Fictitious Boundary and Penalization Methods for Treatment of Rigid Objects in Incompressible Flows

Dissertation
zur Erlangung des Grades eines
Doktors der Naturwissenschften

Der Fakultät für Mathematik der
Technische Universität Dortmund
vorgelegt von

Dan Anca

November 2, 2013

Abstract

The Fictitious Boundary Method (FBM) and the Penalty Method (PM) for solving the incompressible Navier-Stokes equations modeling steady or unsteady incompressible flow around solid and rigid, non-deformable objects are presented and numerically analyzed and compared in this thesis. The proposed methods are finite element methods to simulate incompressible flows with small-scale time-(in)dependent geometrical details. The FBM, described and already validated in [1, 43, 48], is based on a finite element method background grid which covers the whole computational domain and is independent of the shape, number and size of any solid obstacle contained inside. The fluid part is computed by a multigrid finite element solver, while the behavior of the solid part is governed by the mechanics principles regarding motion and interactions of type fluid-solid, solid-solid or solid-wall collisions. A new treatment of imposing the Dirichlet boundary conditions for the case of immersed rigid boundary objects is proposed by using the penalization method as a more general framework then the FBM, but containing it as a special case. The new PM approach has a stronger mathematical background. In contrast to FBM, the PM does not imply a direct modification or artificial techniques over the matrix of the system of equations like the fictitious boundary method. A pairing of the penalty method with multigrid solvers is used, while the computational domain is fixed and needs no re-meshing during the simulations. However, the degree of geometrical details that the coarse mesh contains has an impact onto numerical results, a fact which will be investigated/clarified in this thesis. The presented method is a finite element method, easy to be incorporated into standard CFD codes, for simulating particulate flow or, in general, flows with immersed time-(in)dependent and complicated shaped objects.
The aim is to analyze and validate the penalty method and compare, qualitatively and quantitatively, with the already validated FBM regarding the aspects of accuracy of the solution, efficiency, robustness and behavior of the solvers. Different techniques to avoid the numerical difficulties that arise by using penalty method will be particularly described and analyzed.

keywords: FEM, Immersed Boundary Objects, Fictitious Boundary Method, Incompressible Flows, Monolithic Newton-Multigrid, Penalty Method

Acknowledgements

I am very grateful to my supervisor Prof. Dr. Stefan Turek, who accepted me years ago to be part of the Institute of Applied Mathematics, LSiii, Univeristy of Dortmund. I will never end to thank him for giving me this opportunity and for everything he taught to me into numerics, for his constant guidance and continuous support on my Ph.D study. I am deeply thankful for all the constructive discussions he had with me regarding my work, for helping me with good advices in critical moments when I thought there is no way out. I also thank him for the opportunity he gave me to be active in the student-teaching program, which made me a lot of joy and helped me understand much better the numerics. It was a great experience working here, learning and improving new skills, which will be always helping me in my future life.

I also acknowledge all the professors and tutors of the lectures I participated in. Particularly, I am very grateful to prof. Dr. Heribert Blum for enlighten me on the Finite Element Methods, for his professionalism and for his kindness of being there whenever I was needing him. I thank in the same manner to Prof. Dr. Dmitri Kuzmin for his wonderful teaching in Computational Fluid Dynamics, which helped me in my beginnings understanding this part of applied mathematics and inspired me during my study.

Acknowledgements have to be brought to all my colleagues for developing an harmonious environment for the research, making all to be easier. Special thanks I address to Dr. Michael Köster for spending precious time with me helping to understand and develop the software part. I will also not forget and I am deeply grateful for the important and direct impact of Dipl-Inform. (soon Dr.) Raphael Münster over my work, for all the discussions he had with me regarding the purpose of my study on both theoretical and programming level. I am also grateful to Dr.

Abderrahim Ouazzi for his important advices, especially on the final part of my study, who were very precious for validating the results.

A particular thank to all my friends, which I do not cite not to forget someone. I am very grateful for their incredible support of all kinds during all these years, for encouraging me all the time to follow this course. They were valuable help and I am very glad to have their friendship.

Last but not least, I would like to thank to all my family and especially to my beloved wife Paula Anca who always put positive pressure on me and gave me all her support to do my work. Her own motivation on this matter was inspiring and gave me in all the moments enough resources to continue and finish my work. She is a model for me on how should one fight for fulfilling his aims. I thank to my parents for letting me choose my own way in studying mathematics and physics and for always being there for me. I thanks to my two brothers for supporting and understanding me.

Dan Anca, on 23 September 2013

Contents

Chapter 1

Introduction

1.1 Overview

Any living being has/had contact with the main constitutive of this planet: water. Thus, most of them remain with a low level status of knowledge about water, or fluids in general, and only few go with the involvement beyond this common label, showing eagerness about researching and understanding the natural phenomena in which flows are part. However, over the past century, many studies were dedicated to flow processes, having the same main goals of modeling, describing and proving/showing theoretically and/or practically the behavior of different kinds of fluid structures that surround us. Fluid mechanics is the study on fluids and forces that act on them. Fluid dynamics studies the interactions between fluid and other kind of structures (i.e solids) and also the effects of the forces that determine fluid motions. Physically, fluids might be categorized into 2 major groups: Newtonian and non-Newtonian. All fluids with a linear dependent stress tensor to the strain rate are called Newtonian after Isaac Newton. On the other hand, fluids whose viscosity (the dimension of resistance to deformation to other forces) is dependent on the shear rate, are known as non-Newtonian. As an exception, there can still be fluids with a shear-independent viscosity, but nonetheless behave themselves in a non-Newtonian way. Further on, by changing the criterion of classification and based on the non-dimensional Reynolds number (Re), fluids can be laminar (low Re values), transient (medium Re values) and turbulent (high Re values). This thesis focuses on studying the case of laminar

and transient incompressible Newtonian fluids, proposing a new approach to solve, compute and simulate flows around obstacle configurations. Having a theoretical support in understanding fluids, problems can be probably easy to solve. Thus, fluid problems are in general very complex, having lots of factors which can increase the degree of complexity and therefore it is almost impossible to find analytical solutions. In this direction, over past decades, a new branch has been developed for solving complex-configuration fluid problems. Computational Fluid Dynamics, shortly *CFD*, uses the power of computers to perform calculations and analyze problems that involve fluids. By using numerical methods and algorithms, *CFD* proved its importance in this domain helping in validating theoretical existence and uniqueness assumptions over solutions.

There are many configurations that include a fluid part: Flows around obstacles, fluid-structure interaction, multiphase flows with chemical reactions, bubble dynamics, melting and solidifications, etc. Particulate flow is also one example and presumes interaction between two different phase state components: fluid and rigid solid. More precisely, flow configurations with complex geometrical details and/or moving rigid solid interfaces and boundaries are considered as particulate flow. They gained big importance due to the variety of applications in physics, chemistry, engineering, etc.

How to study such problems? Which techniques will be easier, accurate or efficient with respect to the problem? Researchers studied different methods presenting advantages and disadvantages. Generally, most methods can be divided into two major families: the *Arbitrary Lagrange-Euler* (ALE) and the *fictitious domain methods*. Hu, Joseph and Crochet [22], [21], Maury and Glowinski [32] are pioneers of the Lagrangian approach which is based on a mesh that follows very faithfully the motion of the boundaries. A kind of fictitious domain method is the Eulerian approach of Distributed Lagrange Multipliers (DLM). One fixed finite element particle-independent mesh to compute velocity and pressure and Lagrange multipliers to enforce boundary conditions and/or solid movements are the components of this method. Proposed by Glowinski, Joseph, Pan, Hesla and Perieux [11], [12], [10], the method is very powerful and not restricted to simulate only particulate flow.

Assume, we want to solve the following boundary value problem:

$$\begin{cases} A(u) = f, u \in \omega \\ B(u) = g, u \in \gamma \end{cases} \tag{1.1}$$

where $\omega \in \mathbb{R}^d$ is a bounded domain and $\gamma = \partial\omega$ is its boundary. The differential operators A, B are acting on ω and γ, while f, g are given functions that act on the same domains. Taking into account the complexity of ω, one can use immediately a finite difference method, but it will encounter numerous difficulties for a more complex geometry. Therefore, an alternative can be the finite element method. In order to combine the simplicity and advantages which finite difference methods provide and use a finite element approach to solve such problems, one can use fictitious domain methods, also denoted as embedding method. The avatar of fictitious domain was firstly introduced by Saul'ev [1962, 1963], but it was used years before by Hyman [1952]. The principle of such methods is a very simple one. Instead of solving the problem on the given domain ω, who's shape might be very complex, one can extend it to a larger domain with a simple standard shape Ω. Obviously, the mathematical model has to be changed accordingly such that the solution of the fictitious domain is also a solution of the initial boundary problem. Exactly, the problem will be described as in (1.2)

$$\begin{cases} \tilde{A}(\tilde{u}) = \tilde{f}, \tilde{u} \in \Omega \\ \tilde{B}(\tilde{u}) = \tilde{g}, \tilde{u} \in \Gamma \end{cases} \tag{1.2}$$

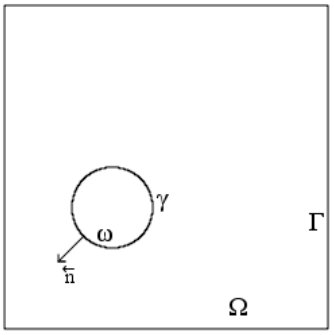


Figure 1.1: Embedding of ω in Ω

where Ω is a larger, simpler domain that includes ω, with the frontier Γ like in figure Fig. (1.1). As mentioned, the new differential operators $\tilde{A}, \tilde{B}$ and the given

functions $\tilde{f}, \tilde{g}$ are chosen such that the solution $\tilde{u}$ of problem 1.2 is easy to compute and nevertheless it satisfies the condition of being also a solution in ω or, at least, the error in Ω is very small: $||\tilde{u} - u)||_{\Omega} \ll \epsilon, \quad \epsilon$ small. By doing this extension to the bigger domain, at least two advantages arise. One will be the possibility of using large domains with corresponding simple and regular meshes which will allow at the same time the usage of fast solution methods. The second one is in the case of time-dependent problems, Ω can still be chosen time-independent even if the original domain ω is a time-dependent and no re-meshing or projection methods as in *ALE* are required. Such a fixed mesh technique is also the nominated *DLM* method which enforces on the boundary a constraint by using Lagrange multipliers. The method was applied also for nonlinear time dependent problems such as the Navier Stokes equations. On the other hand, avoiding the usage of the multipliers, Peskin proposed a non-Lagrangian technique known as immersed boundary method. This method has many application for incompressible viscous flows with elastic moving boundaries.

Many investigations were finding solutions and analyzing them from different points of view. Although, the first question for such problems should be the existence of a solution. Penalty based fictitious boundary methods were used in general to prove the existence of solutions for *PDE*. Many kind of penalization approaches of *PDE* systems are possible. For instance, to solve the incompressible Navier-Stokes equations, a L^2 penalization, inducing a Darcy equation, or a H^1 penalization, inducing a Brinkman equation in the body, are two of the options. Also in the case of penalization methods, the aim was to avoid body-fitted unstructured meshes such that fast and efficient methods on Cartesian meshes may be used. Peskin [34], [35] proposed penalization of the velocity in the momentum equation of the incompressible Navier-Stokes equations. Arquis and Caltagirone [3] proposed a penalization of velocity on the volume of the body. Goldstein, Handler, Sirovich [14] or Saiki, Biringen [38] promoted other kinds of penalization. The presence of a penalty parameter, which is rather very small ($\varepsilon \ll 1$), will obviously lead to numerical difficulties and accuracy problems. Direct question will be if the penalization methods can reproduce equivalent results as fictitious domain methods. This thesis is dedicated to analyze a velocity penalization technique with the aim of setting it as a generalization method for the fictitious boundary method. A general method on solving incompressible (un)steady

Navier-Stokes, with a strong mathematical support is investigated. The computational support is provided by the CFD software FEATFLOW (*www.featflow.de*). Aspects like capturing the interface, accuracy of solution, efficiency of solvers will be discussed inside this thesis.

1.2 Motivation and contribution

The purpose of this study is to implement a velocity penalty method for solving flow around objects problems in a finite element framework for two dimensional cases. Turek and Wan proposed an, already validated, fictitious boundary method *FBM* to simulate and solve such problems. The proposed method clearly possesses many advantages with the benefit of good and accurate results. The idea of this study is to try a generalization, with a much better mathematical support, of the *FBM*. We start with the assumption that *FBM* is actually a special case of the proposed velocity penalty method and, therefore, the new method should be more general and should reproduce the results of *FBM* in specific situations and for particularly selected parameters. Further we try to validate the penalty method for benchmark simulations and to overcome the numerical difficulties that arise with it, such that it remains efficient, accurate and robust. Based on a better mathematical support and benefit of the finite element framework, this approach should also allow the use of fast solvers in order to solve the fluid-solid object interactions problems. Fictitious boundary methods are based on direct manipulations of the matrix system, filtering techniques, in order to impose the boundary presence into the fluid domain. The penalty method uses another approach, which performs modifications of the matrix system mathematically and does not hard-way interfere with local values of it. Introducing a new penalty term in the governing equations, a new matrix will be assembled and added into the system matrix. The presence of the solid object in the fluid domain will be notified by the penalty parameter which is in general very small ($\varepsilon \ll 1$). Using finite element methods, a Dirac representation of the penalty characteristic function $\chi(x)$, also called mask function, will realize the selection of all elements and degrees of freedom *DOFs* inside the computational domain being the solid part. The penalty term is of the following type:

$$\frac{1}{\varepsilon} \cdot \chi(x) \cdot (u - \overline{u}) \tag{1.3}$$

where $\lambda = \frac{1}{\varepsilon} = \frac{1}{10^{-n}}$, $n >> 1$, with ε the penalty parameter, u is the fluid part velocity, $\overline{u}$ the solid part velocity and $\chi(x)$ is the penalty characteristic function with the definition (1.4). This is a discontinuous representation, but based on distance function, also a smooth representation of the mask function can be used, such that the $\chi(\cdot)$ function takes the value of one for the point inside the object and will be set to equal null otherwise. An example of a smooth representation based on the distance function is given in (1.5), where η is a free parameter which determines how steep should be the function and $d(x)$ is the signed distance function to the interface of the object:

$$\chi(x) = \begin{cases} 1, & \text{if } x \in \Omega_s \\ 0, & \text{if } x \notin \Omega_s \end{cases} \tag{1.4}$$

$$\chi(x) = \frac{1}{2}\tanh(\eta * d(x)) - \frac{1}{2}, x \in \Omega \tag{1.5}$$

Ω_s denotes here the domain of the penalized object and Ω the entire domain.

Using fast solvers to get the solution is another point of interest. The employment of such solvers depends directly on the selected grid for the finite elements. If a structured mesh is used, then fast solvers can be applied without arising problems to solve the system. For a moving penalty object, to set up a structured mesh is almost impossible. An investigation of this matter is also presented in this thesis, pointing out that in the case of the penalty method, Cartesian meshes are not optimal and therefore, one needs to start with a body-fitted mesh. For the case of a moving object grid, deformation techniques [15] may be considered.

Is well-known that penalty methods are not very accurate in the vicinity of the boundary object. Different ways of implementing penalty methods are presented to illustrate how problems due to the caption of the interface may be treated and see the effects regarding the accuracy of the solutions. Another issue are the very small values of the ε-penalty parameter which lead to numerical problems. Thus, treatment of this aspect is required such that the solver will converge to accurate solutions.

Beside the complete investigations and analysis of the penalty method, the validated fictitious boundary method proposed by Turek and Wan [47, 48], [1, 43] will be also presented inside this thesis, which is actually a special case of the penalty method as it will be outpointed inside this study. Technical implementation details and benchmark/application results will be shown in order to underline the similitudes and differences between penalty method approaches and compare efficiency, accuracy and the practicality for appliance for the proposed methods. The novelty consists in a new treatment of the assembling process of the penalty matrix by special handling of the quadrature formula or by the special technique of *HRZ* lumping which fits very well together with our finite element approach.

This thesis has the following structure. After the introduction, Chapter 2 is dedicated to the governing equations of a Newtonian fluid and modeling the problem. General theory for fluid dynamics, Newtonian fluids, Navier-Stokes model and boundary conditions are some of the topics mentioned in this chapter. Chapter 3 is more a general survey over finite element methods, time and space discretization techniques and the derivation of the discretized problems to be solved. Iterative numerical solvers have to be used for solving these problems and therefore they will be shortly introduced and presented in Chapter 4. Chapter 5 focuses on the proposed methods, namely fictitious boundary method *FBM* and penalty method *PM*. A relevant comparison between these methods as well as the behavior of the penalty method *PM* for specific selections of parameters will be contained in this chapter. The study focuses more on the implementation and investigation of the penalty method, therefore different ways of implementing penalty methods will be presented with the aim of validation and to give to the penalty method an attribute of a general method to solve fluid-(rigid)solid objects problems. The flow around cylinder benchmark is used to show that the method provides an accurate solution for a medium value of Reynold number. Later on, for dynamical reasons and possibility of application in particulate flow, drag and lift forces calculations are presented. To prove the capabilities of the proposed *PM* for simulating flow with complex moving objects, we show a simple benchmark simulation of an oscillating cylinder in a channel in Chapter 6. Finally, the conclusions are resumed in Chapter 7.

Chapter 2

Modeling and governing equations

2.1 General aspects of Newtonian flows

Newtonian fluids are characterized by a linear stress relation to the strain rate. For example, air and water are considered to be Newtonian flows. Like for any fluid, the main general laws of mechanics are valid also for these flows, namely conservation of mass and momentum. Starting with these 2 physical principles, a mathematical model to analyze and study the mechanics and behavior of fluids was developed. This model is known in the literature as Navier-Stokes equations and comes with different forms depending on the kind of fluid, respectively on the stress tensor representation.

A measure to classify flows is the so called Reynolds number, denoted by Re and defined as

$$Re = \frac{U \cdot L}{\nu}, \tag{2.1}$$

where U and L are the characteristic length and velocity of the flow and ν is the kinematic viscosity. The way of choosing L and U depends on the fluid problem and fluid domain. Usually, the characteristic length is the diameter of the fluid domain, but it can be dependent to any other length related to the domain. For the characteristic velocity, the forces that act on the flow determine its selection. In most of the cases, this referential value for the velocity is set to be the velocity which is applied at the boundary of the fluid domain, often at the inflow. Having as

criteria the Reynolds number, the Newtonian flows are classified into 3 major groups, depending on the values of Re. We talk about highly viscous flows, described by a small velocity with respect to the length scale if $Re \ll 1$. Convection dominated or even turbulent flows are all flows for which $Re \gg 1$. Finally, the flow is called transient if Re is represented by a medium value. From the numerical point of view, the Reynolds number has also an importance in describing the Navier-Stokes equations in a dimensionless way, which we will see within this chapter.

In order to derive the Navier-Stokes equations model, one needs to master mathematical notions like integration by parts, Ostrogradsky's divergence theorem (2.2), Helmholtz decomposition of vector fields, gradient, divergence, curl, etc. Starting with the laws that govern fluid mechanics, i.e. conservation of mass and momentum, the Navier-Stokes equations can be derived in conservative, non-conservative as also in a variational or weak formulation. In literature there are many derivations of Navier-Stokes equations like for example Prager [36], Batchelor [4], Landau and Lifschitz [27] and many others presented. According to Marion and Temam [31], the Navier-Stokes model for incompressible fluids was described for the first time by Leray [28], [30], [29] as a velocity evolution equations, omitting the presence of the pressure. The vector field of velocity, through its evolution, determines in a unique way the pressure and therefore there is no need of an extra condition for the pressure. The Navier-Stokes equations is the most used model to study the behavior and motion of fluids and it was, indeed, validated by numerous analytical, numerical and experimental studies.

In the following chapters, several common notations are used in the process of derivation of the Navier-Stokes equations. A three dimensional domain $\Omega \in \mathbb{R}^2$, occupied by the fluid at time $t \geq 0$, whose points are generically located in the bidimensional case by $\mathbf{x} = (x_1, x_2)$, is considered. The boundary of Ω is denoted by Γ and the elementary volume has the representation $dx = dx_1 dx_2$. The developing approach to build up the Navier-Stokes model is based on Marion and Temam [31] and Glowinski [10].

In the following part, there are some of the mathematical notions often used to build up the (non)-conservative and variational Navier-Stokes equations.

- Gauß - Ostrogradsky's Divergence Theorem

$$\int_V \nabla \cdot \mathbf{F}\, dV = \int_S \mathbf{F} \cdot \mathbf{n}\, dS \tag{2.2}$$

V - an arbitrary volume bounded by the surface S;
$\mathbf{n}$ - the unit normal vector that points outward from $S := \partial V$;
$\nabla\cdot$ - divergence operator $\nabla \cdot \mathbf{F} := \frac{\partial F_1}{\partial x_1} + \frac{\partial F_2}{\partial x_2}$;

- Total time derivative (material derivative) in flow direction

$$\frac{d\mathbf{F}}{dt} = \frac{\partial \mathbf{F}}{\partial t} + \nabla \mathbf{F} \frac{d\mathbf{x}}{dt} = \frac{\partial \mathbf{F}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{F} \tag{2.3}$$

$\frac{d\mathbf{x}}{dt} = \mathbf{u}$ - the path $\mathbf{x}(t)$ follows the fluid described by its velocity $\mathbf{u}$

- Divergence operator of a vector

$$\nabla \cdot \mathbf{u} = \sum_{i=1}^{2} \frac{\partial u_i}{\partial x_i} \tag{2.4}$$

- Laplacian operator of velocity vector

$$\Delta \mathbf{u} = \nabla(\nabla \cdot \mathbf{u}) - \nabla \times (\nabla \times \mathbf{u}) = \nabla \cdot (\nabla \mathbf{u}) \tag{2.5}$$

- Green's formula

$$\int_\Omega u\, \partial_i v\, dx = -\int_\Omega \partial_i u\, v\, dx + \int_\Gamma u\, v\, n\, d\gamma \tag{2.6}$$

2.2 Navier-Stokes equation for incompressible flow

Continuity equation

Conservation of mass is one of the physical principles that applies to any fluid. Consider that we have an arbitrary fluid domain $\omega_0 \in \Omega$ with the boundary $\gamma_0 := \partial\omega_0$ with a space and time dependent mass density $\rho = \rho(\mathbf{x}, t)$. The mass of such a domain writes:

$$M = \int_{\omega_0} \rho \, dx \tag{2.7}$$

The formulation of the mass conservation, or mass balance equation, sustains that the decrease of the mass per unit time is equal to the total outflowed mass through the boundary γ_0 of the considered fluid domain. Thus, we can write mathematically

$$-\frac{d}{dt}\int_{\omega_0} \rho dx = \int_{\gamma_0} \rho \, \mathbf{u} \cdot \mathbf{n} \, d\gamma_0, \tag{2.8}$$

where $\mathbf{u} = (u_1, u_2, u_3)$, with the Eulerian representation $\mathbf{u} = \mathbf{u}(\mathbf{x}, t)$, is the velocity of the fluid, $\mathbf{n}$ is the outward pointing unit normal vector of the boundary γ_0 and $d\gamma_0$ is the elementary surface measure. Applying (2.2) in the right hand side of (2.8) and taking into account that the total time derivative of the mass density is equal with the partial time derivative, it results in the new global formulated equation:

$$\int_{\omega_0} \frac{\partial \rho}{\partial t} + \nabla \cdot \rho \mathbf{u} \, dx = 0 \tag{2.9}$$

Equation (2.9) can be written also in a local form due to the arbitrary selection of the domain ω_0. The local representation (2.10) makes no use of integration, being a simple differential equation:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \rho \mathbf{u} = 0 \tag{2.10}$$

Thus, (2.10) represents the first equation of Navier-Stokes model for a Newtonian fluid.

Momentum equation

The momentum equations is a consequence of the second principle of mechanics, also known as Newton's second law. It is represented in (2.11) in terms of the Cauchy tensor $\sigma := (\sigma_{ij}),\ i, j = 1, 2$:

$$\rho \, \mathbf{a} = \nabla \sigma + \mathbf{F}, \tag{2.11}$$

with $\mathbf{a} = (a_1, a_2)$ the acceleration of the fluid element, $\mathbf{F} = (f_1, f_2)$ volume forces that

are applied or act on it and $\sigma = \sigma(\mathbf{x}, t)$ the symmetric Cauchy stress tensor at point $\mathbf{x}$ at time t. The velocity of the fluid element is defined as the material derivative of the space $\mathbf{u} = \frac{d\mathbf{x}}{dt}$, while the acceleration is the material derivative of the velocity $\mathbf{a} = \frac{d\mathbf{u}}{dt}$. According to (2.3), then the acceleration writes:

$$\mathbf{a} = \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} \tag{2.12}$$

and in consequence the nonlinear term $\mathbf{u} \cdot \nabla \mathbf{u}$ comes into the equation from pure mathematical reasons, not physical, by writing the acceleration in kinematic arguments. This term is also called inertial term.

