figure 6.16 compares the obtained results to the experimental results provided by the industrial partner. Figure 6.16a shows the temperature evolution obtained at the center of the solid and Figures 6.16b and 6.16c show the temperature for two different thermo-couples placed at the vicinity of the interface.



(a)



(b)



(c)

Figure 6.16: Test Case 1: Temperature evolution for the thermo-couple in the center (a) and two thermo-couples in the vicinity of the interface (b & c).

6.3.3 Test case 2: Quenching of an immersed cylinder

This test case considers an immersed cylinder having a radius $0.15m$ and a height of $0.15m$ immersed in a $(4.7 \times 3.1 \times 1.75)m$ tank filled with water. Figure 6.17 shows the position of the cylinder in the tank. The main challenge here is the dimension of the domain that is relatively big and therefore to be able to simulate the cooling process without an increase in the computational cost. The total cooling process is of 15 minutes. The initial conditions are :

- The initial solid temperature is: $T_s = 985°C$
- The initial water temperature is: $T_w = 25°C$

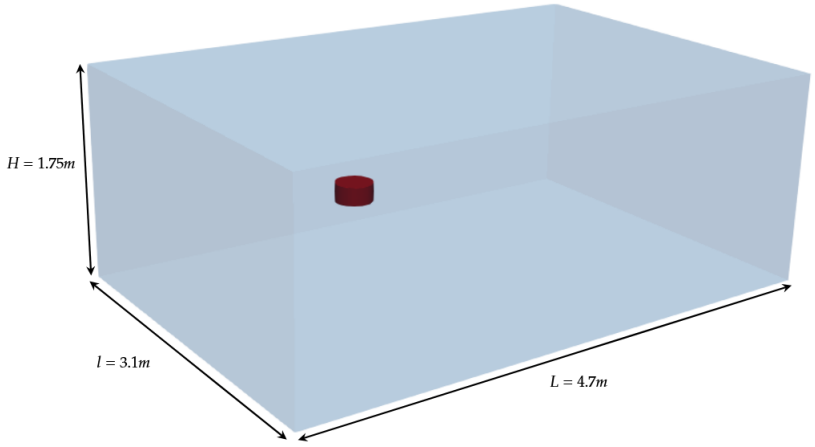


Figure 6.17: Test case 2: Quenching tank and immersed cylinder at T_s and Tw.

Similarly to the previous test case, several meshes have been tested. Figures 6.18 to 6.24 show the different meshes that can be used: filled BF mesh, anisotropic boundary layer mesh, anisotropic BF mesh and anisotropic fitted mesh.

The phase change evolution liquid/vapor can be found in Figure 6.25. Because of the high temperature gradient at the fluid-solid interface, vaporization and ebullition occur directly at the interface. As explained in section 6.3.1, a vapor film surrounds the solid part creating an isolating surface, heat transfer mainly occurs by conduction (figure 6.25a). As vapor bubbles start to form, the cooling rate start to increase reaching nucleate boiling (Figures 6.25b and 6.25c). The thermo-hydrodynamic simulation is computed until steady state is achieved, this can be detected by a repetitive vaporization pattern as seen in figure 6.25c.

The obtained results are compared with the experimental ones provided by SAFRAN. Figure 6.26 shows the temperature evolution of the immersed cylinder using thermo-couples positioned at the center and near the vicinity of the interface.

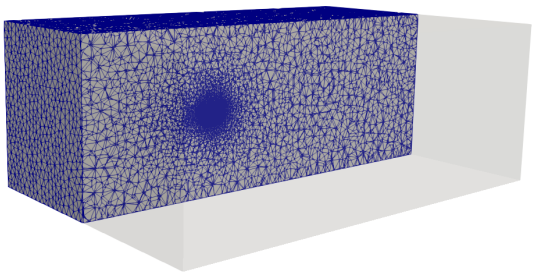
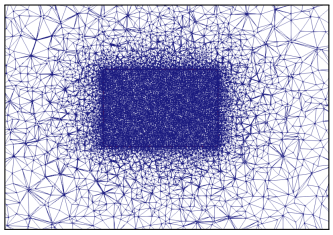
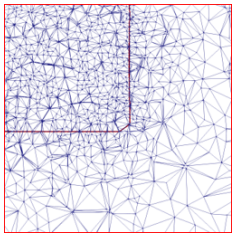


Figure 6.18: Test Case 2: Filled Body-Fitted mesh.



(a)



(b)

Figure 6.19: Test Case 2: Zoom on a cut of the filled BF mesh

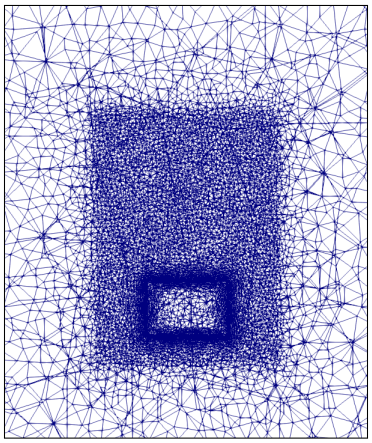


Figure 6.20: Test case 2: Cut of anisotropic boundary layer mesh.

As can be seen, the obtained results show a good correlation with the experimental ones.

Note that the mesh used to solve the presented test cases is predefined, adapted before the simulation. For larger and more complex solid parts, the phase transformation occurring inside the solid due to its metallurgic and elasto-plastic properties might cause the solid to deform. Therefore, a dynamic adaptation of the mesh near the interface is needed to ensure that the mesh captures exactly the interface to

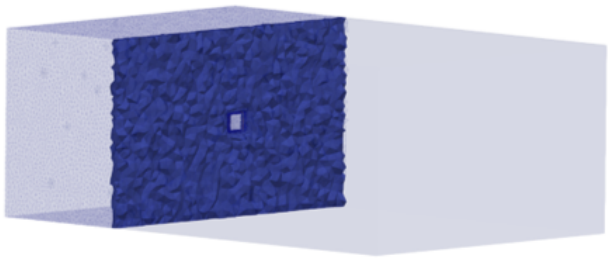


Figure 6.21: Test Case 2: Anisotropic BF mesh.

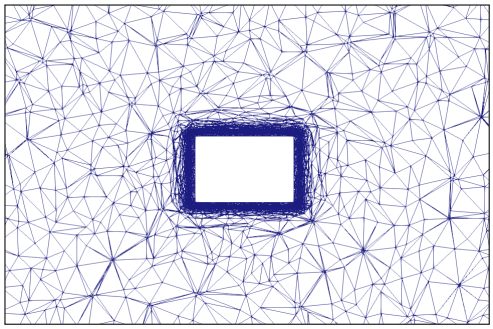


Figure 6.22: Test Case 2: Zoom on a cut of the anisotropic BF mesh.

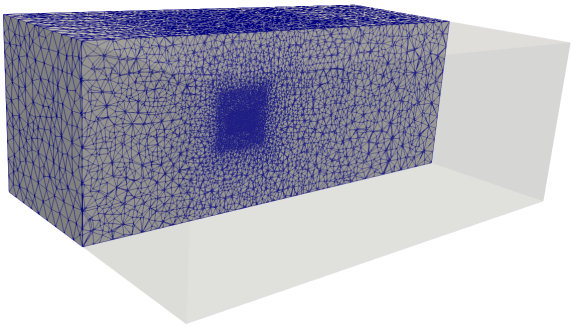


Figure 6.23: Test Case 2: Anisotropic fitted mesh for the immersed cylinder.