The stress tensor possesses different representations, depending on the kind of fluid. It is a symmetric tensor, generally written as in (2.13), with the following notations: $\sigma_x, \sigma_y, \sigma_z$ the normal stresses and $\sigma_{xy} = \sigma_{yx}$ the shear stresses.

$$\sigma_{ij} = \begin{bmatrix} \sigma_{11} & \sigma_{12} \\ \sigma_{21} & \sigma_{22} \end{bmatrix} = \begin{bmatrix} \sigma_x & \sigma_{xy} \\ \sigma_{yx} & \sigma_y \end{bmatrix} \tag{2.13}$$

For example, in the case of a Newtonian fluid, the stress tensor is represented in terms of velocity vector and pressure $p = p(\mathbf{x}, t)$, which is a new physical variable. One common representation is the kinematical one (2.14):

$$\sigma_{ij} = 2\mu D(\mathbf{u}) + (\lambda \nabla \cdot \mathbf{u} - p)\delta_{ij} \tag{2.14}$$

μ and λ coefficients are the dynamic shear viscosity (μ) and dilatation viscosity (3λ), while $D(\mathbf{u})$ is the deformation rate tensor, also symmetric and defined as in (2.15):

$$D(\mathbf{u}) = \frac{1}{2}(\nabla \mathbf{u} + (\nabla \mathbf{u})^T) \tag{2.15}$$

Putting all together, (2.11),(2.12), (2.14) and (2.15), one obtains the momentum equation for compressible fluids in Eulerian representation as Landau and Lifschitz [26] presented, in local form:

$$\rho \frac{\partial \mathbf{u}}{\partial t} + \rho \mathbf{u} \cdot \nabla \mathbf{u} = \mu \Delta \mathbf{u} + (\mu + \lambda) \nabla \nabla \cdot \mathbf{u} - \nabla p + \mathbf{F} \tag{2.16}$$

Thus, the set of PDE equations (2.10) and (2.16) builds up the well known Navier-Stokes equations model in a general representation for the case of compressible flows:

$$\begin{cases} \dfrac{\partial \rho}{\partial t} + \nabla \cdot \rho \mathbf{u} = 0 \\ \\ \rho \dfrac{\partial \mathbf{u}}{\partial t} + \rho \mathbf{u} \cdot \nabla \mathbf{u} = \mu \Delta \mathbf{u} + (\mu + \lambda) \nabla \nabla \cdot \mathbf{u} - \nabla p + \mathbf{F} \end{cases} \tag{2.17}$$

Incompressibility condition

Suppose we have an incompressible fluid. Physically, it is characterized by a much smaller velocity then the velocity of sound in the water. Any element of fluid moving with the flow has the same volume in time, meaning that the mass density of the fluid is invariant in time. Also, incompressible flow has a constant dynamic viscosity $\mu = const$. One can proof by a Lagrangian technique that for any incompressible fluid the following equation holds:

$$\frac{dM}{dt} = \frac{d}{dt} \int_{\omega_0} dx = \int_{\omega_0} \nabla \cdot \mathbf{u} \, dx = 0, \tag{2.18}$$

which leads to the incompressibility condition for a fluid. Since ω_0 is an arbitrary fluid domain, the incompressibility condition mathematically reads:

$$\nabla \cdot \mathbf{u} = 0 \tag{2.19}$$

Navier-Stokes equations for incompressible fluids

Simply bringing together the equations (2.10) and (2.16) and applying the incompressibility condition, the Navier-Stokes equations for incompressible inhomogeneous fluids are described by one of the following systems:

$$\begin{cases} \dfrac{\partial \rho}{\partial t} + \nabla \cdot \rho \mathbf{u} = 0 \\ \\ \rho \dfrac{\partial \mathbf{u}}{\partial t} + \rho \mathbf{u} \cdot \nabla \mathbf{u} = \mu \Delta \mathbf{u} - \nabla p + \mathbf{F} \end{cases} \tag{2.20}$$

or

$$\begin{cases} \nabla \cdot \mathbf{u} = 0 \\ \rho \dfrac{\partial \mathbf{u}}{\partial t} + \rho \mathbf{u} \cdot \nabla \mathbf{u} = \mu \Delta \mathbf{u} - \nabla p + \mathbf{F}, \end{cases} \tag{2.21}$$

keeping in mind that the continuity equation implies the incompressibility condition (not reciprocal). The second system of equations leads to the Navier-Stokes model for incompressible and homogeneous fluids described by Batchelor [5]. A homogeneous fluid has a constant mass density ρ, which does not change in space and time. Therefore, the following equations are true:

$$\begin{cases} \dfrac{\partial \rho}{\partial t} = 0 \\ \nabla \cdot (\rho \mathbf{u}) = (\mathbf{u} \cdot \nabla)\rho = 0 \end{cases} \tag{2.22}$$

Dividing (2.21) by the value of density and introducing the kinematic viscosity $\nu := \frac{\mu}{\rho}$, the kinematic pressure $p := \frac{p}{\rho}$ and the mass density of the body forces $\mathbf{f} = \frac{\mathbf{F}}{\rho}$, then a new system of equations derives for incompressible homogeneous fluids, presented in a nonconservative for the first time by Batchelor [5]:

$$\begin{cases} \nabla \cdot \mathbf{u} = 0 \\ \dfrac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} - \nu \Delta \mathbf{u} + \nabla p = \mathbf{f} \end{cases} \tag{2.23}$$

It is now possible to formulate a dimensionless model for incompressible homogeneous fluids which is convenient for physical discussions. To do this, one considers Ω a domain filled by the fluid and t the time variable which is usually positive. Choose a referential length L and time T for the flow, redefine all variables from Navier-Stokes equations with respect to L and T:

$$x = Lx',\ t = \frac{L}{U}t',\ p = Pp',\ u = Uu' \text{ and } f = \frac{U^2}{L^2}f'$$

where $U = \frac{L}{T}$ and $P = \rho U^2$ are the referential velocity and pressure. Now using the Reynolds number definition (2.1), the resulting system:

$$\begin{cases} \nabla \cdot \mathbf{u} = 0, \quad \text{in } \Omega \times (0,T) \\ \\ \dfrac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} - \dfrac{1}{Re} \Delta \mathbf{u} + \nabla p = \mathbf{f}, \quad \text{in } \Omega \times (0,T) \end{cases} \tag{2.24}$$

represents the non-dimensional Navier-Stokes model for incompressible homogeneous fluids in nonconservative manner. By considering mass density as constant in time and space, the number of unknowns reduces from 3 to only 2: velocity and pressure. However, even if the Navier-Stokes equations is a 2x2 equation system, it is not sufficient to obtain an unique solution since it has infinity of integral solutions. Thus, the need of boundary conditions is mandatory to restrict to a solution and to define the flow.

2.3 Initial and boundary conditions

There are many possible ways to impose boundary conditions and most of them are for the velocity component. The fluid fills a domain Ω bounded by the surface Γ. One direct condition for the flow is an initial distribution for velocity like (2.25):

$$\mathbf{u}(\mathbf{x}, 0) = u_0(\mathbf{x}), \quad \text{on } \Omega. \tag{2.25}$$

This is equivalent with an incompressibility condition $\nabla \cdot u_0 = 0$ at time $t = 0$. If we assume that the boundary of the fluid domain moves and its motion is described by a function g, then we may consider the nonslip boundary condition (2.26) for the fluid, which is a Dirichlet type boundary condition:

$$\mathbf{u} = g, \quad \text{on } \Gamma \times (0,T), \ \Gamma = \partial\Omega. \tag{2.26}$$

If the fluid is at rest, then $g = 0$. Nevertheless, if Ω is a bounded domain, then the Dirichlet boundary condition can be rewritten as

$$\int_\Gamma g(t) \cdot \mathbf{n} \, ds = \int_\Omega \nabla \cdot \mathbf{u}(t) \, dx = 0. \tag{2.27}$$

Furthermore, one can impose a Neumann type boundary condition which is a restriction of the stress tensor. Usually, such a condition is imposed only on a part of the boundary of the domain. One example is the no stress boundary condition know as do-nothing boundary:

$$\sigma \cdot \mathbf{n} = 0, \quad \text{on } \Gamma_p \subset \Gamma. \tag{2.28}$$

But the fluid problem can be formulated with complicated boundary conditions like for example mixed boundary conditions. An example is (2.29):

$$\begin{cases} \mathbf{u} = g_0, & \text{on } \Gamma_0 \times (0,T) \\ \sigma \cdot \mathbf{n} = g_1, & \text{on } \Gamma_1 \times (0,T) \end{cases} \tag{2.29}$$

and illustrated in figure (2.1). It is to be specified that in the case of mixed boundary conditions (2.29), the component parts of the boundary have no common points and together they reconstruct the entire boundary: $\Gamma_1 \cap \Gamma_2 = \emptyset$ and $\Gamma_1 \cup \Gamma_2 = \Gamma$. No point of the boundary can have simultaneously Dirichlet and Neumann boundary conditions and, also, no point is conditioned free as the corresponding figure shows.

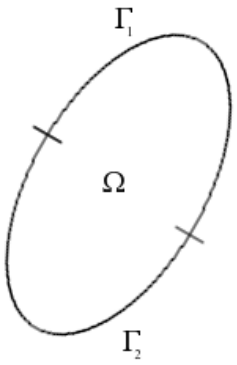


Figure 2.1: Dirichlet and Neumann mixed boundary conditions

Another setting can be a separation of the boundary into three component boundaries, suggestive named: inlet, wall and outlet. Each is defined by the normal component of the velocity like for instance:

$$\begin{cases} \Gamma_i = \{\mathbf{x} \in \Gamma / \ \mathbf{u} \cdot \mathbf{n} < 0\} \\ \Gamma_w = \{\mathbf{x} \in \Gamma / \ \mathbf{u} \cdot \mathbf{n} = 0\} \\ \Gamma_o = \{\mathbf{x} \in \Gamma / \ \mathbf{u} \cdot \mathbf{n} > 0\} \end{cases} \tag{2.30}$$

An immediate example of mixed boundary conditions is also:

$$\begin{cases} \mathbf{u} = \mathbf{v}, \quad \text{on } \Gamma_D \times (0,T) \\ \sigma \cdot \mathbf{n} = 0, \quad \text{on } \Gamma_N \times (0,T) \end{cases} \tag{2.31}$$

with $\Gamma = \Gamma_D \cup \Gamma_N = \Gamma_i \cup \Gamma_w \cup \Gamma_o$. For such conditions, one has to take into account two other consequences. On the wall boundaries, the following compatibility condition has to be fulfilled $\mathbf{n} \cdot \mathbf{v} = \mathbf{n} \cdot \mathbf{u}$. Nevertheless, there is a solvability condition also to be accomplished. In the case that the Dirichlet boundary is identical with the boundary itself $\Gamma_D = \Gamma$, then the inflow has to be equal with the outflow $\int_\Gamma \mathbf{n} \cdot \mathbf{v} \ ds = 0$. On the other hand, if the Neumann condition is set along the complete boundary Γ then a pressure condition is necessary for the uniqueness of the solution. But, since there is no general boundary condition for the pressure, to overcome this, one can fix the pressure point-wise or can set the mean value of pressure to zero:

$$p(\mathbf{x}_0) = p_0 \text{ or } \int_\Omega p \ dx = 0. \tag{2.32}$$

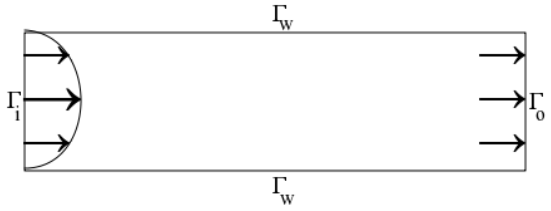


Figure 2.2: Inlet, Wall and Outlet boundary conditions

The fluid problem can become very complicated when it comes to set different complicated boundary conditions. This thesis reviewed only few classical cases of boundary conditions and more over boundary conditions can be read in [18]. A common technique to solve the Navier-Stokes problem endowed with any type of boundary conditions is to make use of a weak or variational formulation of the problem. This supposes the introduction to Sobolev spaces. Such a formulation is very useful for a

finite element treatment of the problem. This aspect will be presented in the following chapter.

Chapter 3

Finite element discretisation

Solving analytically mathematical models for fluids to get exact solutions is almost impossible in the general case. Therefore, they need to be treated in a numerical approach. There are different numerical methods to obtain approximation solutions for model problems that describe the behavior of the fluids. A common approach is to make use of the variational formulation of the problem, discretize it and use the finite element method for obtaining local solutions, respectively global solutions. This chapter is dedicated to the presentation of such a process, pointing out the construction of the weak formulation of incompressible Navier-Stokes equations, the vectorial spaces where the solution should be defined, examples of such spaces in finite element method theory and, finally, the corresponding discrete problem which has to be solved. The finite element method *FEM* knew a large and vast development at the beginning of 1970, when it became a new area in the numerical analysis of applied mathematics. A false idea over *FEM* occurred when it was considered to be a consequence of the growing need to solve *PDE*s, weak equations of boundary value problems and defining Sobolev spaces, but it was not the case. However, *FEM* has proved to be a powerful tool in solving these problems, too.

We consider in the following part a *PDE* system of equations to be solved by using the *FEM*. For such a given *PDE* one can construct the minimization problem with the property that it is simpler to be numerically solved. The foundation principle of constructing minimization problems is the energy principle. By definition, the energy is of type (3.1):

$$J(v) = \frac{1}{2}a(\mathbf{v}, \mathbf{v}) - l(\mathbf{v}), \tag{3.1}$$

where $a(\cdot, \cdot)$ and $l(\cdot)$ represent bilinear, respectively linear, forms. An abstract minimization problems is formulated as follows:

Find $\boldsymbol{u} \in U$, $U \subset V$, V *a Hilbert space endowed with a corresponding norm, such that the energy is minimal:*

$$J(\boldsymbol{u}) = \inf_{\boldsymbol{v} \in U} J(\boldsymbol{v}). \tag{3.2}$$

It is necessary to define suitable vectorial spaces and bilinear and/or linear forms with proper attributes such that the existence and uniqueness of a solution is ensured. The candidates for such spaces are only complete spaces in which the energy is well defined everywhere, like for example the Sobolev spaces. Nevertheless, the bilinear and/or linear forms have to be continuous on the selected spaces. The minimization problems are very useful tools to solve fluid problems due to the strong theory regarding the existence and uniqueness of the solution. Lax-Milgram Lemma is the main tool in this sense and before it is stated, some additional theoretical results regarding minimization problems have to be mentioned. They are formulated beneath the following theorems without any proof. The reader is forwarded to consult Ciarlet [7] for the detailed corresponding theory.

Theorem 1. [1] *Let the following assumption be true:*

i) V *- a complete space*

ii) $U \subset V$ *- closed and convex*

iii) $a(\cdot, \cdot)$ *- symmetric*

iv) $a(\cdot, \cdot)$ *-* V *-elliptic (* $\exists\ \alpha > 0,\ \alpha||\boldsymbol{v}||^2 \leq a(\boldsymbol{v}, \boldsymbol{v}),\ \forall\ \boldsymbol{v} \in V$ *)*

Then the minimization problem (3.2) has one and only one solution.

[1] Ciarlet, P.G. and Lions, J.L., Editors: *"Handbook of numerical analysis", vol. IX, 2003, page 24-25*

Theorem 2. [2] *A vector* $\boldsymbol{u}$ *is solution of the problem (3.2) if and only if it satisfies the relations:*

- $\boldsymbol{u} \in U, \quad a(\boldsymbol{u}, \boldsymbol{v}-\boldsymbol{u}) \geq l(\boldsymbol{v}-\boldsymbol{u}),\ \forall\ \boldsymbol{v} \in U,$

in the general case, or

- $\boldsymbol{u} \in U, \quad a(\boldsymbol{u}, \boldsymbol{v}) \geq l(\boldsymbol{v}),\ \forall\ \boldsymbol{v} \in U,\ \ a(\boldsymbol{u}, \boldsymbol{u}) = l(\boldsymbol{u}),$

if U *is a closed convex cone with vertex 0, or*

- $\boldsymbol{u} \in U, \quad a(\boldsymbol{u}, \boldsymbol{v}) = l(\boldsymbol{v}),\ \forall\ \boldsymbol{v} \in U,$

if U *is a closed subspace.*

As mentioned, the fundamental results is the Lax-Milgram Lemma (Theorem 3) which assures the existence and uniqueness of a solution for abstract variational problems.

Theorem 3. [3] *Let* V *be a Hilbert space, let* $a(\cdot,\cdot) : V \times V \to \mathbb{R}$ *be a continuous* V*-elliptic bilinear form, and let* $l : V \to \mathbb{R}$ *be a continuous linear form. Then the abstract variational problem: Find* $\boldsymbol{u} \in U$ *such that*

$$a(\boldsymbol{u}, \boldsymbol{v}) = l(\boldsymbol{v}),\ \forall\ \boldsymbol{v} \in V,$$

has one and only one solution.

The above theorems are taken without modification from Ciarlet [7] and skipping in here the prove. The theory allows a variational treatment of fluid problems with the clear advantage that, for specific vectorial spaces and specific bilinear and/or linear forms, such problems as 3.2 will possess a solution and it will be unique.

3.1 Sobolev spaces and variational formulation of the Navier-Stokes equations

Sobolev spaces are functional vector spaces which are very often used in the Numerical Theory. They are fundamental in the process of solving incompressible Navier-Stokes

[2] Ciarlet, P.G. and Lions, J.L., Editors: *"Handbook of numerical analysis", vol. IX, 2003, page 25-26*

[3] Ciarlet, P.G. and Lions, J.L., Editors: *"Handbook of numerical analysis", vol. IX, 2003, page 29*

equations by using a variational formulation, also known in the literature as weak formulation. Therefore, a short introduction to Sobolev spaces is necessary such that, at a later point, based on these spaces, the variational treatment of fluid problems like (2.24) or (2.23) can be presented. We consider to have a domain $\Omega \in \mathbb{R}^d$, delimited by the boundary Γ. The boundary should be sufficiently smooth, Lipschitz continuous for example, Ω locally on one side of Γ. We recall two important Sobolev spaces which will serve later to the construction of the variational formulation, namely $H^1(\Omega)$ and $H_0^1(\Omega)$. Short specifications over the used notations are helpful to understand and recognize the function spaces. The space of all square integrable functions over Ω is denoted by $\mathcal{L}^2(\Omega)$, while the set of C^∞-functions with compact support in Ω is denoted by $\mathcal{D}(\Omega)$. Mathematical definition are presented below:

$$\begin{aligned} &\mathcal{L}^2(\Omega) = \{\varphi \text{ Lebesque mesurable, } \textstyle\int_\Omega |\varphi(x)|^2\, dx < \infty\} \\ &\mathcal{D}(\Omega) = \{\varphi \mid \varphi \in C^\infty(\Omega),\ \varphi \text{ has relatively compact support in } \Omega\}. \end{aligned} \tag{3.3}$$

The Sobolev space $H^1(\Omega)$ is defined as follows:

$$H^1(\Omega) = \{v \mid v \in \mathcal{L}^2(\Omega),\ \frac{\partial v}{\partial x_i} \in \mathcal{L}^2(\Omega),\ \forall\, i = \overline{1,d}\}, \tag{3.4}$$

where the derivatives are considered in the distribution sense, forwarding the reader to Schwartz [39] for details. Shortly, it is meant by derivatives in the distribution sense the following equality:

$$\int_\Omega \frac{\partial v}{\partial x_i}\varphi\, dx = -\int_\Omega v\frac{\partial \varphi}{\partial x_i}\, dx,\ \forall\, \varphi \in \mathcal{D}(\Omega). \tag{3.5}$$

$H^1(\Omega)$ is also called in the field of fluid mechanics the space of velocity vector functions with finite kinetic energy. The elements of $(H^1(\Omega))^d$ form an inner product space with the scalar product:

$$< v, w >_{H^1(\Omega)} := \int_\Omega (\nabla v \cdot \nabla w + vw)\, dx = \int_\Omega \left(\sum_{i=1}^{d} \frac{\partial v}{\partial x_i}\frac{\partial w}{\partial x_i} + vw\right)\, dx. \tag{3.6}$$

Correspondingly, the space possesses a norm defined by (3.7), hence it is in consequence a Hilbert space.

$$||v||_{H^1(\Omega)} := \left(\int_\Omega (|\nabla v|^2 + |v|^2)\, dx\right)^{\frac{1}{2}} = \left(\int_\Omega \left(\sum_{i=1}^{d} \left|\frac{\partial v}{\partial x_i}\right|^2 + |v|^2\right)\, dx\right)^{\frac{1}{2}}. \quad (3.7)$$

An important property of the $H^1(\Omega)$ space is that all its elements have a trace on Γ. The trace operator is the linear mapping defined by :

$$\gamma_0 v = v|_\Gamma,\ \forall\ v \in \mathcal{C}^\infty(\overline{\Omega}) \quad (3.8)$$

with

$$\mathcal{D}(\overline{\Omega}) = \{\varphi \mid \varphi \in C^\infty(\overline{\Omega}), v \text{ has a compact support in } \overline{\Omega}\}. \quad (3.9)$$

A corollary result of the trace theorem claims the existence of a continuous linear operator from $H^1(\Omega)$ to $\mathcal{L}^2(\Gamma)$ which fulfills the following inequality

$$||\gamma_0 v||_{\mathcal{L}^2(\Gamma)} \leq c(\Omega)||v||_{H^1(\Omega)},\ \forall\ v \in H^1(\Omega), \quad (3.10)$$

where $c(\Omega)$ is a constant function. A closed subspace of $H^1(\Omega)$ is the space $H_0^1(\Omega)$. It can be equipped with the semi norm (3.11) and therefrom is a Hilbert space.

$$|v|_{1,\Omega} = \left(\int_\Omega |\nabla v|^2\, dx\right)^{\frac{1}{2}} \quad (3.11)$$

By making use of the variational formulation of the differentiable equation, the restrictions over the velocity function will be reduced by avoiding the second derivatives implied by the Laplace operator from (2.23). Suppose that one has to solve the following problem by constructing firstly the weak formulation:

Find the velocity $u \in H_0^1(\Omega)$ *and the pressure* $p \in \mathcal{L}_0^2(\Omega)$ *with the following properties* $\nabla \cdot \mathbf{u}_0 = 0$ *and* $\mathbf{n} \cdot \mathbf{u}_0 = 0$, *on* Γ *such that:*

$$\begin{cases} \dfrac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} - \nu\Delta\mathbf{u} + \dfrac{1}{\rho}\nabla p = \mathbf{f}, \ \text{ in } \Omega \times (0,T), \\ \nabla \cdot \mathbf{u} = 0, \ \text{ in } \Omega \times (0,T), \\ \mathbf{u}(\mathbf{x},0) = \mathbf{u}_0(\mathbf{x}),\ \forall\ \mathbf{x} \in \Omega, \end{cases} \quad (3.12)$$

This is the nonlinear problem of Navier-Stokes equations with initial condition and

boundary conditions. The natural way to treat the nonlinearity is to use the Newton method or other iterative methods into which we will come later. Previously, the variational formulation of the problem has to be constructed with the advantage of reducing the space regularity. If a two dimensional case is considered, then the velocity vector has the components $\mathbf{u} = (u_x, u_y) = (u_\alpha),\ \alpha = x, y$. One may rewrite the momentum equation and incompressibility constraint of (3.12) as scalar equations for the velocity component like:

$$\begin{cases} \dfrac{\partial u_\alpha}{\partial t} + (\mathbf{u}\cdot\nabla)u_\alpha - \nu\Delta u_\alpha + \dfrac{1}{\rho}\dfrac{\partial p}{\partial \alpha} = \mathbf{f} \\ \\ \displaystyle\sum_\alpha \frac{\partial u_\alpha}{\partial \alpha} = 0. \end{cases} \tag{3.13}$$

The standard technique to formulate the weak problem of a differential equation is a quite simple one. For the selected example, the first two equations are multiplied by a $\mathbb{R}^3$-dot product with an arbitrary function $\varphi \in H_0^1(\Omega)$, followed up by an integration over the whole domain Ω. In the case of the momentum equation, it results in

$$\int_\Omega \varphi \cdot \left[\frac{\partial u_\alpha}{\partial t} + (\mathbf{u}\cdot\nabla)u_\alpha - \nu\Delta u_\alpha + \frac{1}{\rho}\frac{\partial p}{\partial \alpha} \right] d\mathbf{x} = \int_\Omega \varphi \cdot \mathbf{f}\ d\mathbf{x}\ , \tag{3.14}$$

$\forall\ \varphi \in H_0^1(\Omega)$. The left integral is divided into 3 smaller integrals and each will be treated individually. The time derivative and nonlinear terms of the velocity component consist one integral, which needs no modification. On the other hand, the two other integrals containing the Laplace operator, respectively the space derivative of the pressure, will be integrated by parts, which will decrease the order of derivatives of velocity and pressure components. Green's formula will lighten the resulting integral equation:

$$\int_\Omega \varphi\cdot\left[\frac{\partial u_\alpha}{\partial t} + (\mathbf{u}\cdot\nabla)u_\alpha \right] d\mathbf{x} + \nu\int_\Omega \nabla\varphi\cdot\nabla u_\alpha\ d\mathbf{x} - \frac{1}{\rho}\int_\Omega \frac{\partial\varphi}{\partial\alpha}\cdot p\ d\mathbf{x} = \int_\Omega \varphi\cdot\mathbf{f}\ d\mathbf{x} \tag{3.15}$$

Here the following equality was used:

$$-\nu\int_\Omega \Delta\mathbf{u}\cdot\varphi\ d\mathbf{x} = \nu\int_\Omega \nabla\mathbf{u}\cdot\nabla\varphi\ d\mathbf{x} + \int_\Gamma \frac{\partial\mathbf{u}}{\partial\mathbf{n}}\cdot\varphi\ ds, \tag{3.16}$$

with the observation that the surface integral disappears due to the imposed boundary

conditions. The same procedure have to be applied for the incompressibility equation with the immediate result:

$$\int_\Omega q \cdot \sum_\alpha \frac{\partial u_\alpha}{\partial \alpha}\, d\mathbf{x} = 0, \forall\, q \in \mathcal{L}_0^2(\Omega) \tag{3.17}$$

The system of two equations (3.15) and (3.17) represents the variational formulation of the Navier-Stokes equations, endowed with the boundary conditions from (3.12). Having such a formulation, one can define bilinear and trilinear forms which will be helpful in reformulating the problem in a continuous way. The following scalar products are useful in defining these forms:

$$\begin{cases} (w, v) := \int_\Omega w\ v\ d\mathbf{x}, \\ (\mathbf{f}, \mathbf{g}) := \int_\Omega \mathbf{f}\ \mathbf{g}\ d\mathbf{x}. \end{cases} \tag{3.18}$$

The bilinear forms are represented by the following definitions:

$$\begin{cases} a(w, v) := \nu(\nabla w, \nabla v), \\ b_\alpha(w, q) := (\frac{\partial w}{\partial \alpha}, q), \end{cases} \tag{3.19}$$

while the trilinear form is represented by the relation:

$$n(\mathbf{v}, w, u) := a(w, u) + C(\mathbf{v}, w, u), \tag{3.20}$$

where C is also a trilinear form which reads:

$$C(\mathbf{v}, w, u) := (w, \mathbf{v} \cdot \nabla u).$$

Having these forms defined, one can formulate the continuous problem of the considered Navier-Stokes system:

Find the velocity $u \in H_0^1(\Omega)$ *and the pressure* $p \in \mathcal{L}_0^2(\Omega)$ *such that:*

$$\begin{cases} (\varphi, \frac{\partial \mathbf{u}_\alpha}{\partial t}) + n(\mathbf{u}, \varphi, u_\alpha) + b_\alpha(\varphi, p) = (\mathbf{f}, \varphi), \forall\, \varphi \in H_0^1(\Omega), \\ \sum\limits_\alpha b_\alpha(u_\alpha, q) = 0, \forall\, q \in \mathcal{L}_0^2(\Omega). \end{cases} \tag{3.21}$$

The following part of this thesis focuses on the matter of time-space discretization techniques for solving (non)-stationary Navier-Stokes problem with initial condition

and boundary conditions.