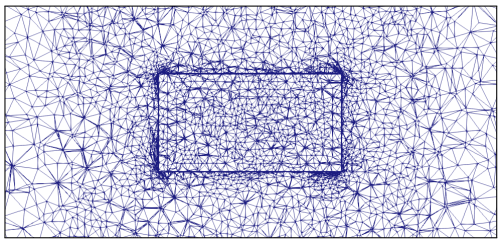
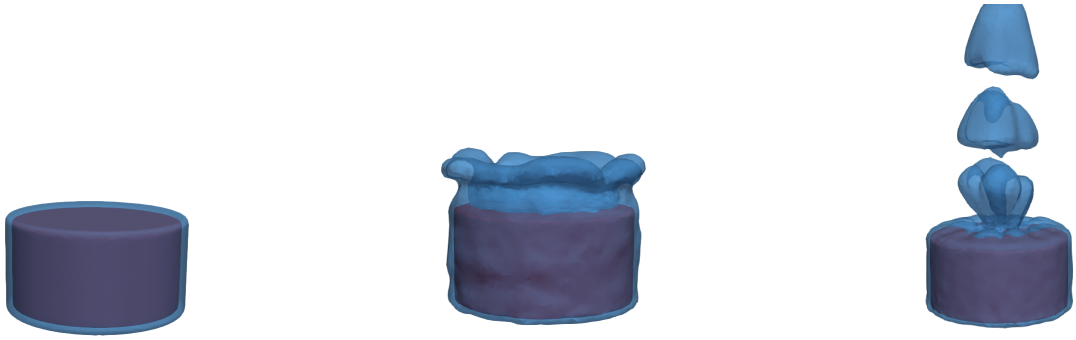


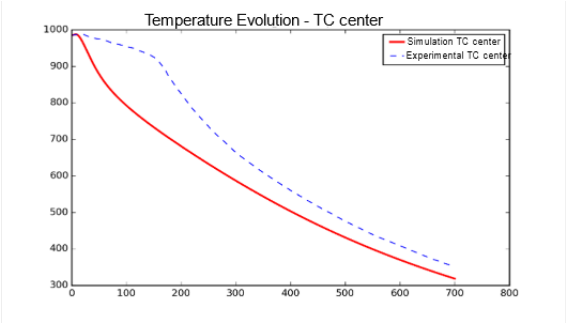
Figure 6.24: Test Case 2: Zoom on a cut of the anisotropic fitted mesh.



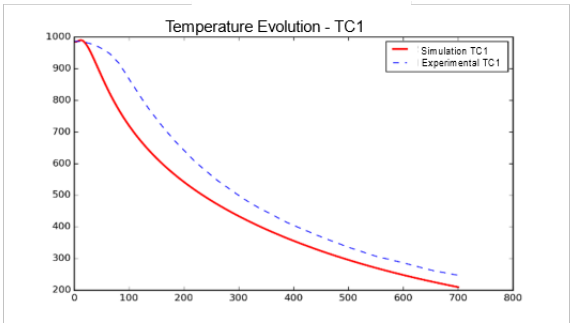
(a) (b) (c)

Figure 6.25: Test case 2: Bubble formation and evolution.

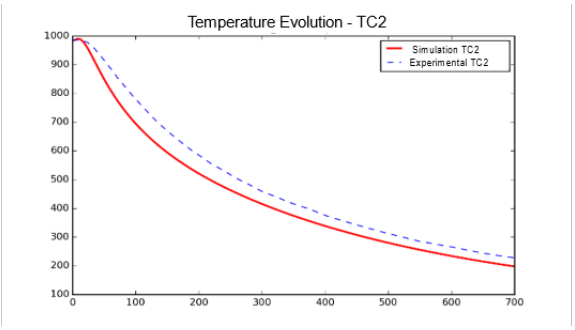
account for these displacements. This can be done using the anisotropic fitted mesh algorithm presented in this thesis (Chapters 4 and 5), once the dynamic parallel



(a)



(b)



(c)

Figure 6.26: Test Case 2: Temperature evolution for the thermo-couple in the center (a) and two thermo-couples in the vicinity of the interface (b & c).

load balancing is finalized.

6.4 Conclusion

In this chapter, the quenching model and simulation setup have been described. It is based on an extrapolation of heat fluxes from a few seconds of simulations of

the fluid domain, and then integrates this information inside a thermal solver along with correlations to account for the entire boiling curve. The 3D test cases presented compare different types of meshes focusing on the advantages of an anisotropic fitted mesh: accurate detection of the geometry and the fluid-solid interface, the ability to directly impose the boundary and initial conditions, and the use of a monolithic approach to track the cooling rate of the immersed solid. Finally, the numerical results obtained are coherent with the experimental results provided by our partners.