3.2 Finite element spaces

Solving a *PDE* system of equations that describes the motion and behavior of fluids is in the general case impossible to solve analytically and therefore is rather a numerical task. In general, such problems have no analytical solution available, therefore they need to be transformed into an algebraic system of equations and solved numerically. This process of obtaining an algebraic system of equations is called discretization of the problem. There are two kinds of discretization: spatial and temporal. By the technique described in the previous sub-chapter, any differential equation may be transformed into a variational/weak formulation and, with the specification that the same differential equation may posses different formulations. One of the advantages of the variational formulation is the possibility of considering weak solutions from spaces like $H^1(\Omega)$. Nonetheless, the natural boundary conditions can be easily and readily imposed in the formulation.
Let us consider the following abstract variational problem:

$$\textit{Find } u \in V, \ \textit{such that } a(u,v) = l(v), \ \forall\, v \in V, \tag{3.22}$$

with the assumptions that $a(\cdot,\cdot), l(\cdot)$ and V are bilinear, linear forms, respectively vectorial spaces, that satisfy the properties of the Lax-Milgram lemma (Theorem 3). It is assured by this lemma that the problem has one and only one solution. Such problems may be treated by a finite element formulation like the Galerkin or Ritz method. For some finite dimensional subspaces of V, the problem can be reformulated as follows:

$$\textit{Let } V_h \subset V. \textit{ Find } u_h \in V_h \textit{ such that } a(u_h, v_h) = l(v_h), \ \forall\, v_h \in V_h. \tag{3.23}$$

The solution of this problem u_h is called discrete solution. The finite element theory represents the process of constructing finite element subspaces and therefore is one of the methods to obtain such finite dimensional spaces like V_h. If one uses *FEM*, one has to take care of three major aspects for a proper usage.

The decomposition of a given domain with its boundary $\partial\Omega$ is called subdivision, no matter what kind of decomposition is used (triangles, squares, rectangles, etc), and it is noted by $\mathcal{T}_h$. The given domain $\overline{\Omega}$ is triangulated if it is subdivided into a finite number of subsets T. The subsets $T \in \mathcal{T}_h$ have to possess a bunch of properties: they have to be closed, with a nonempty interior and connected. Simultaneously, the boundary of any subset T needs to be Lipschitz-continuous. The union of all subsets T is the entire domain $\overline{\Omega}$, with the specification that the intersection of two arbitrary subsets can have just a exactly a closed edge. The above properties of a triangulation can be observed in the figure 3.1.

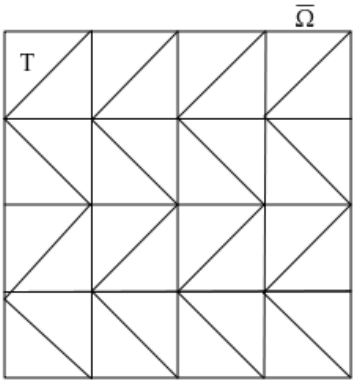


Figure 3.1: Triangulation of a simple shape domain

As long as a triangulation of an arbitrary domain is realized, the finite element space V_h may be defined. The freedom of choosing the space V_h is directly related to how the decomposition is performed. With a finite element space V_h selected, one can define a finite dimensional space by restricting all functions $v_h \in V_h$ to the sets of the triangulation $T \in \mathcal{T}_h$. In other words, one defines the following finite dimensional space $P_T = \{v_h|_T;\ v_h \in V_h\}$. The aim is to get solutions in the described Sobolev spaces $H^1(\Omega)$ or $H_0^1(\Omega)$, therefore certain conditions must be fulfilled such that the selected finite element space V_h is included in them. The following theorem will assure the inclusion:

Theorem 4. [4] A*ssume that* $P_T \subset H^1(T)$ *for all* $T \in \mathcal{T}_h$ *and* $V_h \subset \mathcal{C}^0(\overline{\Omega})$ *hold. Then, the following inclusions hold:*

[4] [7]: *Handbook of Numerical Analysis, Vol II, Chapter II, page 62*

- $V_h \subset H^1(\Omega)$;
- $V_{0h} = \{v_h \in V_h;\ v_h = 0\ on\ \Gamma\} \subset H_0^1(\Omega)$.

The theorem gives good space candidates for solving fluid problems. If the problem is a second-order homogeneous Dirichlet one, then the space to select is V_{0h}. On the other hand, for second-order homogeneous or non-homogeneous Neumann problems, V_h is the the proper one. Further one, there are other results in the theory which suggest different spaces for higher order differential fluid problems, but they are left away since they are not subjects of this thesis.

The sets P_T have an important role as the Theorem 4 indicates. Even more, they play significant role when it comes on the second basic aspect of finite element. Thusly, P_T can mainly be sets of polynomials or functions which behave like polynomials. The benefit of this property will be revealed at a latter point. In practice, if a discrete problem is solved into the space V_h, then the solution has to be a linear combination of the basis vectors of the space. Considering that the space has an arbitrary dimension m and taking the system $\{w_i\}_{i=1}^m$ as basis of V_h, then one might consider the solution to be represented by $u_h = \sum_{i=1}^{m} \xi_i w_i$, where the vector $\{\xi_i\}_{i=1}^m$ satisfies the equality:

$$\sum_{i=1}^{m} a(w_i, w_k)\xi_i = l(w_k),\ 1 \leq k \leq m. \tag{3.24}$$

On the left hand side, the bilinear form $a(\cdot,\cdot)$ builds the so called stiffness matrix, while in the right hand side the linear form determines the load vector. The properties of the stiffness matrix are determined by the properties of the bilinear form $a(\cdot,\cdot)$. For the case of an V-elliptic bilinear form as Lax-Milgram Lemma requires, the stiffness matrix possesses an inverse, therefore the system is solvable. Moreover, if the bilinear form is symmetric, then the stiffness matrix is positive definite, a very important aspect for the numerical solution. Also, it is numerically important that the stiffness matrix contains as many zeros as possible, thus the choice of the basis system of the space V_h is decisive. The following classic example is given to exemplify how the stiffness matrix and load vector are constructed. The model problem is:

$$\begin{cases} -\Delta \mathbf{u} = \mathbf{f}, & \text{in } \Omega, \\ u = 0, & \text{on } \Gamma, \end{cases} \tag{3.25}$$

where $\Gamma = \partial\Omega$. Using the energy principle, the variational formulation of the model problem is:

$$\int_\Omega \nabla \mathbf{u} \nabla \varphi d\mathbf{x} = \int_\Omega f\varphi. \tag{3.26}$$

for any $\varphi \in V$, V being the set of all functions with a square integrable gradient and with the restriction to the boundary Γ equal to zero. In a finite element approach, to solve this problem, the solution is investigated in much smaller spaces $V_h \subset V$ such that:

$$\int_\Omega \nabla \mathbf{u}_h \nabla \varphi_h d\mathbf{x} = \int_\Omega f\varphi_h,\ \forall\, \varphi_h \in V_h. \tag{3.27}$$

For such a space as $V = \{v|\ \nabla v \in \mathcal{L}^2(\Omega),\ v|_\Gamma = 0\}$, the standard norm is:

$$\|v\|_{1,2} = \left(\int |u|^2 dx + \int |\nabla u|^2 dx \right)^{\frac{1}{2}} \tag{3.28}$$

The resulting discrete problem reads:

$$a(u_h, \varphi_h) = (f, \varphi_h),\ \forall\, \varphi_h \in V_h \subset V \tag{3.29}$$

with the definitions of the bilinear and linear forms written below:

$$a(u, \varphi) = \int_\Omega \nabla u \nabla \varphi dx \tag{3.30}$$

$$l(f, \varphi) = \int_\Omega f\varphi dx \tag{3.31}$$

Depending on the triangulation of the domain $\mathcal{T}_h = \{T\}$ and on the selection of the set P_T, one defines a basis of P_T like $\{\varphi_i\}_{i=1}^m$ and approximates the velocity by $u_h(\mathbf{x}, t) = \sum_{i=1}^{N_u} u_i(t)\varphi_i(\mathbf{x})$, where N_u are the number of the degrees of freedom for the velocity. Introducing this solution into (3.29) and using the properties of the bilinear form, the following results arise:

$$\begin{aligned} a\left(\sum_{j=1}^{N_u} u_j(t)\varphi_i(\mathbf{x}), \varphi_i(\mathbf{x})\right) &= (f, \varphi_i(\mathbf{x})),\ \forall\ i = \overline{1,m}, \\ \sum_{j=1}^{N_u} u_j(t) a(\varphi_j(\mathbf{x}), \varphi_i(\mathbf{x})) &= (f, \varphi_i(\mathbf{x})),\ \forall\ i = \overline{1,m}. \end{aligned} \tag{3.32}$$

The algebraic system of equations can be reduced to the generic one:

$$Au = b, \tag{3.33}$$

by using of the following definitions of the stiffness matrix and load vector:

$$A = a_{ij} = a(\varphi_j, \varphi_i) = \int_\Omega \nabla\varphi_i \nabla\varphi_j dx,\ \forall\ i, j = \overline{1,m}, \tag{3.34}$$

$$b = l(\varphi_i) = (f, \varphi_i) = \int_\Omega f\varphi_i dx,\ \forall\ i = \overline{1,m}. \tag{3.35}$$

The solution of the system is the solution of the discrete problem, thus there is an resultant error. It is the purpose of the finite element analysis to investigate how big is the error and how accurate is the process of getting the solution.

Finally, the last basic aspect of a "good" finite element is that in the space V_h exists at least one canonical basis whose corresponding basis functions are easy to be described. If all aspects are together fulfilled, then the corresponding methods are called conforming finite element methods. Indeed, it might be possible to describe different finite element methods which do not fulfill entirely guide themselves over the 3 basic aspects. For instance, the discrete space V_h may not be a subspace of V if we have a curved boundary of the domain, being impossible to realize an exact triangulation in the vicinity of the boundary. The same happens when the domain has inner holes of arbitrary shapes. Also, the bilinear and linear forms can be approximated by numerical integration in order to solve the discrete system. To summarize, the conforming finite element methods require the following steps:

- The finite element space V_h is associated with a triangulation $\mathcal{T}_h$ of the domain together with its boundary. For each element of the triangulation, the space $P_T = \{v_h|_T;\ v_h \in V_h\}$ should contain polynomials or functions that behave like polynomials;

- The space V_h should be a subset of the space of the continuous functions over the domain $\overline{\Omega}$, as stated into Theorem 4;
- The space V_h must posses at least one canonical basis which is easy to describe.

The selection of the finite element and associated finite element spaces have a direct impact over the solution. Another important role is played by the triangulation itself. There are many simple examples of triangulation based on simple geometrical shapes. For the finite element treatment of the methods presented in this thesis: *FEM* and *PM*, the rectangular finite elements were used. The following chapters are an overview over the selected finite element and discretization techniques.

3.3 Time-space discretization of the incompressible Navier-Stokes equations

A non-stationary Navier-Stokes problems can not be discretized by only using finite element method. *FEM* is a specific method to annihilate the spatial derivatives and obtain a simpler algebraic system of equations for the velocity and pressure variables, but is in general not treating the time. Therefore, if the temporal variable is attached to the fluid problem, it is necessary a discretization procedure that involves it also. There are different techniques to realize both time-space discretization of a differential problem, but the most common one is to separate standard discretization in time and space. Firstly, proceed with standard time stepping techniques to discretize in time, followed up by finite element method (or another method) for spatial discretization. The result is a sequence of generalized stationary problems for each time step. In this chapter the whole process of discretization based on this approach will be presented.

3.3.1 Nonconforming $\widetilde{Q_1}/Q_0$ element

As a preamble to the time-space discretization process of the incompressible Navier-Stokes equation, a short introduction to the nonconforming finite element $\widetilde{Q_1}/Q_0$ element is presented, which are in fact the finite element spaces we have used in our investigations. As described in the previous section, the use of the conforming finite

element methods is not always possible or/and sometimes not even recommended or satisfactory. The reasons were presented before, hence they will be omitted for this moment. Thus, nonconforming finite element methods were developed with several critical advantages. For instance, such finite elements posses favorable stability and divergence free nodal bases. In the same time, the pressure variables are permitted to be eliminated. However, the reduction of the unknowns is not the biggest gain, but the possibility to obtain a system of equation for velocity variables alone which may be efficiently solved by multigrid method represent a major strength for this family of finite element methods.

According to Rannacher and Turek [37], the first low order rectangular element with 5 nodal degrees of freedom was introduced by Han, but only theoretically described, not numerically tested. In the same work, Rannacher and Turek introduced the so called $\widetilde{Q_1}/Q_0$ element with both theoretical and numerical support, showing that this element possesses a satisfactory stability and approximations properties. Nevertheless, it satisfies without any additional stabilization the Babuška-Brezzi condition. The $\widetilde{Q_1}/Q_0$ element uses piecewise rotated bilinear shape functions for the velocity component and piecewise constant pressure approximation. Figure 3.2 indicates how the local degrees of freedom are distributed for the actual selected discretization element.

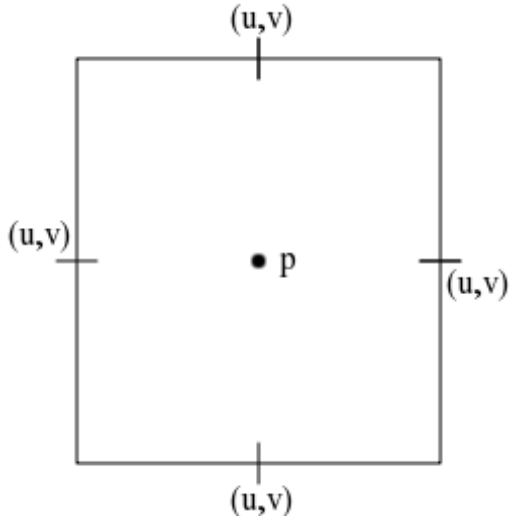


Figure 3.2: Location of the for the $\widetilde{Q_1}/Q_0$ element

As the figure shows, there are 5 nodal values consisting the degrees of freedom for this element: 4 for the velocity vector $\mathbf{u} = (u, v)$ and only 1 for the pressure p. The pressure is a constant value equal to the mean value over the element itself. Therefore, the notation Q_0 consisting of piecewise constant pressure approximation. In the case of the velocity, the 4 values are nothing else than the mean value of velocity over an edge of the local element. The set of functions $\widetilde{Q_1}$ for the velocity uses piecewise rotated bilinear shape functions. The entire description on how the parametric element is obtained for the element $\widetilde{Q_1}/Q_0$, can be found in Turek's work [42]. Considering the reference element $\widehat{T} = [-1, 1] \times [-1, 1]$ and the one-to-one transformation $\Psi_T : \widehat{T} \to T$ from unit reference square to any element T from the triangulation, then the $\widetilde{Q_1}(T)$ element may be defined as follows:

$$\widetilde{Q_1}(T) := \{q \circ \Psi_T^{-1} | q \in \text{ span} < x^2 - y^2, x, y, 1 >\} \tag{3.36}$$

The following nodal functionals, $F_{\Gamma,a/b}$, with $\Gamma \subset \partial T$ being a closed part of an edge of an arbitrary element from the triangulation, determine the degrees of freedom. They are described by (3.37) and lead both to different finite element spaces:

$$\begin{aligned} F_{\Gamma,a}(v) &:= |\Gamma|^{-1} \oint_\Gamma v \, d\gamma, \\ F_{\Gamma,b}(v) &:= v(m_\Gamma). \end{aligned} \tag{3.37}$$

$F_{\Gamma,a}$ represents the mean value of the velocity along Γ and $F_{\Gamma,b}$ is just the velocity value in the midpoint of the edge Γ. Finally, one can set the parametric pair H_h, L_h, $H_h \approx H_0^1(\Omega)$ and $L_h \approx L_0^2(\Omega)$, of finite element spaces corresponding to $\widetilde{Q_1}/Q_0$. They are:

$$L_h := \{q_h \in L_0^2(\Omega) | q_h|_T = \text{ const.}, \forall T \in \mathcal{T}_h\}, \tag{3.38}$$

$$H_{h,a/b} := S_{h,a/b} \times S_{h,a/b}, \tag{3.39}$$

where

$$S_{h,a/b} := \left\{ \begin{array}{l} v_h \in L_0^2(\Omega) | v_h|_T \in \widetilde{Q_1}, \forall T \in \mathcal{T}_h, v_h \text{ continuous w.r.t all} \\ \text{ nodal functionals } F_{\Gamma_{i,j},a/b}(\cdot), \forall \Gamma_{i,j} \text{ and } F_{\Gamma_{i,0},a/b}(v_h) = 0, \forall \Gamma_{i,0}. \end{array} \right\} \tag{3.40}$$

Additionally, one can use also the non-parametric version of $\widetilde{Q_1}/Q_0$ where the reference space $\widetilde{Q_1}(T) := \{q \in span < \chi^2 - \eta^2, \chi, \eta, 1 >\}$ is different and proper for each element T of the triangulation, and defined with respect to a new system of coordinates χ, η which connect the directions of the midpoints of the edges of T. For both parametric and non-parametric versions, the reader is forwarded for more detailed description to the work of Turek [42]. We only recall in this thesis the asymptotic error estimates (3.41) for both pair of finite element spaces $(H_{h,a}, L_h)$, $(H_{h,b}, L_h)$ in the parametric and non-parametric utilization:

$$\begin{cases} ||\mathbf{u} - \mathbf{u}_h||_h + ||p - p_h|| \leq c(h + \sigma_h)\{||u||_2 + ||p||_1\}, \\ ||\mathbf{u} - \mathbf{u}_h||_0 + ||p - p_h||_{-1} \leq c(h + \sigma_h)^2\{||u||_2 + ||p||_1\}; \\ ||\mathbf{u} - \mathbf{u}_h||_h + ||p - p_h|| \leq ch\{||u||_2 + ||p||_1\}, \\ ||\mathbf{u} - \mathbf{u}_h||_0 + ||p - p_h||_{-1} \leq ch^2\{||u||_2 + ||p||_1\}. \end{cases} \tag{3.41}$$

σ_h (3.42) is a new measure to describe the degeneration of the mesh $\mathcal{T}_h$

$$\sigma_h := max\{|\pi - \alpha_T|, \forall T \in \mathcal{T}_h\}, \tag{3.42}$$

where $\alpha_T \in (0, \pi]$ denotes the maximum angle enclosed between the normal unit vectors corresponding to any opposite edges of T. The estimates (3.41) conclude that the convergence of the parametric rotated bilinear elements requires that the mesh is asymptotically uniform, while the non-parametric elements have satisfactory approximation properties on general regular meshes. The approximations (3.41) are corollary results of the necessary and sufficient conditions for existence of a solution $\{U_h, p_h\}$ written below:

$$\begin{array}{l} \inf\limits_{v_h \in H_h} ||v - v_h||_h \leq ch^{m-1}||v||_{H^m}, \forall v \in H \cap \mathbf{H}^m(\Omega), \\ \inf\limits_{q_h \in L_h} ||q - q_h||_h \leq ch^{m-1}||q||_{H^{m-1}}, \forall q \in L \cap H^{m-1}(\Omega) \end{array} \tag{3.43}$$

for some integer values of m greater or equal 2, and

$$\min_{q_h \in L_h/\mathbf{R}} \max_{v_h \in H_h} \frac{b_h(q_h, v_h)}{||v_h||_h ||q_h||0} \geq \tilde{\beta} \tag{3.44}$$

for some constant $\tilde{\beta}$ which does not depend on the aspect ratio of the grid. This set of inequalities are known in the literature as approximation property and, respectively,

stability condition. This pair of element is already implemented in the computational software *FEATFLOW* and are the discretization elements for velocity and pressure components employed during all simulations regarding *FBM* and *PM* investigation and presented into this thesis. The $\widetilde{Q_1}/Q_0$ element is indexed inside the software as *EM30* or *EM31*, depending on the choice of nodal functionals $F_{\Gamma,a/b}$, and referred as non-parametric, non-conformal Rannacher-Turek's element.

3.3.2 Time discretization

The non-stationary Navier-Stokes equations for incompressible fluids requires a "double" discretization approach for time and space. One method which requires less resources is to separate the discretization into time discretization and space discretization. Firstly we proceed with the time discretization and after that, in each created time-step, the spatial discretization is performed. Through the initial time discretization, the temporal derivatives are vanishing and a sequence of time milestones will be determined by an equidistant or adaptive time step size. Simultaneously the problem reduces itself to a stationary Navier-Stokes problem which needs to be discretized in space and solved by using, for example, *FEM* discretization in every time step. It is possible to use simple finite difference method *FDM* for discretization of the time derivatives. For instance usual methods for treating ordinary differential equation *ODE* such as Euler schemes, Crank-Nicolson or Runge-Kutta schemes. However, the main method implemented and used into the *FEATFLOW* software is a Fractional-step-θ-scheme and we will focus on it. The chosen time step scheme has to be accurate in time, easy to realize and efficient/robust from the computational point of view.

Let us consider the simple initial value problem *IVP* (3.45) to be numerically treated:

$$\begin{cases} \dfrac{d\mathbf{u}}{dt} + f(t, \mathbf{u}(t)) = 0, \forall t > 0 \\ \mathbf{u}(0) = u_0. \end{cases} \tag{3.45}$$

with the $\mathbf{u}(t) = ((u_n(t))_{n=1}^d)^T$ and $f(t,x) = ((f(t,x)_{n=1}^d)^T$ vector functions for arbitrary dimension d. There is a very elaborate theory regarding the existence and

uniqueness of solutions, as local/global stability/convergence theory for a large number of categories of different time step schemes. All time-step schemes can be divided into 2 families: explicit and implicit schemes. They are further classified into One-Step-Schemes and Linear-Multistep-Methods *LMM*. The explicit schemes were commonly used to solve non-stationary flow problems even if they have critical disadvantages. In return, there are also clear advantages. They can be easily realized, implemented into a *CFD* code and inexpensive. However, low time step sizes are required for stability reasons, making them inefficient into handling long time-dependent problems. One of the most common explicit schemes is the Explicit Euler *EEM* (3.46), but yet only of first order accurate:

$$u_n = u_{n-1} + h_n f(t_{n-1}, u_{n-1}), n \geq 1 \tag{3.46}$$

To obtain higher order accuracy, one can use explicit Runge-Kutta methods like Heun's schemes of 2nd or 3rd order for example. However, for high stiffness reasons, one should choose an implicit scheme for time-stepping methods. The most frequently used are the Backward Euler-scheme *BE* (3.48) or second order Crank-Nicolson-scheme *CN* (3.49). They belong to the family of one-step-θ-schemes and are also *FD* methods. The basic theta schemes reads:

$$u_{n+1} + \theta h f(t_{n+1}, u_{n+1}) = u_n - (1-\theta) h f(t_n, u_n) \tag{3.47}$$

with a constant time step h and θ parameter chosen in the sub-unitary positive interval $[0, 1]$. For $\theta = 1$ the *BE* method is obtained, while for $\theta = \frac{1}{2}$ the *CN* method arise.

$$u_n = u_{n+1} + h f(t_{n+1}, u_{n+1}), \tag{3.48}$$

$$u_{n+1} + \frac{h}{2} f(t_{n+1}, u_{n+1}) = u_n - \frac{h}{2} f(t_n, u_n). \tag{3.49}$$

BE method is very useful for steady state calculations due to the strong A-stability, assuring a bounded solution, but it is still only of first order accurate. *CN* is on the other hand of second order accurate, but it seems to suffer occasionally from instability because of not being strong A-stable. Glowinski [13] proposed another

θ scheme, the Fractional-step-θ-scheme *FS* (3.51), performing a separation into two problems of nonlinearity and incompressibility. The method uses 3 different values for θ parameter in each time step, so that the (macro)-step $H = t_{n+1} - t_n$ contains 3 sub-steps of variable/equidistant size h. *FS* scheme itself is based on the following settings:

$$\theta = 1 - \frac{\sqrt{2}}{2} \approx 0.293,\ \theta' = 1 - 2\theta \approx 0.414,\ \alpha = \frac{\theta'}{1-\theta} \approx 0.585,\ \text{and } \beta = 1 - \alpha \approx 0.414. \tag{3.50}$$

Choosing $\tilde{\theta} := \alpha\theta h = \beta\theta' h$, the following equations define the time-step scheme *FS*

$$\begin{cases} u^{n+\theta} + \tilde{\theta} f(t^{n+\theta}, u^{n+\theta}) = u^n - \beta\theta h f(t^n, u^n), \\ u^{n+1-\theta} + \tilde{\theta} f(t^{n+1-\theta}, u^{n+1-\theta}) = u^{n+\theta} - \alpha\theta' h f(t^{n+\theta}, u^{n+\theta}), \\ u^{n+1} + \tilde{\theta} f(t^{n+1}, u^{n+1}) = u^{n+1-\theta} - \beta\theta' h f(t^{n+1-\theta}, u^{n+1-\theta}). \end{cases} \tag{3.51}$$

The proposed *FS* method is strong A-stable and, moreover, it possesses second order accuracy.

The problem to be solved is the non-stationary incompressible Navier-Stokes equations (2.24). As mentioned, the primary step is to discretize the system of equations in time. The following problem results:

Given $\mathbf{u}^n$ *and the time step* $H = t_{n+1} - t_n$*, then solve for* $\mathbf{u} = \mathbf{u}^{n+1}$ *and* $p = p^{n+1}$ *the following system of equations*

$$\begin{cases} \frac{\mathbf{u}-\mathbf{u}^n}{H} + \theta\left[-\nu\Delta\mathbf{u} + \mathbf{u}\cdot\nabla\mathbf{u}\right] + \nabla p = \mathbf{g}^{n+1} \\ \nabla\cdot\mathbf{u} = 0,\ \text{in } \Omega \end{cases} \tag{3.52}$$

with the rhs

$$\mathbf{g}^{n+1} := \theta\mathbf{f}^{n+1} + (1-\theta)\mathbf{f}^n - (1-\theta)\left[-\nu\Delta\mathbf{u}^n + \mathbf{u}^n\cdot\nabla\mathbf{u}^n\right]. \tag{3.53}$$

For making easier to write the equations, the diffusive and advective parts are denoted as $N(\mathbf{v})\mathbf{u}$. The following definition for $N(\mathbf{u})\mathbf{u}$ follows:

$$N(\mathbf{v})\mathbf{u} := -\nu\Delta\mathbf{u} + \mathbf{v}\cdot\nabla\mathbf{u}. \tag{3.54}$$

The fractional-θ-scheme (3.51), applied to the problem (3.52), decomposes it into 3

consecutive sub-steps for a macro-step of size H.

$$\left\{\begin{array}{l}\left\{\begin{array}{l}[I+\tilde{\theta}N(\mathbf{u}^{n+\theta})]\mathbf{u}^{n+\theta}+\theta H\nabla p^{n+\theta}=[I-\beta\theta HN(\mathbf{u}^{n})]\mathbf{u}^{n},\\ \nabla\cdot\mathbf{u}^{n+\theta}=0,\end{array}\right.\\ \left\{\begin{array}{l}[I+\tilde{\theta}N(\mathbf{u}^{n+1-\theta})]\mathbf{u}^{n+1-\theta}+\theta' H\nabla p^{n+1-\theta}=[I-\alpha\theta' HN(\mathbf{u}^{n+\theta})]\mathbf{u}^{n+\theta},\\ \nabla\cdot\mathbf{u}^{n+1-\theta}=0,\end{array}\right.\\ \left\{\begin{array}{l}[I+\tilde{\theta}N(\mathbf{u}^{n+1})]\mathbf{u}^{n+1}+\theta H\nabla p^{n+1}=[I-\beta\theta HN(\mathbf{u}^{n+1-\theta})]\mathbf{u}^{n+1-\theta},\\ \nabla\cdot\mathbf{u}^{n+1}=0.\end{array}\right.\end{array}\right. \tag{3.55}$$

With this time discretization technique, the problem to be solved in each macro-step H is:

Given $\mathbf{u}^n$, *parameters* $h=H(t_{n+1})$, $\theta=\theta(t_{n+1})$ *and* $\theta_i=\theta_i(t_{n+1}), i=1,...,3$, *then then solve for* $\mathbf{u}=\mathbf{u}^{n+1}$ *and* $p=p^{n+1}$

$$\left\{\begin{array}{l}[I+\theta hN(\mathbf{u})]\mathbf{u}+h\nabla p=[I-\theta_1 hN(u^n)]\mathbf{u}^n+\theta_2 h\mathbf{f}^{n+1}+\theta_3 h\mathbf{f}^n\\ \nabla\cdot\mathbf{u}=0,\ \text{in}\ \Omega\end{array}\right. \tag{3.56}$$

FS method has the property that it combines both advantages of BE and CN methods, being easy to realize, A-stable, second order accurate and with no extra numerical effort. In the works [44, 46], Turek at. all presented a modified fractional-θ-scheme starting with a BE sub-step, continuing with extrapolation of the last two approximated solutions to obtain an accurate one and applying another BE sub-step to get the solution for the new macro-step. The method is fully implicit one and is part of Runge-Kutta family methods. The numerical simulations offered in [44, 46] prove that the modified method is strong A-stable and second order accurate, in fact, almost third order accurate. All these schemes are already implemented into our *FEATFLOW* code and may be used in combination with the FBM and PM. In Chapter 7, we present a simple problem of an oscillating cylinder, which is a non-stationary simulation, discretized by using BE since we only intended to prove the capabilities of the PM. However, the FS was successfully implemented together with FBM in [47].

3.3.3 Space discretization

Second and last step in the discretization process is the space discretization by finite element method. As presented in the previous chapters, to proceed, one needs to formulate a variational or weak corresponding problem, to realize a triangulation of triangles or quadrilaterals (in 2D) that covers the entire fluid domain and define polynomial trial functions/spaces for both unknown variables: velocity and pressure. Nevertheless, the selected approximate spaces $\mathbf{H}_h$ and L_h should provide the possibility of receiving stable approximations if the mesh size h becomes smaller and smaller during the refinement process. This last aspect is fulfilled if the pair of elements satisfies the Babŭska-Brezzi condition (3.57) with a mesh independent constant β:

$$\min_{p\in L_h}\max_{\mathbf{v}_h\in\mathbf{H}_h}\frac{(p_h,\nabla\cdot\mathbf{v}_h)}{||p_h||_0||\nabla\mathbf{v}_h||_0}\geq\beta\geq 0 \tag{3.57}$$

One candidate that satisfies (3.57) is the pair of elements $\widetilde{Q}_1/Q_0$ with both parametric and non-parametric versions. Theoretical and numerical results regarding this element were presented by Turek and Rannacher [37], proving that $\widetilde{Q}_1/Q_0$ behaves comparable with the standard conforming finite elements of first order in energy norm from the point of view of accuracy, but unconditionally stable, even on anisotropic meshes. Consider the pair of variables $\{\mathbf{u},p\}\in H(\Omega)\times L(\Omega)$, with H and L being the special Sobolev space $H_0^1(\Omega)$ and the space $L_0^2(\Omega)$. Further, the two bilinear forms $a(\mathbf{u},\mathbf{v})$ and $b(p,\mathbf{v})$ defined in (3.19) and rewritten as (3.58) will help to formulate a variational problem for each nonlinear problem (3.56) of each time step for the non-stationary case:

$$\begin{cases} a(\mathbf{u},\mathbf{v}):=\nu(\nabla\mathbf{u},\nabla\mathbf{v}),\\ b(p,\mathbf{v}):=-(p,\nabla\cdot\mathbf{v}).\end{cases} \tag{3.58}$$

Eventually, one weak formulation of the equation (3.56) is:

$$\begin{cases} (\mathbf{u},\mathbf{v})+\theta_1 H[a(\mathbf{u},\mathbf{v})+n(\mathbf{u},\mathbf{u},\mathbf{v})]+\theta_2 Hb(p,\mathbf{v})=(\mathbf{f},\mathbf{v}),\forall\ \mathbf{v}\in H,\\ b(q,\mathbf{u})=0,\forall q\in L.\end{cases} \tag{3.59}$$

Note that the term $n(\mathbf{u},\mathbf{u},\mathbf{v})$ is the trilinear form defined by (3.20). To discretize (3.59), one has to perform a triangulation $\mathcal{T}_h$, i.e. based on quadrilaterals, for the

whole computational domain Ω_T, with h symbolizing the maximum width of the elements $T \in \mathcal{T}_h$. To refine the mesh, starting with a coarse one $\mathcal{T}_h$ and obtaining a finer one $\mathcal{T}_{2h}$, one can just simply connect the opposing midpoints of all elements $T \in \mathcal{T}_h$, such that they become in the new element $T \in \mathcal{T}_h$ vertices. By choosing the finite element pair $\widetilde{Q}_1/Q_0$ and so the finite element spaces H_h and L_h defined as in (3.39) and (3.38), then the discrete solution $\{u_h, p_h\} \in H_h \times L_h$ has to satisfy the system:

$$\begin{cases} (\mathbf{u}_h, \mathbf{v}_h) + \theta_1 H[a(\mathbf{u}_h, \mathbf{v}_h) + n_h(\mathbf{u}_h, \mathbf{u}_h, \mathbf{v}_h)] + \theta_2 H b_h(p_h, \mathbf{v}_h) = (\mathbf{f}, \mathbf{v}_h), \forall\ \mathbf{v}_h \in H_h, \\ b_h(q_h, \mathbf{u}_h) = 0, \forall q_h \in L_h \end{cases} \tag{3.60}$$

where

$$\begin{cases} a_h(\mathbf{u}_h, \mathbf{v}_h) := \sum\limits_{T \in \mathcal{T}_h} a_h(\mathbf{u}_h, \mathbf{v}_h)|_T, \\ b_h(p_h, \mathbf{v}_h) := \sum\limits_{T \in \mathcal{T}_h} b(p_h, \mathbf{v}_h)|_T, \\ n_h(\mathbf{u}_h, \mathbf{u}_h, \mathbf{v}_h) := \sum\limits_{T \in \mathcal{T}_h} n(\mathbf{u}_h, \mathbf{u}_h, \mathbf{v}_h)|_T, \end{cases} \tag{3.61}$$

It has to be remarked that for the selected finite element spaces there is no need of extra stabilization techniques regarding the *LBB* condition. However, any kind of stabilization methods like upwind stabilization, streamline-diffusive stabilization or edge-oriented stabilization, can be applied in certain configurations or for different purposes (i.e. high Re numbers). If such stabilization methods are used, then they will be implemented inside the trilinear form which contains the non-linearity. The problem is now discretized in both temporal and spatial dimensions and has to be solved by applying linear and nonlinear Navier-Stokes solvers, subject of the following chapter.

Chapter 4

Numerical solver

After the discretization of the incompressible Navier Stokes problem in time and space, it results a discrete nonlinear system of equations to be numerically solved. The nonlinearity and the incompressibility are just two of the points to be taken into account. Suppose that the non-stationary Navier Stokes problem (2.23) with some boundary conditions and implemented with penalty term has to be solved. It is firstly discretized in time by one of the proposed time-step schemes, like *BE*, *CN* or *FS* presented in the previous chapter, followed by a spatial discretization by using one of the following methods *FEM*. It results discrete a system of equations like (3.60), which is in every time step a stationary system of equations and can be rewritten in a matrix form as follows:

$$\begin{cases} \Big[\mathbf{M} + \theta H(\lambda\chi(\mathbf{x})\widetilde{\mathbf{M}} + \mathbf{N}(\mathbf{u}_h^n)\Big)]\mathbf{u}_h^n + H\mathbf{B}p_h = \mathbf{g}_h^n, \\ \mathbf{B}^T\mathbf{u}_h = 0. \end{cases} \tag{4.1}$$

with

$$\begin{aligned} \mathbf{g}_h^n = \mathbf{u}_h^{n-1} + (1-\theta H)(\lambda\chi(\mathbf{x})\widetilde{\mathbf{M}} + \mathbf{N}(\mathbf{u}_h^{n-1})\mathbf{u}_h^{n-1} + \\ +\theta(f_n + \lambda\chi(\mathbf{x})\overline{\mathbf{u}}^n) + (1-\theta)(f_{n-1} + \lambda\chi(\mathbf{x})\overline{u}^{n-1}). \end{aligned} \tag{4.2}$$

$\mathbf{M}, \widetilde{\mathbf{M}}, \mathbf{N}, \mathbf{B}$ and $-\mathbf{B}^T$ stand for the mass matrix, penalty matrix, the nonlinear (convection-diffusion) operator, the gradient matrix containing the discrete derivatives and the transposed divergence matrix. Here, the right hand side term $\overline{\mathbf{u}}^n$ represents the velocity of the solid object, which is time-dependent for a moving object. The consistent mass matrix M_c is not a diagonal matrix, is symmetric and it is defined

in the case of *FEM* by the equality:

$$\mathbf{M}_c = \{m_{ij}\} = \int_\Omega \varphi_i \varphi_j d\mathbf{x} \in \mathbb{R}^{N_u \times N_u}, \tag{4.3}$$

with N_u the number of degrees of freedom for the velocity components and φ_i, φ_j the test, respectively trial, functions of the selected finite element spaces. Usually, the consistent matrix, which is a full-matrix, is substituted by the lumped mass matrix M_L defined below:

$$\mathbf{M}_L = \text{ diag } \{m_i\} = \sum_{j=\overline{1,N_u}} m_{ij} \tag{4.4}$$

$\widetilde{\mathbf{M}}$ represents the penalty matrix, which is nothing else than a modified mass matrix. For the construction of $\widetilde{\mathbf{M}}$, the same integrals (4.3) are computed to obtain the entries, but only those are activated by the mask function $\chi(\cdot)$ which correspond to the degrees of freedom (*DOF*s) inside the penalized object. We usually decide for a full penalty matrix and do not apply lumping techniques. However, if the mass matrix is lumped, penalty matrix has to be in general also lumped and therefore we have chosen a HRZ-lumping method [20], which will be presented later. The mass matrix is linear, is not depending on the solution and, hence needs to be assembled only once and recalled in every time step for non-stationary simulations. On the other hand, the penalty matrix has to be reassembled every time step for the non-stationary case, since the position and velocity of the solid object may change in time and so the penalty matrix has to be updated. The nonlinear convention-diffusion operator $\mathbf{N}$, also known as Burgers operator, consists of two terms: convective and diffusive. The relations of definition are the following:

$$\begin{cases} \mathbf{N}_\alpha = \{n_{ij}\} \in \mathbb{R}^{N_u \times N_u}, \quad n_{ij} = c_{ij} + d_{ij}, \\ c_{ij} = (\varphi_i, \mathbf{u}_h \cdot \bigtriangledown \varphi_j) = \int_\Omega \varphi_i \mathbf{u}_h \cdot \bigtriangledown \varphi_j d\mathbf{x}, \\ d_{ij} = \nu \left(\bigtriangledown \varphi_i, \bigtriangledown \varphi_j\right) = \nu \int_\Omega \bigtriangledown \varphi_i \cdot \bigtriangledown \varphi_j d\mathbf{x}. \end{cases} \tag{4.5}$$

with $\alpha \in \{x, y\}$ for the bi-dimensional case. The convection-diffusion operator is non-symmetric and nonlinear. It depends on the solution and needs to be reassembled in every time step. For the particular case of a laminar fluid, with very small value of Reynolds number, the convective part can be neglected and so the operator becomes

linear and rectangular. The discrete gradient $\mathbf{B}$ and its divergence $\mathbf{B}^T$ are given through the equalities:

$$\mathbf{B}_\alpha = \{b_{ik}^\alpha\} = \left(\frac{\partial \varphi_i}{\partial \alpha}, \psi_k\right) \in \mathcal{R}^{N_u \times N_p} \tag{4.6}$$

with N_p the degrees of freedom for the pressure component.
One can reduce the matrix system of equations (4.1) to a simpler one:

$$\begin{cases} \mathbf{S}(\mathbf{u}_h)\mathbf{u}_h + H\mathbf{B}p_h = \mathbf{g}_h, \\ \mathbf{B}^T\mathbf{u}_h = 0, \end{cases} \tag{4.7}$$

with the velocity matrix $\mathbf{S}$ given by the relation:

$$\mathbf{S}(\mathbf{u}) = \left[\mathbf{M} + \theta H(\lambda\chi(\mathbf{x}) + \widetilde{\mathbf{M}}\mathbf{N}(\mathbf{u}))\right] \tag{4.8}$$

and the right hand side (rhs) defined by (4.2). The matrix S has the structure:

$$\mathbf{S}(\mathbf{u}) := \alpha M + \theta_1 H\lambda_\chi \widetilde{M} + \theta_1 \nu HL + \theta_2 HK(\mathbf{u}), \tag{4.9}$$

with the parameters θ_i, α, H given by the discretization method in time and space, ν the viscosity, λ the penalty parameter which is rather very high, M the mass matrix, L the discretized Laplacian and K the corresponding convective term.
After discretization in time by applying one of the presented θ-schemes, the resulting stationary problem can be treated in general in two ways in order to solve the problem. One possibility is to treat firstly the nonlinearity by a fixed point defect correction type method, resulting linear subproblems of Oseen type in each nonlinear step and further solved by a direct coupled or a splitting approach separately for velocity and pressure. This method, denoted into *FEATFLOW* under the names of *cc2d/cc3d* or *cp2d/cp3d* is belonging to the Galerkin type of schemes. The second possibility supposes a splitting into definite problems in velocity, Burgers equations, as well as in pressure, Pressure-Poisson problems and the treatment of the nonlinearity through an appropriate linearisation technique. This method is a Projection scheme and is denoted into *FEATFLOW* software with *pp2d/pp3d*. Both methods deliver similar solutions if they converge at all and if the time steps k are sufficiently small, or the values of the nonlinear N or linear L iterations are sufficiently large selected.

For the present study, we used the direct coupled velocity-pressure software $cc2d$, meaning that for both FM and PM a Galerkin type of method to solve the resulting algebraic system is applied. Firstly, we treat the nonlinearity by a fix-point defect-correction method, an iterative scheme which provides a solution u^{n+1} by starting with the known solution u^n, followed by a linear solver to solved the linearized system. Also a Newton iteration scheme can be applied, but since the PM does not affect at all the nonlinear term, the defect-correction loop scheme will provide a better behavior of the solver. Besides, the defect-correction loop can accelerate the rate of convergence by choosing an optimal damping parameter. In the following part the actual solving process of the penalized Navier-Stokes problem will be shortly described. Suppose we have the generic problem governed by the algebraic equation:

$$A(\mathbf{u})\mathbf{u} = b \tag{4.10}$$

where $b = f - B_p$. One can apply iterative methods in order to treat the nonlinearity for both stationary or non-stationary situations. Abstractly, such methods are actually successive approximations of the solution. One possibility is the iterative defect-correction scheme, in which for each iteration a solution u^n is known and is updated to u^{n+1} by performing one (nonlinear) relaxation step. This is described trough the equation:

$$\mathbf{u}^{n+1} = \mathbf{u}^n + \omega^m \left[\tilde{A}(\tilde{\mathbf{u}}^n)\right]^{-1} \left[b - A(\mathbf{u}^n)\mathbf{u}^n\right], n = 0, 1, 2, \ldots. \tag{4.11}$$

ω^n is the relaxation parameter and should be chosen appropriately, while $\tilde{A}(\tilde{\mathbf{u}}^n)$ is a preconditioner of the operator $A(\cdot)$. The iterations stops if the nonlinear residual is sufficiently small. The convergence rate depends on the properties of $\tilde{A}$ and, in the case of non-stationary problems, on the time step size and control. Also, the relaxation parameter may be fixed or chosen adaptively. We prefer actually the adaptive fixed point defect correction approach. In the practical implementation, firstly we evaluate the residual:

$$r^m = b - A(\mathbf{u}^n)\mathbf{u}^n \tag{4.12}$$

of the algebraic system (4.10). In a second step, we solve the linear subproblem:

$$\tilde{A}(u^m)\delta\mathbf{u}^n = r^m \tag{4.13}$$

for $\delta\mathbf{u}^n$. The obtained solution is multiplied by the relaxation parameter and added to the previous solution, such that the result after an iteration is:

$$\mathbf{u}^{n+1} = \mathbf{u}^n + \omega^n \cdot \delta\mathbf{u}^n. \tag{4.14}$$

Applying this defect correction technique to the corresponding continuous problem (4.7) it follows:

$$\begin{bmatrix} \mathbf{S}(\mathbf{u}) & \mathbf{B} \\ \mathbf{B^T} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ p \end{bmatrix} = \begin{bmatrix} g \\ 0 \end{bmatrix} \tag{4.15}$$

1. Residual evaluation

$$\begin{bmatrix} r_u^n \\ r_p^n \end{bmatrix} = \begin{bmatrix} g \\ 0 \end{bmatrix} - \begin{bmatrix} \mathbf{S}(u^n) & \mathbf{B} \\ \mathbf{B^T} & 0 \end{bmatrix} \begin{bmatrix} u^n \\ p^n \end{bmatrix} \tag{4.16}$$

2. Linear subproblems

$$\begin{bmatrix} \mathbf{S}(u^n) & \mathbf{B} \\ \mathbf{B^T} & 0 \end{bmatrix} \begin{bmatrix} \delta u^n \\ \delta p^n \end{bmatrix} = \begin{bmatrix} r_u^n \\ r_p^n \end{bmatrix} \tag{4.17}$$

3. Relaxation and update

$$\begin{bmatrix} u^{n+1} \\ p^{n+1} \end{bmatrix} = \begin{bmatrix} u^n \\ p^n \end{bmatrix} + \omega^n \begin{bmatrix} \delta u^n \\ \delta p^n \end{bmatrix} \tag{4.18}$$

where g contains the rhs and the penalty term generated by the moving object. The standard Newton's method is obtained if the $\tilde{A}(\cdot)$ is the exact Frechét derivative. As mentioned before, the method provides a quadratic convergence if it converges at all, but often destroys the diagonal dominance of the matrix $\tilde{A}(\cdot)$ because of the convective part, causing instabilities of the solution. On the other hand, if $\tilde{A}(\cdot)$ is the operator $A(\cdot)$ itself, then a fixed point scheme is used. The convergence rate in this case is not anymore quadratic as in Newton's scheme, but the method may be more efficient in solving the linear subproblems. The combination of adaptive

damping and Newton iteration will provide the best method. By applying strategies of adaptive relaxation parameter, then the corresponding adaptive fixed point defect correction method may have even better convergence behavior then Newton's method. There are many possibilities of applying an adaptive fixed point method, depending on the preconditioning operator and on the damping parameter. In general, for the continuous incompressible Navier Stokes problem:

$$\begin{cases} \alpha\mathbf{u} + \lambda\chi(\mathbf{x})\mathbf{u} - \nu\Delta\mathbf{u} + \mathbf{u}\cdot\bigtriangledown\mathbf{u} + \bigtriangledown p = f + \lambda\chi(\mathbf{x})\overline{\mathbf{u}}_t, \\ \bigtriangledown\cdot\mathbf{u} = g \end{cases} \tag{4.19}$$

and introducing the following approximate Frechét operator

$$\tilde{T}(\mathbf{v}) := \begin{bmatrix} \tilde{S}(\mathbf{v}) & \bigtriangledown \\ \bigtriangledown\cdot & 0 \end{bmatrix}, \tag{4.20}$$

then in a nonlinear iteration, knowing $\mathbf{u}^n$ and p^n, a relaxation step is performed as follows:

$$\begin{bmatrix} res\mathbf{u}^n \\ resp^n \end{bmatrix} = \begin{bmatrix} \alpha\mathbf{u}^n\lambda\chi(\mathbf{x}) - \nu\Delta\mathbf{u}^n + \mathbf{u}^n\cdot\bigtriangledown\mathbf{u}^n + \bigtriangledown p^n - f - \lambda\chi(\mathbf{x})\overline{\mathbf{u}_t} \\ \bigtriangledown\cdot\mathbf{u}^n - g \end{bmatrix}, \tag{4.21}$$

such that the iterate solutions $\mathbf{u}^{n+1}, p^{n+1}$ are obtained:

$$\begin{bmatrix} \mathbf{u}^{n+1} \\ p^{n+1} \end{bmatrix} = \begin{bmatrix} \mathbf{u}^n \\ p^n \end{bmatrix} - \omega^n \begin{bmatrix} \tilde{S}(\mathbf{u}^n) & \bigtriangledown \\ \bigtriangledown\cdot & 0 \end{bmatrix}^{-1} \begin{bmatrix} res\mathbf{u}^n \\ resp^n \end{bmatrix}. \tag{4.22}$$