Bibliography

[1] M. Khalloufi, R. Valette, E. Hachem, Adaptive eulerian framework for boiling and evaporation, Journal of Computational Physics 401 (2020) 109030. 107

[2] J. U. Brackbill, D. B. Kothe, C. Zemach, A continuum method for modeling surface tension, Journal of Computational Physics 100 (2) (1992) 335–354. 107

[3] P. Archambault, S. Denis, A. Azim, Inverse resolution of the heat-transfer equation with internal heat source: Application to the quenching of steels with phase transformations, Journal of Materials Engineering and Performance 6 (2) (1997) 240–246. 108

Chapter 7

Conclusion and Perspectives

Résumé en Français

Ce chapitre résume l'objectif de la thèse en récapitulant le contributions de chaque chapitre. L'objectif de ce travail est le développement d'un maillage anisotropique adaptatif simple, rapide et robuste pour les applications CFD, et en particulier pour le processus de trempe, notre principale application dans la chaire INFINITY.
Le chapitre un introduit le phénomène de la trempe et certains des défis posés par la simulation du processus. Dans le chapitre deux, les outils numériques adoptés pour la modélisation et la simulation des écoulements de fluides et des transferts thermiques conjugués ainsi que les différentes méthodes de stabilisation, développés par le groupe CFL (Computing and Fluids) du CEMEF, et utilisés dans la simulation de problèmes multiphases et multi-composants ont été décrits. Le chapitre trois décrit, en premier lieu, les méthodes de création de maillage puis présente en détails l'adaptation anisotrope utilisée tout au long de ce travail. Dans le chapitre quatre, l'algorithme de création d'un maillage adaptative anisotrope ajusté à l'interface est décrit. Le chapitre cinq montre comment cet algorithme est appliqué en parallèle et étendu aux applications 3D. Finalement, le dernier chapitre résume le framework de la trempe industrielle numérique et est appliqué à des cas tests industriels, en se focalisant sur l'importance de la résolution du maillage utilisé.

7.1 Conclusion and Perspectives

This thesis was devoted for the capturing of fluid-solid interface and the creation of an anisotropic fitted mesh for the treatment of multi-component systems. With advances in technology and AI, single-component systems cease to exist, and simulation of multi-component systems is increasingly required. The need to solve such problems plays a huge impact in different fields like environment, health, biomedical, security, industrial and many more. The focus of this work was the development of a simple, fast and robust anisotropic adaptive body fitted meshes for CFD applications, and especially for the quenching process, our main application in the INFINITY chair:

- **Simple:** in terms of implementation and use
- **Fast:** automatic with minimum interference for the entire simulation
- **Robust:** able to handle any geometry no matter how complex
- **Anisotropic:** needed for high gradients and complex physics
- **Adaptive:** evolves dynamically with the interface as well as the solution
- **Body-fitted:** sharp interface

In chapter 1, we have introduced the quenching process and some of the challenges arising from the simulation of the process. As part of the INFINITY chair and in order to consolidate a unified multi-scale framework around understanding and simulating the quenching process, we proposed the development of a novel method that combines the immersed methods with fitting approaches to enhance the quenching or any multi-component environment.

In chapter 2, the numerical tools adopted for the modeling and simulation of fluid flows and conjugate heat transfers, developed by the CFL (Computing and Fluids) group at CEMEF, and used in the simulation of multi-phase and multi-component problems were described. Since during the resolution of the PDE's many numerical solutions might oscillate especially near sharp gradients, we resort to stabilization methods, such as the Variational Multiscale method for the Navier-Stokes equations and the SUPG technique for the convection-diffusion-reaction equations. The test cases considered reflect their accuracy in simulating physical phenomena.