For further schemes regarding the choice of the preconditioner, the relaxation parameter and the time step, or treatment of the nonlinearity by an iterative method, the reader is forwarded to [42], Chapter 3.3. We summarize the above described methods into a typical example of Galerkin scheme. It is the scheme which, from our background, solves exactly on a high level the discrete nonlinear problems of type

$$\begin{cases} S\mathbf{u} + kBp = \mathbf{g} + \lambda_\chi\overline{\mathbf{u}_t}, \\ B^T\mathbf{u} = 0, \end{cases} \tag{4.23}$$

with the nonlinear operator:

$$S\mathbf{u} := \left[\alpha M + \theta_1 k\lambda_\chi \widetilde{M} + \theta_1 k\nu HL + \theta_2 kK(\mathbf{u})\right]\mathbf{u}. \tag{4.24}$$

One knowns the iterates $(\mathbf{u}^{l-1}, p^{l-1})$ and performs a defect correction step to obtain $(\mathbf{u}^l, p^l)$:

$$\begin{bmatrix} \mathbf{u}^l \\ p^l \end{bmatrix} = \begin{bmatrix} \mathbf{u}^{l-1} \\ p^{l-1} \end{bmatrix} - \omega^n \begin{bmatrix} \tilde{S}(\mathbf{u}^{l-1}) & kB \\ B^T & 0 \end{bmatrix}^{-1} \times \left(\begin{bmatrix} S & HB \\ B^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u}^{l-1} \\ p^{l-1} \end{bmatrix} - \begin{bmatrix} \mathbf{g} + \lambda_\chi \overline{\mathbf{u}}_t \\ 0 \end{bmatrix} \right). \tag{4.25}$$

The upcoming task is to solve the linearized Oseen equations, resulting in each nonlinear iterations. One family of methods are the general Pressure Schur Complement *PSC* which firstly calculate the pressure component and then determines the velocity field. Several schemes belong to the class of *PSC* methods like: projection methods, pressure correction schemes, *SIMPLE*, Uzawa iteration or Vanka smoother. We do not describe all these methods, but more details can be found in [42]. Due to the form of the matrix S, which represents the velocity matrix and contains the mass, penalty matrix and also the laplacian and convective terms, we have chosen a bankable smoother [49] for multigrid approach. It is actually a local multilevel Pressure Schur Complement *MPSC* and has the advantage of assembling the whole velocity block in a single full matrix. Generically, the method is very similar to a domain decomposition approach, based on the idea of solving small problems in contrast with the global *MPSC* and use direct solvers like Gauß-Seidel or Jacobi iteration to solve the linearized system. Therefore, the actual computational domain is decomposed in patches Ω_i with the property that the union of all patches give the entire domain Ω. The number of patches NP is selected in the range $[1, NEL]$, where NEL are the number of the finite element. A local coupled stiffness matrix A_i (4.26) is assembled and the corresponding subproblem is solved applying a direct solver, i.e. Gauss elimination, meaning that an inverse matrix A_i^{-1} is directly applied to the rhs vector. Therefore, the local stiffness matrix should not be singular, in order to obtain a solution.

$$A_i = \begin{pmatrix} \widetilde{S_{\Omega_i}} & kB_{\Omega_i} \\ B_{\Omega_i}^T & 0 \end{pmatrix}. \tag{4.26}$$

The entries of the matrix and the boundary conditions are taken from the global matrix A. During the solving process, the compact matrix

$$P_i = B_{\Omega_i}^T \widetilde{S_{\Omega_i}^{-1}} B_{\Omega_i} \tag{4.27}$$

is used to solve an equivalent local pressure Schur complement problem. This matrix is usually a full-matrix, containing all velocity matrix components, i.e. mass, penalty, convective and diffusive terms, but possesses smaller number of unknowns. After the local pressure is obtained, the velocity field can be determined as solution of the momentum equation. The basic iteration of the local *MPSC* is finally summarized by (4.25), where $\mathbf{u}^{l-1}$ and p^{l-1} are given. In practice we have several sub-steps: calculate the defect for the given solutions, solve the local problem and update the new solutions via the relaxation parameter. All these steps are described by the following mathematical model:

1. Residual evaluation

$$\begin{bmatrix} \mathbf{defu}_i^{l-1} \\ defp_i^{l-1} \end{bmatrix} = \left(\begin{bmatrix} g + \lambda_\chi \overline{\mathbf{u}}_t \\ 0 \end{bmatrix} - \begin{bmatrix} \mathbf{S} & k\mathbf{B} \\ \mathbf{B^T} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u}^{l-1} \\ p^{l-1} \end{bmatrix} \right)_{\Omega_i} \tag{4.28}$$

2. Solve linear subproblem

$$\begin{bmatrix} \widetilde{\mathbf{S}_{\mathbf{\Omega_i}}} & k\mathbf{B} \\ \mathbf{B^T} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{v}_i^l \\ q_i^l \end{bmatrix} = \begin{bmatrix} \mathbf{defu}_i^{l-1} \\ defp_i^{l-1} \end{bmatrix} \tag{4.29}$$

3. Relaxation and update of solution

$$\begin{bmatrix} \mathbf{u}_{\Omega_i}^l \\ p_{\Omega_i}^l \end{bmatrix} = \begin{bmatrix} \mathbf{u}_{\Omega_i}^{l-1} \\ p_{\Omega_i}^{l-1} \end{bmatrix} - \omega^l \begin{bmatrix} \mathbf{v}_i^l \\ p_i^l \end{bmatrix} \tag{4.30}$$

We can use direct solvers like *UMFPACK*, which is set of routines to solve linearized problems. However we prefer the so called Multigrid (*MG*) approach, hence a local *MPSC* is applied. The framework of *MG* is based on cycles of type V-, W- or F-(full) as illustrated in the figure (4.1). A cycle is consisting on 3 major steps: pre-smoothing, local *MPSC* and post-smoothing step. The pre-smoothing step gives a better approximation of the initial solution starting from the given one:

$$u_k^m = S_m(u_k^0). \tag{4.31}$$

The number of pre-smoothing steps m is free of choice, but a big number will directly have effect on the computational time. Trough the smoothing process, the high frequency error are eliminated. This step is followed by a complete local *MPSC* step, with the correction step:

$$f_{k-1} = I_k^{k-1}(f_k - A_k u_k^m) \tag{4.32}$$

recursively repeated trough restriction until the coarser grid is reached. After each iteration results a solution of the kind:

$$u_{k-1} = MPSC(k-1, u_{k-1}^{i-1}, f_{k-1}), 1 \leq i \leq p, u_{k-1}^0 = 0. \tag{4.33}$$

The solution obtained on the coarse level is prolongated to the higher grids, such that the final solution on the finer grid is:

$$u_k^{m+1} = u_k^m + \alpha_k I_{k-1}^k u_{k-1}^p, \tag{4.34}$$

It is further smoothed to get the final solution u_k^{m+1}. Here, α_k controls the error between the solution of two neighbor levels m and $m+1$. A proper selection of α_k will minimize these errors.

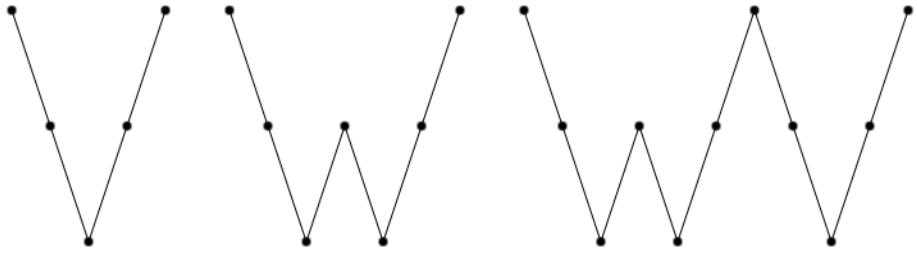


Figure 4.1: Multigrid Cycles

Multigrid method and its applications can be found in the work of Hackbusch [16]. To summarize the behavior of the *MG* technique, notions like smoothing operator, restriction/prolongation or grid transfer operator, coarse grid operators, step-length

control and matrix-vector multiplication are the typical components. Each aspect has direct impact on the behavior of the *MG* approach. Schemes like Classical Richardson, Jacobi iteration, Gauss-Seidel/SOR iteration, ILU schemes and its variants are some of the linear basic iterations to solve the linear problem

$$A\mathbf{x} = b. \tag{4.35}$$

In general, these schemes behave as solvers with unsatisfactory results regarding accuracy and efficiency, therefore, they need to be improved by smoothing processes like in the case of *MG*. The smoothing techniques are responsible for damping the high frequencies. In this study, the *MG* is coupled with a *VANKA* like smoother, more details over this smoothing technique can be found in the work of Vanka [49]. It is to be said that usually, the smoothing steps are not many, but in the case of *PM* with high values of the ε-penalty parameter a larger number of smoothing steps may be imposed to obtain or ensure convergence and satisfactory accuracy.

Chapter 5

Fictitious Boundary and Penalty Methods

This chapter is fully dedicated to the actual *FBM* and *PM* approaches, presenting the methods, the implementation details and also the numerical results for the stationary simulation of flow around cylinder for validation. Firstly we proceed with a short survey over the class of methods in which the investigated methods are included.

Already from the middle of the last century solving Navier Stokes equations for steady or unsteady incompressible flows was one of the main topics in the field of numerical simulation. Several methods were developed over the years, classified into two major classes: Arbitrary Lagrangian-Eulerian *ALE* or standard Galerkin finite element methods, and the so called fictitious domain methods, also known over the time as distributed Lagrange multiplier *DLM* or immersed boundary methods. The purpose of all these methods in solving Navier Stokes equations is to develop an easy to implement accurate, efficient and robust method, with the advantage of using fast solvers.

R. Glowinski, D. D Joseph and coauthors have developed a method which uses the principle of embedded or fictitious domains known as the distributed Lagrange multiplier *DLM* method. This method is also considered to be an implicit fictitious boundary approach. Based on this method, Turek et. all [50] proposed a multigrid *FEM* explicit fictitious boundary method *FBM*. The flow is computed by a multigrid finite element solver and the solid objects are allowed to move freely trough the fluid

domain. The behavior of the objects is described by the physical laws of movement, while the fluid behavior is described by the Navier Stokes equations. This particularly allows the use of cartesian grids which do not need to be re-meshed during the simulation.

Starting from the already validated *FBM* method proposed by Turek et. all the focus of this study is to investigate a penalization method *PM* as a generalization method of the *FBM*. The method is a penalization type method for computing incompressible Navier Stokes equations around complex geometrical shapes. The proposed penalty method penalizes only the velocity part of the Navier-Stokes problem. We compare *FBM* and *PM* w.r.t. aspects of accuracy, efficiency of nonlinear and linear solvers and calculation of quantitative measurements such as velocity and pressure point values, but also drag and lift forces calculations. We set as reference values the results obtained with the $FEATFLOWcc2d$ software tool in simulating fluid problems. The investigated methods are clearly both easy to implement as their descriptions suggest. However, it is known that penalization methods prevent the use of standard fast solvers but they are at the same time good techniques for describing the embedded object with a better mathematical background and usually used to prove the existence of a solution. Therefore, we realized the implementation of the velocity penalization method into our *FEATFLOW* code and compared it with the already validated FBM method, analyzing the difficulties that arise with it and searching/finding solutions to overcome them. Like for *FBM*, the concept of being "inside" or "outside" the solid object plays the main role in developing and implementing the *PM*. We do not realize an explicit surface or interface tracking for the solid object. An immediate consequence is that quantities like hydrodynamical drag and lift forces are not straightforward to compute by using a line integration, but can be obtained by a volume integration technique over the solid object.

5.1 Fictitious boundary method

We focus in this sub-chapter to present the fictitious boundary method *FBM*, already validated by the group of S. Turek [1, 43, 50]. It is an Eulerian approach for simulating and solving Navier-Stokes equations. As all other fictitious methods, the *FBM*

is very suitable for non-stationary flows with moving or fixed solid objects, such as particulate flows, but can be applied also in the case of stationary flows with significant results. The method presents important advantages with direct impact on the total time-cost of the simulations. First of all, fluid and solid parts are treated separately, using the Navier-Stokes model for the flow part and the Newton-Euler equations for the solid part. At the same time, the boundary conditions are seen as a new constraint to the Navier-Stokes equations. The main idea is to avoid the use of unstructured grids/meshes which may be hard to obtain for different configurations and influence the entire numerical process. By extending the fluid domain over the whole domain, covering both fluid and solid parts, the use of simple, even cartesian, meshes is possible. The mesh should contain already sufficiently fine-scale geometrical details, while the large-scaled structures are described by imposing boundary parametrization. Another direct gain is the unnecessary re-meshing process in time or iterative steps. The fine-scale structures are considered as interior objects and the corresponding components are unknown *DOF*s implicitly incorporated into all iterative solution steps. For such an approach, the calculation of quantities like the hydrodynamical forces, drag and lift forces, can not be done through the known way, using a line integration, since the wall surface of the objects on whom the forces are acting is not accurately captured. A solution was presented by V. John [25], C. Duchanoy and T.R.G. Jongen [9] by using a volume integration instead the standard one. This approach will treat the disadvantage of the unpleasant implicit representation of boundaries of the solids because it does not need the reconstruction of them. Clearly some of the advantages of the *FBM* refers to the efficiency of the solver, less computational time and good accuracy. It also allows the use of fast solvers and grid deformation techniques with the aim of better approximation of the interior interfaces, or collision models to describe fairly the behavior of the objects which may move freely inside the fluid. Such techniques were already successfully implemented in the test software *FEATFLOW*.

In the following part, we describe roughly the procedure of solving fluid problems with the help of *FBM*. Consider the following system of equations with a given matrix A and right hand side f as external forces:

$$Au = f. \tag{5.1}$$

By using methods described in the previous chapters, one can obtain a weak formulation of the problem, to discretize it and finally to solve it. The finite element method requires as starting point the assembling of the weak/variational formulation of the problem. We consider a Hilbert space V embedded with a proper inner product $(\cdot, \cdot)$ and corresponding norm $|| \cdot ||$ and also $a(\cdot, \cdot)$ as bilinear form. Then one variational formulation of the problem can be described as follows:

$$\text{Find } u \in V \text{ such that: } a(u, \phi) = (f, \phi), \forall \phi \in V. \tag{5.2}$$

Next step is to discretize the obtained matrix vector equation such that a discrete linear system is obtained:

$$A_h U_h = F_h. \tag{5.3}$$

At this point, the boundary conditions can be implemented with the following representation:

$$B(u) = g. \tag{5.4}$$

This is one of the advantage of this method: it allows the implementation of boundary conditions after the problem is discretized. The matrix and vectors A_h,U_h and F_h do not contain any boundary conditions after discretization as we solve the problem on whole computational domain which consists as a reunion of fluid and solid parts. There are different possibilities to implement the boundary conditions. We refer to the fully-explicit, semi-implicit and fully-implicit treatments presented in [48] by Turek, Wan and Rivkind. In any approach the matrix A_h and vector F_h are modified in a hard-way, meaning that the matrices and vector entries/values are modified based on the boundary conditions during the iterative solving process, having as leading concept the "in or out" question. This supposes a $\mathbf{L}^2$ projection loop through out all *DOF*s from the computational domain, checking which are inside or outside the solid area/volume. For the inside *DOF*s, the *FBM* will modify the entries of the matrix and the right hand side vector accordingly to the chosen filtering technique. The

validation of the *FBM* was already done by the group of S. Turek using the software tool *FEATFLOW*. We will show in the following chapters that *FBM* is a special case of the proposed $\mathbf{L}^2$ penalty method.

5.2 Penalty method

Before presenting the work and conclusions for the *PM*, some theoretical aspects regarding this family of methods will be remembered in this sub-chapter. Penalty methods represent another class of methods to solve and simulate numerical problems of fluids with immersed boundaries. It was introduced by Courant [8] for variational calculus and applied in the context of Navier-Stokes by Temam [41]. Like in the case of fictitious methods, the idea is to solve the Navier-Stokes problem on the whole domain, penalizing the possible obstacles like, for instance, solid objects. There are lots of different penalty methods described and used during the last decades. For example the $\mathbf{L}^2$ penalization method restricts itself only to the solid obstacle area and imposes a very small velocity for that region. New velocity $\tilde{u_s}$ and pressure $\tilde{p_s}$ penalty components are introduced into the Navier-Stokes equations, with the remark that the velocity $\tilde{u_s}$ satisfies a Darcy type of law associated with a Neumann boundary condition on the pressure, hence the obstacles are considered more like a porous media. By extending the penalization to the whole linear part of Navier-Stokes equations, a $\mathbf{H}^1$ penalization method is obtained. In this case, the new velocity $\tilde{u}$ component satisfies a Brinkman type of equation associated with a Neumann type condition for the stress tensor. The theoretical and numerical validation of these two methods are presented into [2]. Other examples of penalty methods were presented by Shen [40], Bruneau [6], Hansbo [17] and others.

Maury [33] presented a simpler velocity penalization method for solving a Poisson problem. We opted for such a method in this study, penalizing only the obstacle domain into the velocity component and denote it as *PM*. As a consequence, the presence of the solid objects is not anymore realized through a filtering technique applied on the matrix system A and will be determined trough a mathematical approach. A new constant parameter is introduced, the penalty parameter $\lambda = \frac{1}{\varepsilon}$, having the characteristic of being very high ($\varepsilon = 10^{-n}, n \geq 3$) in the inner part of the solid

object/s. The penalty term is described by the relation (1.3) and hereby recalled:

$$\lambda(u-\overline{u}) = \frac{1}{\varepsilon}\chi(\mathbf{x})(u-\overline{u}) = \frac{1}{10^{-n}}\chi(\mathbf{x})(u-\overline{u}). \tag{5.5}$$

A $\mathbf{L}^2$ projection selects the group of *DOF*s which are inside the solid area. The characteristic function $\chi(\mathbf{x})$ is responsible for the activation of the penalty term. The velocity for the *DOF*s inside the solid object is denoted as $\overline{u}$ and is set to 0 if the object is in relaxation state and not 0 if it is moving. u denotes the velocity of the fluid part in this region, tending to be very small, in fact 0, due to the penalty parameter λ. We formulate now the vectorial Navier-Stokes problem implemented with the penalty term:

Find the velocity $\mathbf{u} \in H_0^1(\Omega)$ and the pressure $p \in \mathcal{L}_0^2(\Omega)$ such that:

$$\begin{cases} \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u} - \nu\Delta\mathbf{u} + \nabla p + \lambda(\mathbf{u}-\overline{\mathbf{u}}) = 0, \\ \nabla\cdot\mathbf{u}_0 = 0. \end{cases} \tag{5.6}$$

The problem may be paired with any kind of boundary conditions: initial, Dirichlet, Neumann, mixed, and so on. We follow the same path presented in the previous chapters of solving such a problem, obtaining a variational formulation and discretize it. For the easiness of the situation, we consider just a 2D dimensional case and a fixed solid object, therefore $\overline{u} = 0$. The scalar equation for the velocity component $u_\alpha \in H_0^1(\Omega)$, $\alpha = \{\mathrm{x}, \mathrm{y}\}$ is the following:

$$\begin{cases} \dfrac{\partial u_\alpha}{\partial t} + (\mathbf{u}\cdot\nabla)u_\alpha - \nu\Delta u_\alpha + \dfrac{\partial p}{\partial \alpha} + \dfrac{1}{\varepsilon}\chi(\mathbf{x})u_\alpha = 0, \\ \\ \displaystyle\sum_\alpha \frac{\partial u_\alpha}{\partial \alpha} = 0. \end{cases} \tag{5.7}$$

The definition (3.19) and (3.20) are useful for the linear, bilinear and trilinear forms, helping to develop a continuous variational formulation of the problem as it follows:

Find $\mathbf{u} \in H_0^1(\Omega)$ *and* $p \in \mathcal{L}_0^2(\Omega)$ *such that:*

$$\begin{cases} \left(\omega, \dfrac{\partial u_\alpha}{\partial t}\right) + \frac{1}{\varepsilon}\chi(\mathbf{x})\,(\omega, u_\alpha) + n\,(\mathbf{u}, \omega, u_\alpha) + b_\alpha\,(\omega, p) = 0,\ \forall \omega \in H_0^1(\Omega), \\ \sum\limits_\alpha b_\alpha(u_\alpha, q) = 0,\ \forall q \in \mathcal{L}_0^2(\Omega). \end{cases} \tag{5.8}$$

In the spirit of the finite element method, one defines the polynomial finite element spaces $V_n \subset H_0^1(\Omega)$ and $Q_n \subset \mathcal{L}_0^2(\Omega)$ such that both velocity and pressure can be approximated by the N piece-wise polynomial basis functions $\{\varphi_1, \varphi_2, \ldots, \varphi_{N_u}\}$, respectively $\{\psi_1, \psi_2, \ldots, \psi_{N_p}\}$.

$$\begin{aligned} u_{\alpha,n}(\mathbf{x}, t) &= \sum_{j=1}^{N_u} u_{\alpha_j}(t)\varphi_j(\mathbf{x}) \in V_n, \\ p_n(bx, t) &= \sum_{k=1}^{N_p} p_k(t)\psi_k(\mathbf{x}) \in Q_n, \end{aligned} \tag{5.9}$$

where N_u and N_p are the degrees of freedom for velocity and pressure. The discrete problem can be written:

Find $u_n \in V_n$ and $p_n \in Q_n$ such that:

$$\begin{cases} (\omega_h, \frac{\partial u_{\alpha,h}}{\partial t}) + \frac{1}{\varepsilon}\chi(\mathbf{x})(\omega_h, u_{\alpha,h}) + n(\mathbf{u}_n, \omega_h, u_{\alpha,h}) + b_\alpha(\omega_h, p_n) = 0,\ \forall \omega_h \in V_h, \\ \sum\limits_\alpha b_\alpha(u_{\alpha,h}, q_h) = 0,\ \forall q_n \in Q_n. \end{cases} \tag{5.10}$$

The corresponding matrix form of the discrete equations after applying also a time-step technique for time discretization, is then given by:

$$\begin{cases} \mathbf{M}\mathbf{u} + \widetilde{\mathbf{M}}\mathbf{u} + \mathbf{N}(\mathbf{u}) \cdot \mathbf{u} + \mathbf{B}p = 0, \\ \mathbf{B}^T\mathbf{u} = 0. \end{cases} \tag{5.11}$$

For an moving solid object, the right hand side will not be anymore 0, but of the form $\widetilde{\mathbf{M}}\overline{u}$. Comparing with the standard matrix representation of a Navier-Stoke problem, a new matrix appears: $\widetilde{\mathbf{M}}$. We call it penalty matrix and is nothing else than a modified mass matrix. All its entries are multiplied by the characteristic function χ, such that nonzero values are obtained only in the case of the *DOF*s which are inside the solid object or penalized volume. The size of these entries depends on the *varepsilon*-penalty parameter ε, but also on the size of local finite element h to the

square, which can be rewritten as $\mathrm{O}\left(\frac{1}{\varepsilon}h^2\right)$.

We propose different implementations of the *PM*, which are consequences of the observations regarding the behavior of the method and the difficulties that have arised during the numerical test cases. We denote the main implementation technique as *full-lambda PM*, characterized by a constant value of penalty parameter λ and activated only for the quadrature points inside the solid object. It might happen in a finite element approach that the solid object has only a common intersection with the finite element cell and does not fully include it. For this reason, depending on the quadrature formula used to assemble the mass matrix, only some of the cubature points will have activated the penalty parameter, while the others will have a corresponding 0 value. Therefore, a discontinuity arises in the process of computing the integral. We proposed as a solution against the discontinuity a *fractional-lambda PM*, characterized by an average value of penalty parameter λ for all cubature points which are inside one finite element that has contact with the solid object. The technique leads to a diffusive capturing of the interface, but for finer levels of the mesh this problem is neglectable. The gain is that the integration will not be discontinuous anymore, hence the assembling process of the penalty matrix is mathematically correct. We were able to show that despite the discontinuity, the *full-lambda* method gives anyway better results. Another aspect is the use of different kind of meshes: cartesian versus body-fitted. We were able to show that a body-fitted mesh will provide better results since the capturing of the interfaces is more accurate. Further on we implemented also an *adaptive-lambda* method to investigate if the solution improves by getting the boundaries of the solid object much better. The immediate disadvantage are the longer time of computations, so the lost in efficiency of the solver, without a clear improvement of the results comparing with the main *PM*. Also, due to $\widetilde{Q_1}/Q_0$ element used in our finite element approach, in the mass matrix might happen that off-diagonal entries are negative and the standard lumping techniques can not be efficiently used. This is due to the combination of the density jump and non-positiveness of parts of the basis functions which may create dominating off-diagonal entries according to Hysing [23]. Instead of standard lumping, an HRZ-lumping method can be assembled, presented by Hinton, Rock and Zienkienwicz in [19]. The method supposes a diagonal lumping of the local matrices by eliminating the non-positive entries and followed by a scaling

such that the total mass of the elements is preserved. We successfully implemented this method with our *PM*, with improvements regarding the linear solver efficiency, but on the other hand no accuracy changes. All configurations of the penalty method and its variants will be detailed presented later in this chapter together with numerical results for the benchmark test of flow around cylinder.