In order to solve these equations analytically, space discretization of the physical domain into a finite number of elements is needed. Chapter 3 starts with an overview of different mesh generation techniques. Anisotropic mesh adaptation is then presented in details: the optimized metric map is obtained via an edge-based a posteriori error estimator and gradient recovery. The mesh adaptation presented can be applied on multiple variables and therefore, results in a high resolution and accuracy of the solution during the entire simulation. The anisotropic metric applied on the level-set function creates stretched elements near the interface and captur-

ing and separating the different components. However, the mesh generated is not conformal to the immersed geometry.

In Chapter 4, an overview of the existing methods to treat immersed multi-component applications have been described, presenting the different families and methods. It can be seen that the adaptation of the immersed methods towards a fitted mesh can be done numerically or geometrically. Our objective and main focus is to extend the Immersed Volume method, based on the immersed anisotropic mesh adaptation, towards a fitted mesh. To do so, a novel anisotropic fitted algorithm has been developed. In this chapter its 2D implementation is described followed by some numerical test cases for validation. In the vicinity of the interface the anisotropic cut elements are isolated to create a sub-mesh. On this sub-mesh an R-adaptation on the vertices closest to the interface on each element is then applied followed by a topological optimization, edge-swapping of the remaining cut edges, resulting in a conformal mesh at the interface. Finally, through some numerical illustrations, we show that the objective of creating a simple, fast and robust anisotropic adaptive body fitted mesh has been accomplished.

The algorithm is extended to 3D and parallel. Chapter 5 describes the parallel algorithm with the implementation of a new data structure for the sub-mesh created to ensure an efficient communication between the processors involved. For the three-dimensional implementation, 2D edge-swapping is replaced by 3D cutting of the cut elements: four different cases are described and generate new fitted elements at the interface.

In Chapter 6, we have validated the entire work presented: the numerical framework and the proposed anisotropic fitted algorithm. A new simulation setup desynchronizing the direct quenching process into two main parts: thermo-hydrodynamic and thermal simulation, is explained and then validated by applications for the quenching process, provided by our industrial partners, have also been treated.

The quenching simulation may also be coupled with solid phase transformation solver depicting the metallurgic transformation happening inside the solid during cooling. These transformations can potentially generate elasto-plastic deformations in the solid especially some displacements of the interface. In order to track during the simulation the interface and solid deformation, a full dynamic robust and adaptive algorithm is needed. The dynamic parallel implementation of the algorithm ensuring load balancing will allow re-distribution of the added elements in 3D while trying to maintain a balanced inter-process interface and minimizing the communication cost. This ongoing work will allow us to automatically and dynamically generate an anisotropic conformal fitted mesh that can handle more complex three-dimensional geometries.

With the advancement of technologies and artificial intelligence, coupling machine learning with this existing work has infinite potential and possibilities. For instance, during a quenching process, the position of the solid part inside the quenching tank plays an important role on the the cooling rate: deep reinforcement learning can test and predict different positions until the optimal one is achieved. This testing, and the change in states can be coupled with the anisotropic fitted algorithm to ensure an accurate and high resolution of the cooling process. Furthermore, the addition of mixers in the tank, their position and their effect on the cooling rate can also be tested.

The work presented is not limited to industrial applications but can be generalized to several areas and critical applications and thus benefit a large number of people in different branches. It improves the performance of simulations and solves them with more precision. As for the simulation of aneurysm rupture (Figure 7.1), the reduction of the calculation cost makes it possible to develop a decision support tool. Without the generation of suitable meshes, the implementation of such simulations for a specific patient would be too costly.

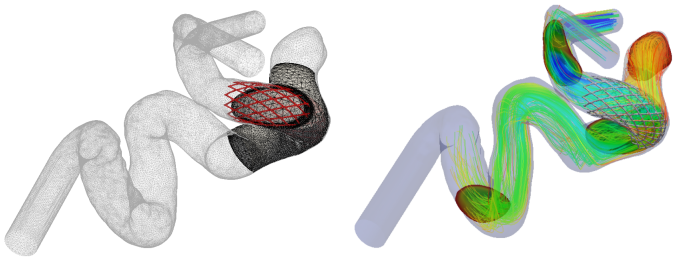


Figure 7.1: Simulation of blood flow in arteries through finely meshed aneurysms.