5.3 Test case of flow around cylinder

We consider the stationary benchmark case of a 2D flow around cylinder in a channel. For the referential values we choose the finite element *cc2d* source code from *FEATFLOW* software, solving the problem in a fully coupled way. We use in the performed simulations two kind of meshes: a cartesian type and a body-fitted type. The statistical data of the benchmark test are the following: a channel of height $H = 0.41$ and length $L = 2.2$ contains a disk of diameter $D = 0.1$, centered at position $O(x_0, y_0) = (0.2, 0.2)$; the inflow profile is parabolic and described by the equation:

$$U(0, y, t) = 6.0 \cdot \overline{U} \cdot \frac{y \cdot (H - y)}{H^2}. \tag{5.12}$$

$\overline{U}$ represents the mean velocity, is a given parameter and makes the differentiation between laminar and transient flows, classified by the values of Reynold number. The relation $Re = \frac{\overline{U} \cdot D}{\nu}$ defines the Reynold number. In the case of steady flow, the mean velocity is set to $\overline{U} = 0.2$ which is an equivalent of Reynold number equal 20, transient flow. We recall that ν is the viscosity of the fluid given by $\nu = \frac{\mu_f}{\rho_f}$, while ρ_f is the density of the fluid and consider to be 1 in our simulations. The simple *cc2d* code requires the use of an unstructured mesh, denoted as *BENCH1* and presented in figure 5.1.

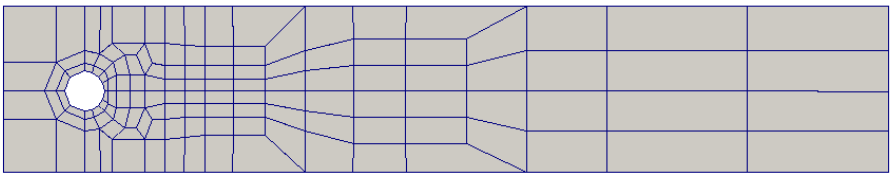


Figure 5.1: Unstructured mesh (Level 1) for *cc2d*

The statistical informations of *BENCH1* grid are given in the next table:

LEVEL	NVT	NMT	NEL	NEQ
1	156	286	130	702
2	572	1092	520	2704
3	2184	4264	2080	10608
4	8528	16848	8320	42016
5	33696	66976	33280	167232
9	8526336	17046016	8519680	42611712

Table 5.1: Statistical data for the unstructured grid *BENCH1*

The notation correspond to the numbers of vertices (NVT), the number of midpoints (NMT), the number of elements (NEL) and the number of unknowns (NEQ). The level denotes the degree of refinement of the grid, a higher level being obtained by connecting the midpoints of opposite edges of one element and constructing in this way other smaller 4 elements from one bigger.

We focused in our investigation firstly on point values solutions. Quantitative measurements for velocity and pressure were calculated in selected points of the computational domain, given by the following cartezian coordinates: $P_1(0.15; 0.2)$, $P_2(0.25; 0.2)$, $P_3(0.1375; 0.2)$ and $P_4(0.2625; 0.2)$ for the pressure evaluation, respectively $P5(0.4; 0.2)$ and $P_6(0.65; 0.2)$ for the velocity. In other words, we test the solutions of different methods in the selected points, which are situated as follows: 2 on the interface of the cylinder, 2 in the near vicinity (for pressure) and 2 in the rear region of the cylinder (for the velocity). Since there are no reference values regarding point values of the solution for flow around cylinder in the enumerated points, we

have chosen as reference value the results given by *cc2d* with *BENCH1* on level 9 of refinement. The *cc2d* is already validated and the accuracy increases as the number of the degrees of freedom are bigger, this being similar with the increase of the mesh level.

	P_1	P_2	P_3	P_4
Level	$\times 10^{-1}$	$\times 10^{-2}$	$\times 10^{-1}$	$\times 10^{-2}$
9	1.32220	1.47560	1.21158	1.38843

Table 5.2: Referential point values for the pressure component

	P_5		P_6	
	u_x	u_y	u_x	u_y
Level	$\times 10^{-2}$	$\times 10^{-3}$	$\times 10^{-1}$	$\times 10^{-4}$
9	5.05319	2.01330	2.01724	8.42345

Table 5.3: Referential point values for the velocity components

The reference values for the Drag and Lift forces will be presented in a following chapter.

5.3.1 Fictious Boundary Method values

For the case of *FBM* and *PM* we do not need to use unstructured meshes as the one in the figure 5.1 and we had the free choice of selecting a coarser mesh, with less number of vertices, midpoints, elements and unknowns. Although, the matter of accuracy of the solution should be still compared between the different methods for meshes who have approximative similar settings as the reference one, namely same order of number of unknowns. In our numerical simulations we used two kind of meshes: a cartesian *BENCH2M55* presented in figure 5.2 and a body-fitted *BENCH1_FBM* presented in figure 5.3. The second grid contains more geometrical details around the region of the cylinder and it is derived from the reference *BENCH1* grid. All meshes were created and validated with the *DEVISOR GRID 3D* program.

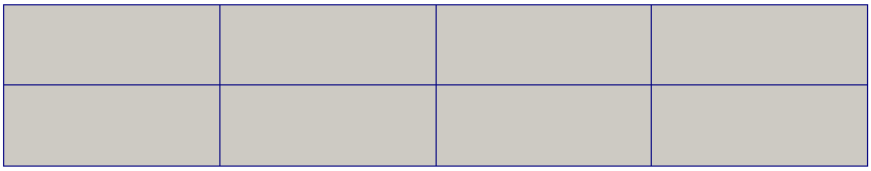


Figure 5.2: Cartesian mesh *BENCH2M55* (Level 2)

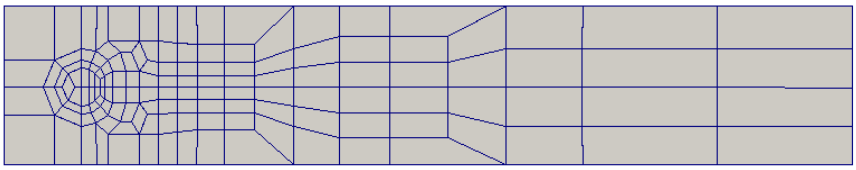


Figure 5.3: Body-fitted mesh *BENCH1_FBM* (Level 1)

We have performed simulations on different levels of refinement, keeping the number of unknowns on the same order. In this thesis we present results for only 2 levels: 8 and 9 for *BENCH2M55* and 6 and 7 for *BENCH1_FBM*. The simulations were run on the same machine, such that the methods could be compared also from the point of CPU-times. Firstly, the statistical data of the used meshes are below:

Level	NVT	NMT	NEL	NEQ
9	131841	262912	131072	656896
10	525825	1050112	524288	2624512
12	8394753	16783360	8388608	41955328

Table 5.4: Statistical data for the cartesian grid *BENCH2M55*

In the following tables are the point values for pressure and velocity obtained by using the *FBM*, compared to the referential values of steady *cc2d*.

Level	NVT	NMT	NEL	NEQ
6	142977	285312	142336	712960
7	570625	1139968	569344	2849280
9	9114625	18224128	9109504	45557760

Table 5.5: Statistical data for the body-fitted grid *BENCH1_FBM*

	P_1	P_2	P_3	P_4	$P_{5_{u_x}}$	$P_{5_{u_y}}$	$P_{6_{u_x}}$	$P_{6_{u_y}}$
Lvl	$\times 10^{-1}$	$\times 10^{-2}$	$\times 10^{-1}$	$\times 10^{-2}$	$\times 10^{-2}$	$\times 10^{-2}$	$\times 10^{-1}$	$\times 10^{-4}$
ref.	1.32220	1.47560	1.21158	1.38843	5.05319	2.01330	2.01724	8.42345
6	1.31968	1.47101	1.20771	1.38489	5.07732	2.01914	2.01815	8.40870
7	1.32086	1.47410	1.20997	1.38737	5.06596	2.01264	2.01772	8.41199

Table 5.7: Point values with *FBM* on *BENCH1_FBM*

	P_1	P_2	P_3	P_4	$P_{5_{u_x}}$	$P_{5_{u_y}}$	$P_{6_{u_x}}$	$P_{6_{u_y}}$
Lvl	$\times 10^{-1}$	$\times 10^{-2}$	$\times 10^{-1}$	$\times 10^{-2}$	$\times 10^{-2}$	$\times 10^{-2}$	$\times 10^{-1}$	$\times 10^{-4}$
ref.	1.32220	1.47560	1.21158	1.38843	5.05319	2.01330	2.01724	8.42345
9	1.28833	1.44534	1.19404	1.39721	5.13692	1.98582	2.01974	8.48419
10	1.30274	1.45759	1.20206	1.39419	5.13447	1.98222	2.02011	8.31398

Table 5.6: Point values with *FBM* on *BENCH2M55*

We use the relative error (5.13) with respect to the reference values such that we get a qualitative and quantitative comparison for the solution provided by the fictitious method:

$$\varepsilon_{rel} = \frac{|V_{ref} - V_{fbm}|}{V_{ref}} \tag{5.13}$$

where V_{ref} is the reference value and V_{fbm} is the corresponding point value provided by *FBM*.

As table 5.8 is showing, the absolute errors are under 3% for the coarser levels and decrease for the finer ones. We acknowledge that the body-fitted mesh is giving much

Lvl	P_1	P_2	P_3	P_4	$P_{5_{u_x}}$	$P_{5_{u_y}}$	$P_{6_{u_x}}$	$P_{6_{u_y}}$
bench2m55								
9	2.56	2.05	1.45	0.63	1.66	1.36	0.12	0.72
10	1.47	1.22	0.79	0.41	1.61	1.54	0.14	1.30
bench1_fbm								
6	0.19	0.31	0.32	0.25	0.48	0.29	0.05	0.18
7	0.10	0.10	0.13	0.08	0.25	0.03	0.02	0.14

Table 5.8: Relative error for *FBM* with respect to *cc*2*d*

better point solutions due to better approximation of the interface of the solid object. As a matter of fact, one can use a cartesian mesh which is only body-fitted in the region where the object is located to obtain improved results as the ones generated by only using a simple cartesian mesh. In the case of relaxed objects, not moving, with simple geometry, it is easy to create such grids, but when the objects freely move inside the domain, then moving or deformation grid strategies are required. It also possible, for fast results, to use a cartesian mesh which is finer and has more elements just in the region where the object/s is/are moving, if its/their movement can be predicted or is prescribed.

5.3.2 Penalty method generalization of FBM

Starting point of this this study was the motivation of finding a method which can be considered as generalization of the validated *FBM*. We claim that the proposed *PM* encounters this attribute, with a better mathematical background. We will show in this sub-chapter that, indeed, *PM* will reproduce the solution of *FBM* under certain configurations. The candidate method for this purpose is the main full-lambda implementation of the *PM*, with the completion that a Gauss 3-by-3 quadrature formula is used in the process of assembling the modified penalty mass matrix. The results will show the importance of the penalty parameter λ, which leads to small values for velocity in all nodal points which are inside the penalty area. We use the the nonparametric and non-conforming $\widetilde{Q_1}/Q_0$ Rannacher-Turek element with the two variations *EM30* and *EM31* for the $\widetilde{Q_1}$ discretization. In the definition (3.40) of

the nodal functionals of this element, the mean value of the velocity along the edge is assigned to the *EM30* element and the value of the velocity in the midpoint of the edge is assigned to the *EM31*. The fictitious boundary approach consists of a filtering technique over the already discretized system matrix to determine the set of $Dofs$ for which the wanted boundary conditions should be imposed. There is the possibility of an explicit, semi-implicit and implicit treatment of the boundary conditions. If we denote by S_h the set of such $Dofs$, then all rows and columns of the discretized matrix A_h belonging to the set of $Dofs$ S_h are eliminated and the right hand side F_h is accordingly modified in an explicit approach. On the other hand, a semi-implicit treatment supposes a substitution of those rows in A_h which correspond to S_h by rows of the identity matrix, while all other rows and columns remain unmodified. However, the given Dirichlet for all components of the velocity vector U_h and right hand side F_h which belong to S_h have to be prescribed. The implicit technique is performing before and after each iteration step this prescription of the given Dirichlet values. In any situation, the *EM30* element is used for assembling the component matrices. In order to reproduce the results of *FBM* by using *PM* technique, we perform the following settings: assemble the penalty matrix with *EM31* element and all other components matrices with *EM30*. Choosing as quadrature rule for assembling the penalty matrix the midpoint rule *MID2D*, then the final system matrix will be similar like in the case of *FBM*. Hence, we could show in this manner that *PM* can reproduce the solution of *FBM*. Moreover, if the penalty parameter is very high and the penalty matrix is row-wise scaled with respect with the diagonal entry, then the system matrices are identical for both methods. The tables 5.9 and 5.10 state the possibility to reproduce the solution of *FBM* by only using special settings in the discretization process of *PM*. Therefore, we consider justified the idea that the proposed *PM* is a generalization of the fictitious boundary method.

In tables 5.9 and 5.10 we illustrate the identity for both meshes used during the simulations and for different level of refinement. However, we only present here results for few levels and for several ε-penalty parameter $(10^{-3}, 10^{-6}, 10^{-9})$. For coarser levels, the size of the ε-penalty parameter has to be very small in order to reach identical results like *FBM*, but for finer levels same results are obtained for even smaller ε-penalty parameter. Regarding the free choice of the ε-penalty parameter,

	P_1	P_2	P_3	P_4	U_1	V_1	U_2	V_2
	$\times 10^{-1}$	$\times 10^{-2}$	$\times 10^{-1}$	$\times 10^{-2}$	$\times 10^{-2}$	$\times 10^{-3}$	$\times 10^{-1}$	$\times 10^{-4}$
FBM	1.3512	1.4386	1.1862	1.4128	5.2310	1.9370	2.01950	8.3415
$PM10^3$	1.2628	1.4165	1.1666	1.3879	5.1898	1.9434	2.0161	8.4371
$PM10^6$	1.3516	1.4385	1.1861	1.4127	5.2307	1.9370	2.0194	8.3418
$PM10^9$	1.3512	1.4386	1.1862	1.4128	5.2310	1.9370	2.0195	8.3415

Table 5.9: Results of *FBM* and *PM* on *BENCH2M55* Level 8

	P_1	P_2	P_3	P_4	U_1	V_1	U_2	V_2
	$\times 10^{-1}$	$\times 10^{-2}$	$\times 10^{-1}$	$\times 10^{-2}$	$\times 10^{-2}$	$\times 10^{-3}$	$\times 10^{-1}$	$\times 10^{-4}$
FBM	1.3164	1.4620	1.2026	1.3773	5.0912	2.0079	2.0186	8.3626
$PM10^3$	1.2900	1.4314	1.1878	1.3659	5.1907	2.0002	2.0210	8.3101
$PM10^6$	1.3163	1.4617	1.2025	1.3773	5.0916	2.0079	2.0186	8.3623
$PM10^9$	1.3164	1.4620	1.2026	1.3773	5.0912	2.0079	2.0186	8.3627

Table 5.10: Results of *FBM* and *PM* on *BENCH1_FBM* Level 5

we have to specify that is actually limited by the mesh size parameter h. As other studies show [2], the numerical error estimates due to the penalty term are of order $\|u - u_\varepsilon\| = O(\varepsilon)$. Angnot et. all proved that the theoretical error estimates should be $O(\varepsilon^{3/4})$ inside the object and $O(\varepsilon^{1/4})$ in the fluid. The numerical results show anyway a even better estimate of $O(\varepsilon)$. However, the error due to the discretization $\|u - u_h\|$ will cover the error due to the penalty term if the penalty parameter is very small ($\varepsilon << h$) and therefore, usually there is not need to set the ε-penalty term to small. From this reason, we set mostly our penalty parameter values in the range $\varepsilon \in [10^{-3}, 10^{-6}]$, but also smaller values are possible. That is an advantage, since a combination of finer level and small penalty parameter lead to numerical problems, bad efficiency of the solver and sometimes, without smoothing techniques, no convergence can be reached.

5.3.3 Volume approximation with Penalty Method

It is a fact that also in the case of *PM* the capture of the boundary-walls of the solid object is usually not accurate enough. Actually, for coarser levels of refinement of the grid, the capture of the interface is leading to approximation errors which find themselves also in the final solution. As the level increases, the capturing becomes more accurate. If numerical quantities like drag and lift forces are necessary, they are usually calculated by using a line integration along the boundary of the object for which the calculation is done. Obviously, *PM* is not expected to be able to provide good and accurate results for these forces if a line integration is applied. Hence, the new direction that has to be taken is a volume integration. In this sub-chapter we focus to show that the volume approximation of the solid object by *PM* is very well done. This is done with the help of the penalty matrix itself, since it is a mass matrix which keeps in only the corresponding entries for the inside *DOF*s. All other will are equal to zero. In the test case, the static object is a circle with radius $r = 0.05$ in a 2D case. Therefore, the volume is nothing else but the area of a circle with this radius and given by:

$$A_{ob} = \pi \cdot r^2 \simeq 7.85398 \times 10^3. \tag{5.14}$$

We performed calculations for both meshes: cartesian and body-fitted and observed that in the case of a body-fitted kind of mesh the approximation of the volume is better for any level (0.0004% for level 6) in comparison to the cartezian one (0.0122% for level 9). This is due to much finer meshes around the solid object. As the refinement level increases, the approximation of the volume becomes better also in the case of the cartezian mesh (0.0045% on level 12), but it will not reach the same accuracy like the body-fitted mesh (0.0008% on level 9). So even if the time-computation is longer and the efficiency of the solver might decrease, it is motivating to use a mesh which is already containing many geometrical informations in the vicinity of the solid object since the mass matrix is better approximated. We claim that the final solution of any problem numerically solved by *PM* will be improved in the case of a body-fitted mesh. In the table (5.11) we present results only for the main *full-lambda* implementation for both kind of meshes and for $\varepsilon = 10^3$. The calculations presented in here, also prove

that the penalty matrix is correct and accurate enough, assembled and implemented into the total matrix A.

Value	7.85398×10^{-3}		
bench2m55		bench1_fbm	
Level	Value	Level	Value
7	7.89579×10^{-3}	4	7.85016×10^{-3}
8	7.84370×10^{-3}	5	7.85110×10^{-3}
9	7.85520×10^{-3}	6	7.85402×10^{-3}
10	7.85555×10^{-3}	7	7.85402×10^{-3}
11	7.85303×10^{-3}	8	7.85401×10^{-3}
12	7.85353×10^{-3}	9	7.85399×10^{-3}

Table 5.11: Volume approximation by *PM*

5.3.4 Penalty Method values

As already mentioned, we have opted for several implementations of the Penalty Method. At the base of the classification are the Penalty matrix assembling process, the penalty parameter settings and the quadrature formulas used for the calculation of the integrals. Since it is a modified mass matrix and basically for all entries the relation (4.3) is involved, one has to take care that the chosen quadrature formula for the approximation of the integrals is at least of order 2. Therefore, we couple in general the penalty matrix assemble process with the use of the Gauss 3-by-3 quadrature formula, denoted in our study as *G3x3*, which uses 9 cubature points inside each finite element of the grid and has the order 6. Another option is the possibility of using an adaptive quadrature formula. Regarding this second option, we will come into it with more details at a later point. For the moment we specify the importance of selecting the elements which only intersect the boundary of the solid object. For them we apply a preferential handling of the mass matrix assembly. Locally we consider a grid refinement together with the use of a simple quadrature formula, like the trapezoidal rule TRZ_n, for all elements of the new element sub-grid. In here, n stands for the local refinement level, but we specify that the mesh itself does not suffer any physical modifications and the entire process of local refinement

is virtual and contained in the calculations of the entries. The trapezoidal rule is of second order, therefore it is enough to approximate the mass matrix. Both quadrature formulas are given for the 1D case by the equations (5.15) and (5.17). In there, $\omega(\cdot)$ is the weight function, $\phi(\cdot)$ is a function which can be approximated by a polynomial in the selected points x_k such that $f(x) = \omega(x)\phi(x)$. α_k are the weights and defined by (5.16). Being a Gauss quadrature formula, the two dimensional *G3x3* is of order 6. The trapezoidal quadrature formula has always the order 2 and for a non-uniform grid the h value is given by the difference between neighboring points: $h = (x_{k+1} - x_k)$. The explicit use of the adaptive formula will be described later.

$$\int_a^b f(x)dx = \int_a^b \phi(x)\omega(x)dx \simeq \sum_{k=1}^{N} \phi(x_k)\alpha_k \tag{5.15}$$

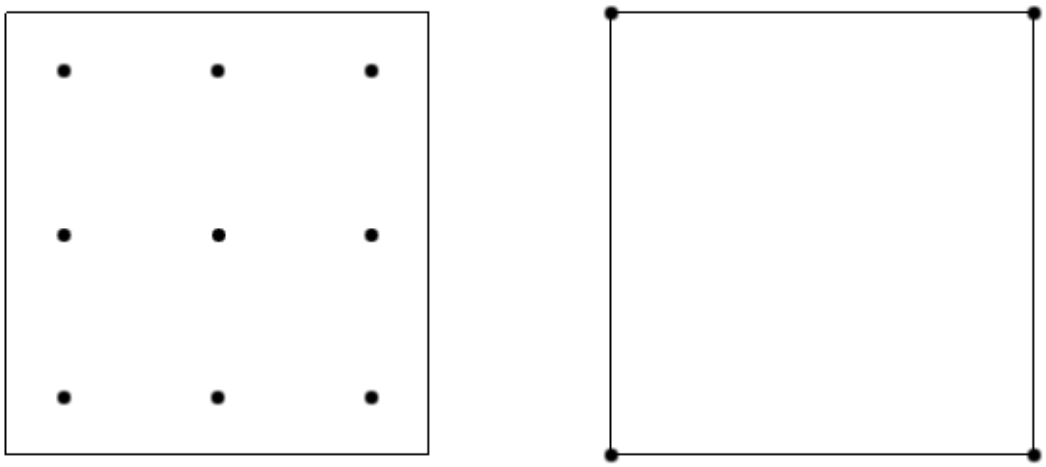
$$\alpha_k = \int_a^b \omega(x) \prod_{j=1, j\neq k}^{N} \frac{x - x_j}{x_k - x_j} dx \tag{5.16}$$

$$\int_a^b f(x)dx = (b-a)\frac{f(a) - f(b)}{2} \simeq \frac{h}{2}\sum_{k=1}^{N}(f(x_{k+1}) - f(x_k)) \tag{5.17}$$

We consider the reference element $[-1, 1] \times [-1, 1]$ and want to apply in the bi-dimensional case the integration by using the *G3x3* quadrature formula. It follows the rule:

$$I = \int_{-1}^{1}\int_{-1}^{1} f(s,t)dsdt \approx \sum_{i=1}^{N}\sum_{j=1}^{M} \alpha_i\alpha_j f(s_i, t_j) = \sum_{i=1}^{N}\sum_{j=1}^{M} W_{ij} f(s_i, t_j). \tag{5.18}$$

The reference coordinate system has the origin in the central cubature point and has the coordinates $(0, 0)$. The rest of the points are located as it follows: (a, a), $(-a, a)$, $(-a, -a)$, $(a, -a)$, $(0, a)$, $(-a, 0)$, $(0, -a)$, $(a, 0)$, where $a = \sqrt{\frac{3}{5}}$. The corresponding weights in 2D with some change of notations are: $W_1 = \frac{64}{81}, W_2 = W_3 = W_4 = W_5 = \frac{25}{81}$ and $W_6 = W_7 = W_8 = W_9 = \frac{40}{81}$. The calculations with the trapezoidal rule are simpler since the cubature points are situated into the vertices of the reference square element. Then the 2D integration reads:



G3x3 TRZ_0

Figure 5.4: Quadrature points

$$I = \int_{-1}^{1}\int_{-1}^{1} f(s,t)dsdt \approx \frac{1}{4}h^2 \sum_{i=1}^{N}\sum_{j=1}^{M} f(s_i,t_j). \tag{5.19}$$

To sustain the several implementations of the *PM*, we want to clarify why we took them into consideration. Starting with the main implementation full-lambda coupled with *G3x3*, some questions had to be answered. Is the matrix provided with this method accurate enough? Is the interface pleasantly approximated? To get the answers, we applied two other approaches. With the fractional-lambda method the discontinuity in the elements which are not completely inside or outside the solid object vanishes, while with the adaptive cubature technique we aimed a better approximation of the interface and in the same time a reduction of the discontinuity. The issues were generated by the elements which only had a common contact part with the solid region. For such elements it might happen that not all quadrature points are inside, hence the characteristic function could be equal 1 (in) and 0 (out) within the same element. We use the following simple idea to overcome the discontinuity as it follows. Firstly, we get the intersection points between the solid and the finite elements which are not completely in- or outside. Secondly we calculate the area of this common part and divide it by the area of the element such that a fractional

part f is obtained. The last step is to multiply the value of the penalty parameter λ with f and set the penalty parameter element-wise constant. This implies a constant value for the characteristic function $\chi(\cdot)$ equal to 1 for all elements which intersect the solid object and 0 for the others. If the shape of the particle is a simple one, then the desired area is easy to compute. However, we usually deal with objects of different forms, and their area can not be analytically calculated. So we took the convention to calculate only the area of the polygon determined by the identified intersection points. In figure 5.5 this is the triangle $\Delta P1P2P3$. Depending on the number of the intersection points, we can use the general formula for a polygon with n nodes:

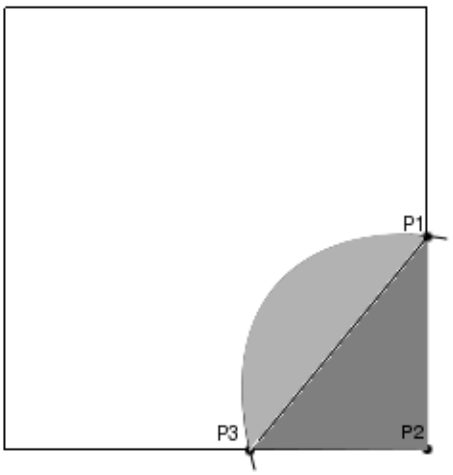


Figure 5.5: Intersection between an element and solid object

$$A_p = \frac{1}{2} \cdot \sum_{i=1}^{n} (x_i y_{i+1} - y_i x_{i+1}). \tag{5.20}$$

By choosing this approach we took the risk of an inaccurate solid interface capture and therefore the simulations should be performed for higher levels of refinement of the mesh, such that this inconvenience is significantly reduced. Another idea was to calculate the integrals by using the adaptive quadrature formula, which lowers the impact of the discontinuity. Firstly we determine which elements of the grid are completely included into the penalty region and assemble the local mass matrix by using $G3$x3. Then, for the elements without any common region with the solid object we can use any quadrature formula since any DOFs inside them will not be

activated by the characteristic function and therefore the corresponding entries of the matrix will be zero. The only elements remained are the one which have contact part with the solid. For them we apply the adaptive cubature formula as it follows: we establish a local refinement level n and for all local created elements we calculate local mass matrix entries by using the TRZ_n. Then we sum them up to obtain the corresponding mass matrix entries for the entire element. The refinement only appears in the calculation and is not hardly imposed into the mesh, meaning that the mesh suffers no modifications. The level of refinement is 2^{2n} and is shown in the figure (5.6) together with the cubature points for TRZ_n. The remaining elements, completely included into the solid object, will be treated as in the full-lambda approach, by using the *G3x3* quadrature formula.

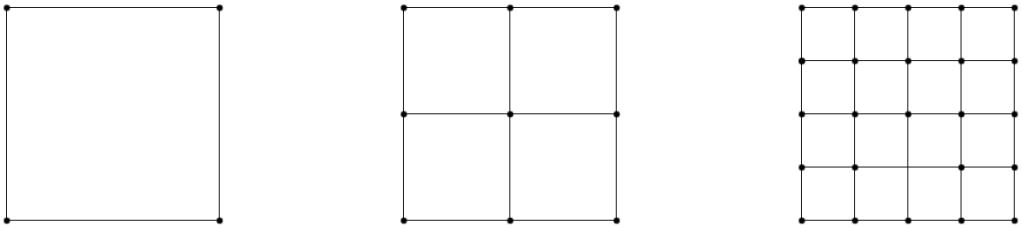


Figure 5.6: Refinement for adaptive quadrature formula

With this approach the boundary of the solid object is better captured even for coarser meshes. In conclusion we have the full- and fractional-lambda *PM*s which can be both coupled with the *G3x3* or TRZ_n quadrature formulas. However, we preferred to apply the adaptive technique only in the case of full-lambda and denote it as adaptive-lambda *PM*. Moreover, we can also use the *HRZ* lumping method in combination with any of the proposed *PM*s. The use of this lumping technique is justified by the fact that in a $\widetilde{Q_1}$ discretization, negatives entries on the off-diagonal position can be generated into the matrix. With the *HRZ* lumping, these negative entries are eliminated together with a scaling process of all other entries to preserve the total mass. The following tables represent point values for velocity and pressure in the selected points $P_i,\ i = \overline{1,6}$, for both meshes on the levels 6,7 for *BENCH1_FBM*, respectively levels 9,10 for *BENCH2M55*. We present the results for ε-penalty parameter values of $\varepsilon \in \{10^{-3}, 10^{-6}\}$, with $\lambda = \frac{1}{\varepsilon}$, in comparison with the referential values

written in red. Higher values a also possible to use, but reasons like computer power, CPU-times and so on limit the free-choice of the ε-parameter. Moreover, for such values, due to a bad conditioned matrix, the convergence was not obtained without imposing strong stabilization techniques and/or pre/post-smoothing step methods. This is due to the size of the penalty parameter λ and the mesh size h which are involved in the calculation of the integrals, leading to very high values in specific cases. That is a typical problem for penalty methods in general. However, for this test case, even values of order 10^3 are sufficient as we shall see in the tables of values.

	P_1	P_2	P_3	P_4	$P_{5_{u_x}}$	$P_{5_{u_y}}$	$P_{6_{u_x}}$	$P_{6_{u_y}}$
	$\times 10^{-1}$	$\times 10^{-2}$	$\times 10^{-1}$	$\times 10^{-2}$	$\times 10^{-2}$	$\times 10^{-3}$	$\times 10^{-1}$	$\times 10^{-4}$
ref.	1.32220	1.47560	1.21158	1.38843	5.05319	2.01330	2.01724	8.42345
BENCH2M55 Level 9								
10^3	1.27399	1.43533	1.17931	1.38165	5.14418	2.01628	2.01752	8.45256
10^6	1.29799	1.49283	1.21804	1.40866	4.89248	1.95608	2.01078	8.50309
BENCH2M55 Level 10								
10^3	1.28832	1.44034	1.18565	1.37979	5.19552	2.00359	2.02069	8.34252
10^6	1.31871	1.46644	1.21289	1.39318	4.98575	2.01194	2.01452	8.44657
BENCH1_FBM Level 6								
10^3	1.29219	1.44016	1.19133	1.37563	5.20891	2.00301	2.02178	8.32103
10^6	1.32022	1.45324	1.21362	1.38512	5.01783	2.01462	2.01585	8.44264
BENCH1_FBM Level 7								
10^3	1.29489	1.44557	1.19210	1.37831	5.21330	1.99886	2.02195	8.31689
10^6	1.32034	1.46588	1.21183	1.38761	5.04924	2.01060	2.01707	8.42011

Table 5.12: Point values for full-lambda *PM*

It is obvious from the table of values that the full-lambda *G3x3* provides accurate results for the body-fitted mesh and for higher penalty parameters. The results obtained with a λ-penalty parameter of 10^3 are also pleasant, but the penalty object, namely the solid object, behaves in this case like it has penetrable boundaries and the inside velocities can be bigger then zero, transforming it into a porous media. This loss could be reason of errors in the point values near the object and in the rear

part of it, because of incorrect velocity and pressure distribution. So, it is necessary to set the penalty parameter as high as possible, to ensure that the object is strongly penalized and behaves like a solid, meaning that the fluid velocity is tending to zero inside it.

	P_1	P_2	P_3	P_4	$P_{5_{u_x}}$	$P_{5_{u_y}}$	$P_{6_{u_x}}$	$P_{6_{u_y}}$
BENCH2M55 Level 9								
10^3	3.65	2.73	2.66	0.49	1.80	0.15	0.01	0.35
10^6	1.83	1.17	0.53	1.46	3.22	2.84	0.32	0.95
BENCH2M55 Level 10								
10^3	2.56	2.39	2.14	0.62	2.82	0.48	0.17	0.96
10^6	0.26	0.62	0.11	0.34	1.33	0.07	0.13	0.27
BENCH1_FBM Level 6								
10^3	2.27	2.40	1.67	0.92	3.08	0.51	0.23	1.22
10^6	0.15	1.52	0.17	0.24	0.70	0.07	0.07	0.23
BENCH1_FBM Level 7								
10^3	2.07	2.04	1.61	0.73	3.17	0.72	0.23	1.27
10^6	0.14	0.66	0.02	0.06	0.08	0.13	0.01	0.04

Table 5.13: Relative errors of full-lambda *PM* w.r.t. reference values

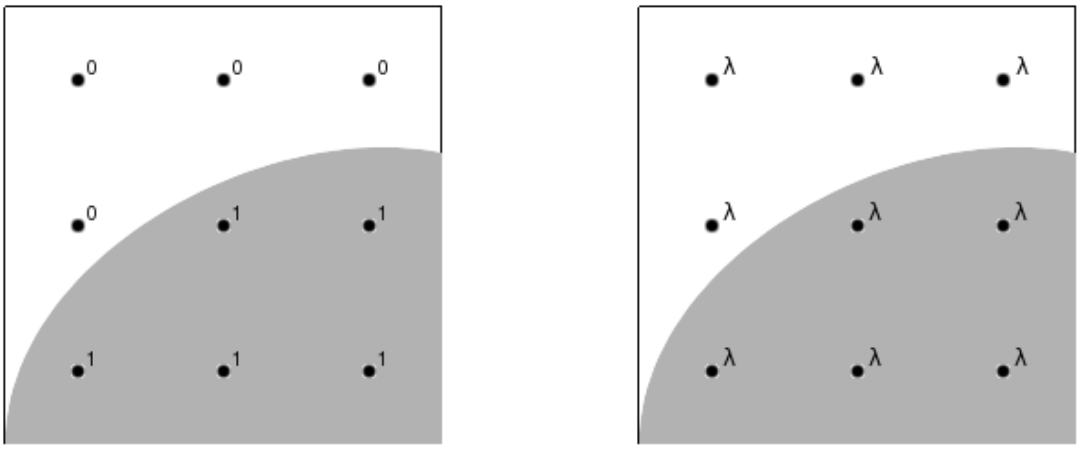
As we specified before, we can also perform simulations for even higher values, i.e 10^9, but such values determine ill-conditioned matrix and numerical problems arise for the linear solver. However, there are techniques to use against this inconvenient fact. One solution against it is to diagonalize the penalty matrix and scale it with respect to the highest diagonal entry. This procedure will permit the free-choice of the penalty parameter, so it can be arbitrarily selected, as high as we want, with the property that the problem still converges to accurate solutions. Using high penalty parameters could lead to diverging problems which can be avoided by pre- and post-smoothing techniques. This will have also direct impact onto the computational time and the efficiency of the solvers in general. The nonlinear solver is not at all influenced by the penalty matrix, since it does not interfere with the nonlinear term.

In the main method, full-lambda *PM* coupled with a *G3x3* quadrature formula, because of using a finite element approach, it happens often that the cubature points belonging to one element will be treated different. If we analyze a possible situation like in the example from the figure 5.7, one can see that only 5 cubature points inside the solid area have an activated penalty parameter in the calculation of the matrix entries, while the other 4 will have a corresponding value of 0. Thus, inside one element can occur a discontinuity. With the fractional-lambda method we activate every cubature point of one element which is in contact with the rigid object and endow them with the same penalty parameter value λ. We proceed by an $\mathbf{L}^2$ projection over the elements and quadrature points, calculate the corresponding fractional parameter as a fraction between the area of the element and the area of the common region shared but the element and the solid object. This fraction is always sub-unitary and multiplies the default value of the penalty parameter. We do not care about the shape of the object and we focus to obtain the intersection points. In general they are solution of a simple system of two equations: one is the edge of the element, the second is the boundary of the solid. In this particular case, we have a circle, easy to take into consideration, but for arbitrary cases of non elementary shapes, we have to make use of approximation techniques to get the intersection points. We remain to the simple case, but we claim that this technique can be successfully applied also for more general cases. In this particular configuration, we rapidly obtain the coordinates of the intersection and we prefer to calculate the area determined by the polygon whose nodes are the points of intersection and the vertices of the element. Often it comes to a triangle, but also other polygons with more nodes can result, up to an octagon. Thus, to keep the calculation to a minimum of simplicity, we prefer to calculate the area of the polygon without taking into account possible round parts, because they are very small quantities and can be neglected. For example, in figure 5.5, we will calculate the area of the triangle determined by the points $\Delta P1P2P3$, without calculating the area of the arch $\widehat{P1P3}$. The penalty parameter is set for the whole element as:

$$\lambda_{fr} = \frac{1}{\varepsilon} \cdot \frac{A_{pol}}{A_{el}} \tag{5.21}$$

where A_{el} is the area of the element and A_{pol} is the area of the polygon. It will be

always a value in the interval $[0, 1]$, with 0 meaning that the element has no contact with the solid and 1 meaning that the element is completely included in the solid.



$full - lambda$ $fractional - lambda$

Figure 5.7: Handling of the cubature points of one element with *PM*

In the tables 5.14 and 5.15 are presented the point values for the fractional-lambda method coupled with *G3x3*. The results are not only not improved, but they are even not coherent, in fact being a little oscillating. The conclusion is that the full-lambda approach does not affect by its discontinuity the final result, moreover it gives better results regarding the point values. However, for higher λ-penalty parameters and finer meshes, the fractional technique provides acceptable values, hence it can be used for fast results and with the reason of no discontinuity implied.

With the adaptive-lambda *PM* we took care of a better capturing of the interfaces of the penalized region. We have realized that the discontinuity issue in the integral calculations does not affect the final accuracy of the solution, but is still a subject to be taken into consideration. With this new approach two purposes were aimed: firstly the discontinuity was reduced to much smaller local virtual elements inside the same level of refinement of the mesh and secondly, the boundaries were better approximated. There are basically two aspects: the method is for most of the finite elements identical with the full-lambda method coupled with *G3x3*, while for elements like in figure 5.8 the adaptive quadrature formula is imposed. In such elements, we perform locally a refinement of the element of order 2^{2n}, where n is the level of

	P_1	P_2	P_3	P_4	$P_{5_{u_x}}$	$P_{5_{u_y}}$	$P_{6_{u_x}}$	$P_{6_{u_y}}$
	$\times 10^{-1}$	$\times 10^{-2}$	$\times 10^{-1}$	$\times 10^{-2}$	$\times 10^{-2}$	$\times 10^{-3}$	$\times 10^{-1}$	$\times 10^{-4}$
ref.	1.32220	1.47560	1.21158	1.38843	5.05319	2.01330	2.01724	8.42345
BENCH2M55 Level 9								
10^3	1.26427	1.45765	1.18417	1.41228	5.31752	1.98118	2.02398	8.34013
10^6	1.32031	1.54745	1.21729	1.44467	5.21607	1.96257	2.02297	8.37805
BENCH2M55 Level 10								
10^3	1.27869	1.45077	1.18662	1.39452	5.29471	2.01667	2.02433	8.35107
10^6	1.32527	1.52775	1.21377	1.41975	5.16990	2.01567	2.02139	8.45548
BENCH1_FBM Level 6								
10^3	1.28975	1.43930	1.19245	1.37449	5.18905	2.00175	2.02104	8.33256
10^6	1.31772	1.45115	1.21704	1.38306	4.96418	2.02721	2.01392	8.50174
BENCH1_FBM Level 7								
10^3	1.29351	1.44542	1.19237	1.37805	5.20901	1.99900	2.02179	8.32005
10^6	1.31945	1.46401	1.21401	1.38622	5.01580	2.01519	2.01585	8.45053

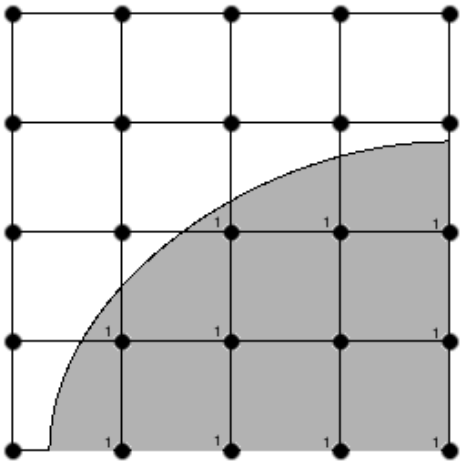
Table 5.14: Point values for fractional-lambda *PM*

refinement, and construct local penalty matrices for the new elements by using a trapezoidal rule *TRZ* and at the end they are summed up into the local mass matrix of the entire element. In the figure 5.8 it is represented how the adaptive-lambda *PM* works for one element which has a common intersection with the solid. The example is given for level 3 of local refinement, with 16 new virtual elements. The cubature points are located into the vertices of the new created elements. The same three possibilities regarding the position of the elements with respect to the solid object are to be treated: completely inside, completely outside and with common intersection part. The discontinuity does not vanish completely, but since the new elements are much smaller in volume than the parent element, the influence is also reduced. On the other hand, in the region of the solid object more quadrature points are chosen, hence a probable better mass matrix approximation is obtained. The reason of using a quadrature rule with lower order is to keep the calculations as simple as possible and because of the local refinement, the selected formula acts as a summarized trapezoidal

	P_1	P_2	P_3	P_4	$P_{5_{u_x}}$	$P_{5_{u_y}}$	$P_{6_{u_x}}$	$P_{6_{u_y}}$
BENCH2M55 Level 9								
10^3	2.45	2.46	1.58	1.00	2.69	0.57	0.19	1.08
10^6	0.34	1.66	0.45	0.39	1.76	0.69	0.16	0.93
BENCH2M55 Level 10								
10^3	2.17	1.05	1.59	0.75	3.08	0.71	0.23	1.23
10^6	0.21	0.79	0.20	0.16	0.74	0.09	0.07	0.32
BENCH1_FBM Level 6								
10^3	4.38	1.22	2.26	1.72	5.23	1.60	0.33	0.99
10^6	0.14	4.87	0.47	4.05	3.22	2.52	0.28	0.54
BENCH1_FBM Level 7								
10^3	3.29	1.68	2.06	0.44	4.78	0.17	0.35	0.86
10^6	0.23	3.53	0.18	2.26	2.31	0.12	0.21	0.38

Table 5.15: Relative errors of fractional-lambda *PM*

rule and meets the necessary order for mass matrix assembling.



adaptive-lambda

Figure 5.8: Handling of the cubature points of one element with adaptive *PM*

Regarding the solution provided with the adaptive-lambda *PM* we present the corresponding point values for TRZ_5 on the same meshes and same levels like for the

other two methods.

	P_1	P_2	P_3	P_4	$P_{5_{u_x}}$	$P_{5_{u_y}}$	$P_{6_{u_x}}$	$P_{6_{u_y}}$
	$\times 10^{-1}$	$\times 10^{-2}$	$\times 10^{-1}$	$\times 10^{-2}$	$\times 10^{-2}$	$\times 10^{-3}$	$\times 10^{-1}$	$\times 10^{-4}$
ref.	1.32220	1.47560	1.21158	1.38843	5.05319	2.01330	2.01724	8.42345
BENCH2M55 Level 9								
10^3	1.28415	1.43272	1.17081	1.37756	5.22567	1.94053	2.02102	8.38175
10^6	1.36604	1.51860	1.22998	1.42977	4.96592	2.06779	2.01362	8.78079
BENCH2M55 Level 10								
10^3	1.29281	1.44299	1.18255	1.37950	5.22725	1.99797	2.02189	8.42381
10^6	1.32511	1.47060	1.21630	1.40329	5.02312	2.01512	2.01591	8.63307
BENCH1_FBM Level 6								
10^3	1.29219	1.44016	1.19133	1.37563	5.20891	2.00301	2.02178	8.32103
10^6	1.32022	1.45324	1.21362	1.38512	5.01783	2.01462	2.01585	8.44264
BENCH1_FBM Level 7								
10^3	1.29489	1.44557	1.19210	1.37831	5.21330	1.99886	2.02195	8.31689
10^6	1.32034	1.46588	1.21183	1.38761	5.04924	2.01060	2.01707	8.42011

Table 5.16: Point values for adaptive-lambda *PM*

We observe a slightly improvement of the point values, especially in the case of the cartezian mesh, comparing with the standard full-lambda method. For the body-fitted mesh, the results are very similar and the gain is almost not important.

In conclusion, we proposed 3 main implementations of *PM*: full-, fractional- and adaptive-lambda. Each methods has its own advantages and disadvantages, treating the calculation of the mass matrix integrals and the capture of the solid interface. From the point of view of solution accuracy, the adaptive-lambda method provides better results due to the reduced influence of the characteristic function discontinuity and a closer approximation of the boundary of the solid object. However, the CPU-times increased as we shall see. The full-lambda method provides very close results to the adaptive-lambda and being easier to implement and not requiring extra handling of special finite elements, we consider it to be the main technique and we will refer always to it. The fractional-lambda intends to exclude the discontinuity introduced

	P_1	P_2	P_3	P_4	$P_{5_{u_x}}$	$P_{5_{u_y}}$	$P_{6_{u_x}}$	$P_{6_{u_y}}$
BENCH2M55 Level 9								
10^3	2.88	2.91	3.37	0.78	3.41	3.61	0.19	0.50
10^6	3.32	2.91	1.52	2.98	1.73	2.71	0.18	4.24
BENCH2M55 Level 10								
10^3	2.22	2.21	2.40	0.64	3.44	0.76	0.23	0.00
10^6	0.22	0.34	0.39	1.07	0.60	0.09	0.07	2.49
BENCH1_FBM Level 6								
10^3	2.27	2.40	1.67	0.92	3.08	0.51	0.23	1.22
10^6	0.15	1.52	0.17	0.24	0.70	0.07	0.07	0.23
BENCH1_FBM Level 7								
10^3	2.07	2.04	1.61	0.73	3.17	0.72	0.23	1.27
10^6	0.14	0.66	0.22	0.06	0.08	0.13	0.01	0.04

Table 5.17: Relative errors of adaptive-lambda *PM*

by the penalty characteristic function, considering that it may produce oscillations of velocity. However, the results did not improve, but were still acceptable in comparison with the other results provided by the main technique. The disadvantages of the methods are in the necessity of finding intersection points and approximate the common area between the solid object and the finite element.

5.3.5 Efficiency, robustness, solver aspects and solution

Two important properties of numerical solvers are the efficiency and robustness. In this sub-chapter, we present statistical data about the solvers to analyze the efficiency, robustness and to visualize the solutions of the implemented *PM* methods. We have performed all simulations on the same computer with the following characteristics: 8 processors with 4 CPU cores, 3.33GHz and 63 GiB of memory. As nonlinear solver we use a preconditioning by defect correction loop, using Multigrid solver to solve the linearized system of equations, together with BiCGStab with full VANKA for the smoothing process. The stopping criteria for the linear solvers were set as: 10^{-2} for the relative and 10^{-5} for the absolute error of the residuals on both fine and coarse

grids, while for the nonlinear solver the stopping critterions were set as: 10^{-5} for U-defect, divergence, U (velocity) and P (pressure) changes. We used for the nonlinear term a Streamline Diffusion stabilization as default one. Also, in the normal cases the smoother performs only few pre- or post smoothing steps with the relaxation parameter equal to 1. For some settings, depending on the penalty parameter, more steps were required in order to obtain a convergent solution. The behavior of the solvers in using the different implementation of the *PM* is indicated in the following tables.

λ	NIT (it)	LIT (it)	NT (s)	LT (s)	Mem (GB)
BENCH2M55 Level 9					
10^{-3}	12	26	118	108	0.467
10^{-6}	12	24	2439	2429	0.467
BENCH2M55 Level 10					
10^{-3}	12	25	476	438	1.82
10^{-6}	12	24	11705	11664	1.82
BENCH1_FBM Level 6					
10^{-3}	12	25	130	120	0.506
10^{-6}	12	30	147	137	0.506
BENCH1_FBM Level 7					
10^{-3}	12	25	598	553	1.98
10^{-6}	12	29	591	551	1.98

Table 5.18: Efficiency for full-lambda *PM*

Comparing the results of full-lambda on different meshes, the linear solver converges to a solution without any extra smoothing steps for the body-fitted mesh, but performs more iterations to reach the imposed stopping critterions. On the other hand, for higher values of the penalty parameter and for the cartesian mesh, the linear solver encountered divergence situations and only by increasing the smoothing steps and changing the relaxation parameter the problem converged. For example, for $\lambda = 10^6$ we had to set higher numbers of smoothing steps with a modified value of the relaxation parameter. The result was a minimum number of linear iterations, but

a much higher computational time because of to the smoothing process.