RÉSUMÉ

Le développement de méthodes efficaces pour simuler des systèmes multi-composants fait partie des défis d'ingénierie et reste un besoin pour les industriels, notamment dans le cas de l'interaction fluide-structure ou du transfert de chaleur conjugué. Le processus de trempe s'inscrit dans ce cadre puisqu'il a un impact direct sur la modification des propriétés mécaniques et physiques des pièces industrielles. De nombreuses formulations numériques de ce processus ont été développées, mais une grande imprécision subsiste, notamment en raison des hypothèses faites sur l'utilisation de géométries simples et d'environnements de trempe approximatifs. Pour le processus de trempe, plusieurs types de géométries de complexités différentes sont étudiés et analysés. Par conséquent, la génération de maillages pour des géométries aussi complexes reste un défi. En améliorant les méthodes pour la multi-physique, en particulier les couplages fluide-thermique et fluide-solide, le cadre mathématique global de cette thèse permettra de relever ce défi.
Dans ce travail, la méthode des volumes immergés est étendue : une nouvelle méthode de maillage adaptatif anisotrope adaptée à la géométrie est proposée. Sa simplicité et sa généralité lui permettent d'aborder des géométries complexes, et sa robustesse permet de traiter des problèmes physiques complexes. Deux itérations successives sont combinées : tout d'abord, la construction d'une métrique basée sur le gradient utilise les gradients de la fonction level-set de l'objet immergé pour générer un maillage anisotrope bien adapté. Elle est suivie d'une adaptation géométrique utilisant la R-adaptation et la permutation afin de créer un maillage ajusté net. Cette nouvelle approche permet d'atteindre la résolution géométrique locale souhaitée d'un maillage adapté au corps et d'obtenir la précision numérique nécessaire à l'interface grâce aux éléments anisotropes non structurés. La nouvelle approche permettra de résoudre les interactions fluide-solide et les problèmes de CFD, y compris les solutions de couche limite, de courbure et de gradient élevé, couvrant les applications parallèles 2D et 3D, et les problèmes pratiques du monde réel.

MOTS CLÉS

Méthodes d'immersion, Adaptation anisotrope, Maillage Conforme, Fonction Level-set, CFD

ABSTRACT

The development of efficient methods to simulate multi-components systems is among engineering challenges and still a need for industrials, especially in the case of fluid-structure interaction or conjugate heat transfer. The quenching process falls within this framework since it impacts directly the change in the mechanical and physical property of industrial parts. Many numerical formulations of this process have been developed, but considerable imprecision still exists, especially because of the assumptions made on the use of simple geometries and approximate quench environments. For the quenching process, several types of geometries with different complexities are studied and analyzed. Therefore, generating meshes for such complex geometries is a challenge. By improving methods for multi-physics, particularly fluid-thermal and fluid-solid couplings, the overall mathematical framework of this thesis will address this challenge.
In this work, the Immersed Volume Method is extended: a new anisotropic adaptive body-fitted mesh method is proposed. Its simplicity and generality allow it to tackle complex geometries, and its robustness allows one to deal with complex physical problems. Two successive iterations are combined: first, gradient-based metric construction uses the gradients of the level-set function of any immersed object to generate an anisotropic well-adapted mesh. It is followed by a geometric adaptation using R-adaptation and swapping in order to create a sharp fitted mesh. This new approach achieves the desired local geometry resolution of a body-fitted mesh and obtains the needed numerical accuracy at the interface due to the anisotropic unstructured elements. The new approach will allow to solve fluid-solid interactions and CFD problems including boundary layer, curvature, and high-gradient solutions, covering 2D and 3D parallel applications, and real-world practical problems.

KEYWORDS

Immersed methods, Anisotropic adaptation, Body-fitted mesh, Level-set function, CFD