λ	NIT (it)	LIT (it)	NT (s)	LT (s)	Mem (GB)
BENCH2M55 Level 9					
10^{-3}	12	29	138	128	0.467
10^{-6}	12	32	1675	1665	0.467
BENCH2M55 Level 10					
10^{-3}	12	26	558	516	1.82
10^{-6}	12	25	6001	5960	1.82
BENCH1_FBM Level 6					
10^{-3}	12	25	126	116	0.506
10^{-6}	12	24	2979	2968	0.506
BENCH1_FBM Level 7					
10^{-3}	12	25	518	478	1.98
10^{-6}	12	24	10852	10314	1.98

Table 5.19: Efficiency for fractional-lambda *PM*

The fractional-lambda method gives shorter computational time in comparison with the full-lambda if the same settings for the smoother are used. In the case of $\lambda = 10^3$ the times are shorter for the same number of linear iterations for the body-fitted mesh and a little more for the cartezian mesh. It was again necessary to make use of more smoothing steps for higher penalty parameter, but the solvers were converging faster to the solution.

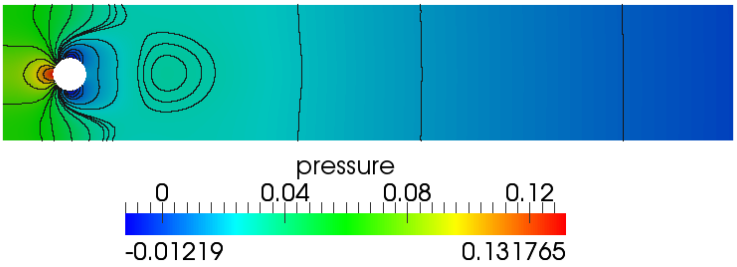
The adaptive-lambda *PM* converges to a solution for any penalty parameter 10^3, 10^6 without changing the smoother settings. However, for higher values of λ the linear solvers will need more iterations to reach the required stopping critterions.

We show in the next part some snapshots with contour lines of the solution in terms of pressure and velocity for all investigated methods. The reference ones are those obtained by the *cc2d* code. Analyzing the solutions provided by all *PM* approaches, we conclude that they are similar to the reference one. However, a closer analysis is required before we can claim other characteristics.

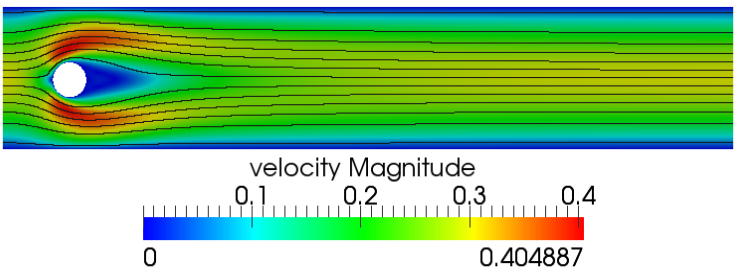
Following the idea of a closer analysis of the *PM*, we have realized cutlines of pressure

λ	NIT (it)	LIT (it)	NT (s)	LT (s)	Mem (GB)
BENCH2M55 Level 9					
10^{-3}	12	27	124	115	0.467
10^{-6}	12	120	525	515	0.467
BENCH2M55 Level 10					
10^{-3}	12	27	522	483	1.82
10^{-6}	12	84	1538	1500	1.82
BENCH1_FBM Level 6					
10^{-3}	12	25	126	116	0.506
10^{-6}	12	30	154	144	0.506
BENCH1_FBM Level 7					
10^{-3}	12	25	523	482	1.98
10^{-6}	12	29	677	633	1.98

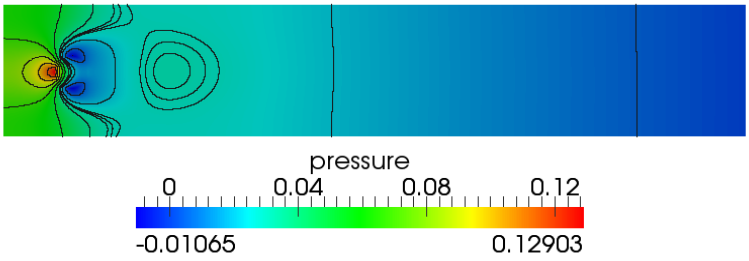
Table 5.20: Efficient for adaptive-lambda *PM*



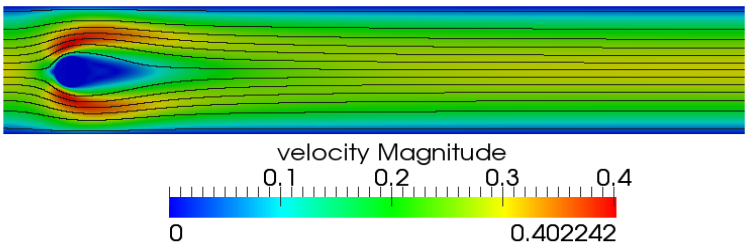
cc2d - Pressure



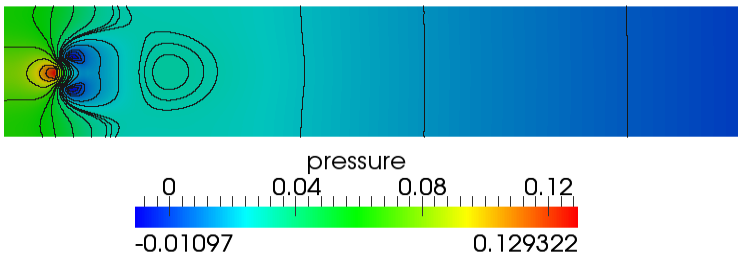
cc2d - Velocity



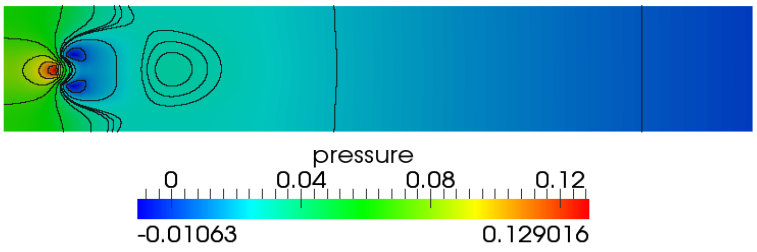
full-lambda - Pressure



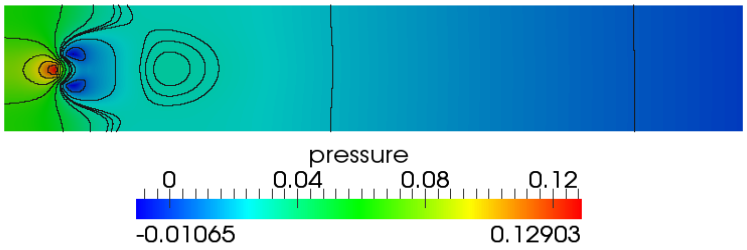
full-lambda - Velocity



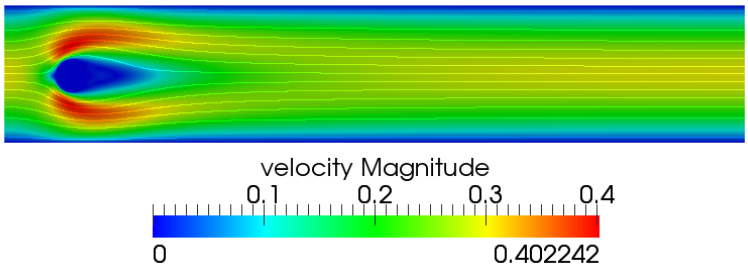
fractional-lambda - Pressure



fractional-lambda - Velocity



adaptive-lambda - Pressure



adaptive-lambda - Velocity

and velocity out of the presented snapshots and plot them. We present the plots into the following figures, but before some details are necessary. The cutlines were special selected for the horizontal line $y = 0.2$ and for the vertical line $x = 0.2$ since they pass the solid area along its diameter. Also, we only present here plotting figures for the main implementation of *PM*, full-lambda technique. Since we have realized that the use of a body-fitted mesh leads to better solutions with respect to the reference one, we restrict in here to present the analyze of several levels of refinement of $BENCH\_FBM$ and for different penalty parameters.

Firstly we debate the comparison of different penalty parameter values and same level of refinement for the grid. Clear differences between full-lambda *PM* with different values of ε-penalty parameter can be observed only along x-cutline. For y-cutline, the values are very similar and different only after 5-6 digits. From the x-cutline we observed that for smaller penalty parameter, the maximums for pressure and velocity are higher, while by increasing the penalty parameter these values decrease. They will tend to be the same no matter how much bigger the penalty constant will be set. This fact lead us to the idea of searching an optimal penalty parameter, since

the size of it will determine ill-conditioned system matrix and will not improve the solution. Moreover, it will arise numerical problems for the linear solver which will lead to divergence if no smoothing or stabilization techniques are applied. In the interior part of the rigid object, the velocity of the fluid should tend to be zero, but if λ is not sufficient small, then the penalized area acts as an porous media and small velocities can be detected around the inner part of the boundary. If the penalization is stronger, then the velocity tends to 0 in the entire region of solid. The y-cutlines for the same grid level do not show discrepancies between the different parameters.
The behavior of the *PM* for different levels of refinements is also presented for these two cutlines. However, we just plot them for $\lambda = 10^3$. We observe again the gain of a better interface approximation if the grid is finer. This is outputted trough the fact that around the object the velocity vector tends to 0. This can be seen for both types of cutlines. Regarding the pressure, in the case of the x-cutline, the maximum inside the solid area grows around the middle of the object, but suffers a decay on the y-cutline.
For the purpose of drag and lift coefficient calculations, we present also plots around the boundary of the solid object, in terms of arc length. Also in this case, the velocity field should tend to 0. In the case of $\varepsilon = 10^3$ it does not really happen, but already $\varepsilon = 10^6$ will force the velocity to be smaller and almost zero. However, we notice the small oscillations of the velocity around the upper (N) and lower (S) positions with respect to the interface of the solid. These oscillations are propagated also for the pressure, which will strongly oscillate as the λ-penalty parameter increases. As we shall see, the drag and lift coefficients are better in the case of smaller ε-parameters, although the solution is not accurate. This is because of the pressure which has the most percentage in the calculation of these two quantities. Hence, post-processing smoothing techniques for the pressure should be considered in order to improve the drag and lift for higher values of the penalty parameter.

5.3.6 Drag and lift coefficients

For the purpose of possible applications of *FBM* and *PM* methods, like particulate flows for example, we have computed the values of the well know Drag and Lift coefficients, calculated around the surface of the solid object [45]:

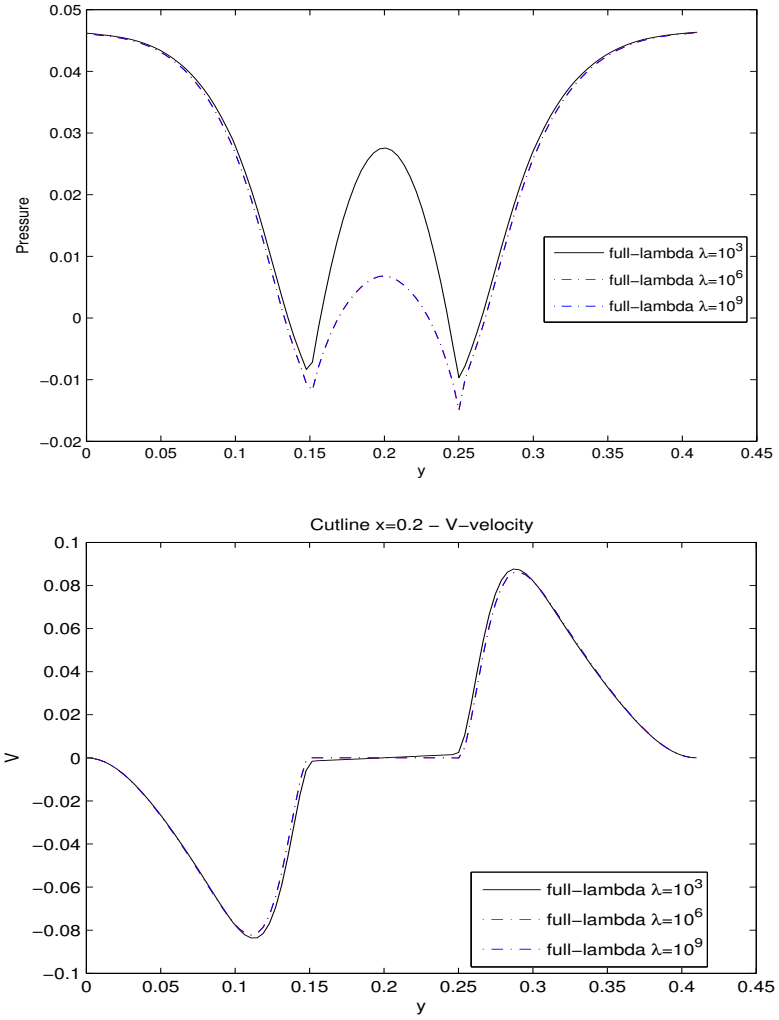


Figure 5.9: Cutline of pressure and velocity at $x = 0.2$

$$C_{d/l} = \frac{2f_{d/l}}{\rho \overline{U}^2 D} F_{d/l} \tag{5.22}$$

U and L are the characteristic velocity and length. The coefficients $f_{d/l}$ are set to 1 and $F_{d/l}$ are the drag and lift forces acting on the object. There are two approaches to calculate these forces. Since they are defined as surface integrals (5.23), one may use a line or a volume integration. More about this subject can be read in [24]. The line integration is not recommended to use in the case of *PM*. Instead, the volume

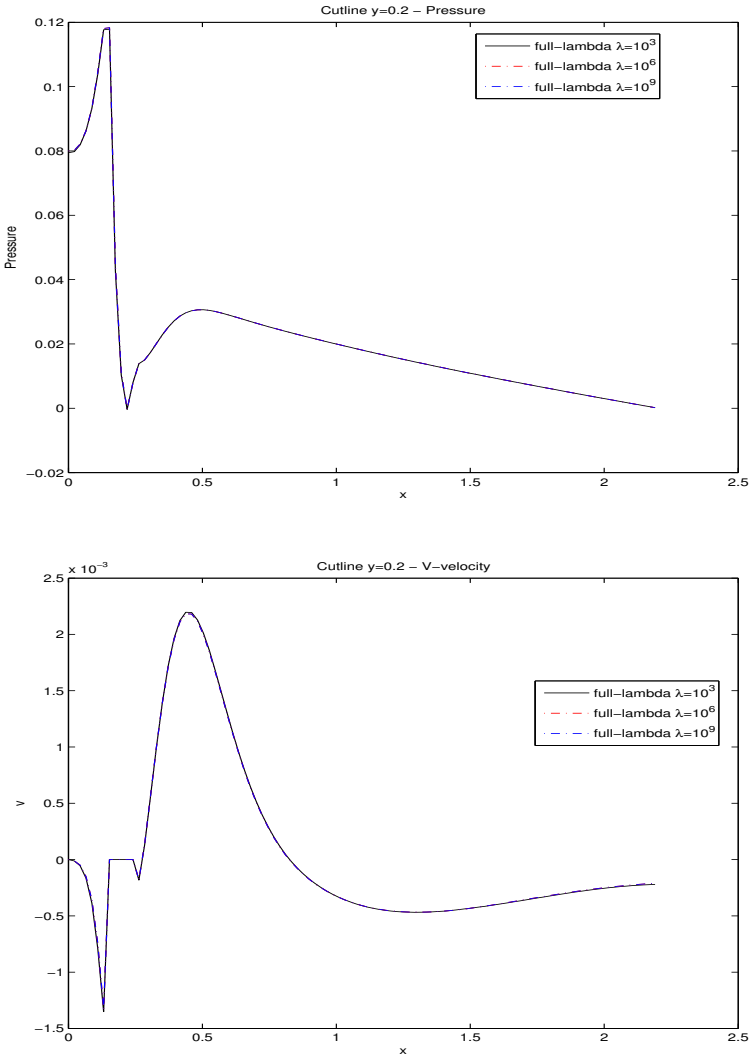


Figure 5.10: Cutline of pressure and velocity at $y = 0.2$

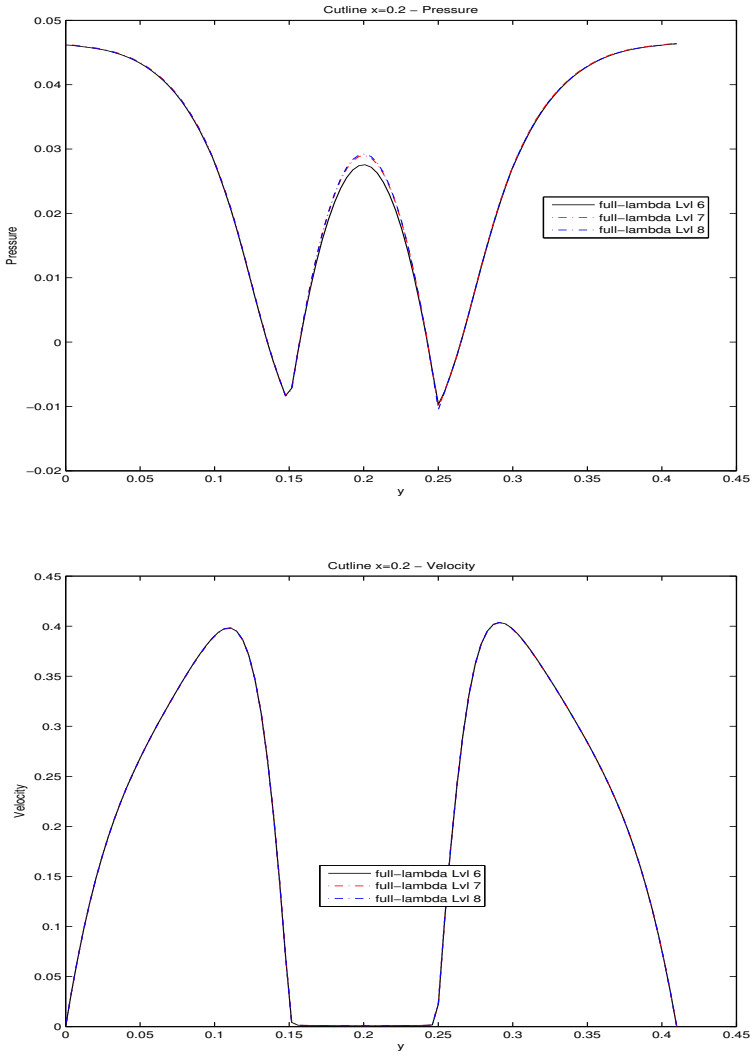


Figure 5.11: Cutline of pressure and velocity at $x = 0.2$

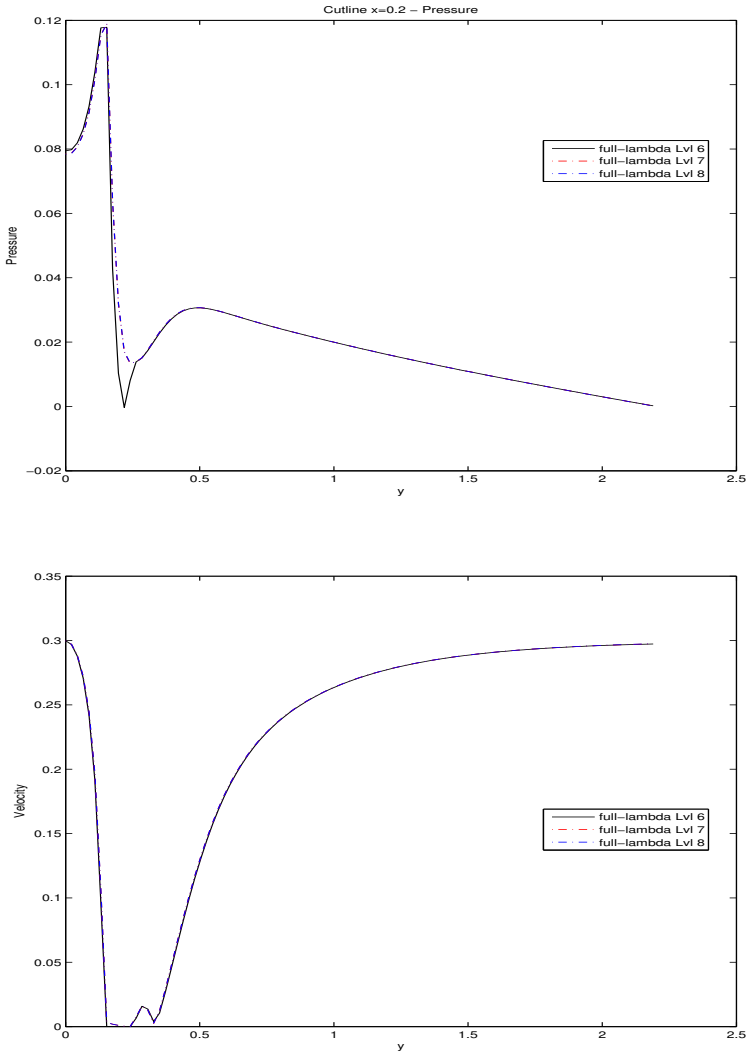


Figure 5.12: Cutline of pressure and velocity at $y = 0.2$

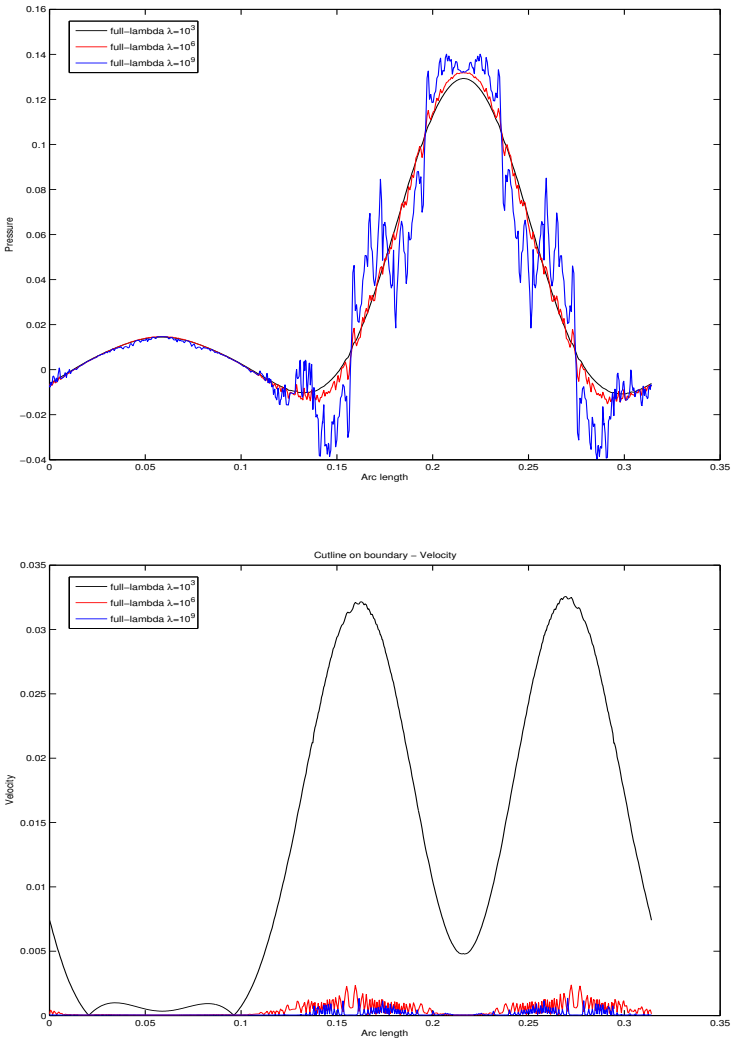


Figure 5.13: Cutline of pressure and velocity around the interface

integration is an option to be considered. However, line integration is also possible with the specification that the connection points obtained as intersections between the finite elements and boundary of the object are necessary and will construct the corresponding line. We give further a short description of the volume integration of drag C_d and lift C_l coefficients acting on the solid object. Let S be the surface of the object, $\mathbf{n}_S$ the unit normal vector oriented inward with respect to Ω and the tangential vector with the components $\tau = (n_u, -n_x)$. Drag and lift forces are defined by the following surface integrals:

$$\begin{cases} F_d = \int_S \left(\mu \frac{\partial \mathbf{u}_\tau}{\partial \mathbf{n}_S} n_y - p\, n_x\right) ds, \\ F_l = -\int_S \left(\mu \frac{\partial \mathbf{u}_\tau}{\partial \mathbf{n}_S} n_x + p\, n_y\right) ds. \end{cases} \tag{5.23}$$

The corresponding drag and lift coefficients are then obtained by using (5.22). For the calculation of the integrals we do not reconstruct the shapes of the boundary and we define a parameter α as:

$$\alpha(\mathbf{x}) = \begin{cases} 1, \mathbf{x} \in \Omega_c, \\ 0, \mathbf{x} \in \Omega \setminus \Omega_c. \end{cases} \tag{5.24}$$

which selects the midpoints of the edge of cells which are inside the volume occupied by the solid object Ω_c. $\Omega \setminus \Omega$ represents the fluid domain. The new parameter has the property that the gradient is 0 everywhere except at the wall surface of the solid and given as described in [9] by the relation:

$$\mathbf{n} = -\nabla \alpha \tag{5.25}$$

Hence the drag and lift forces acting on the solid can be calculated by using the total stress tensor σ as follows:

$$F = \int_\Gamma \sigma \mathbf{n} d\Omega = -\int_{\Omega_T} \sigma \nabla \alpha d\Omega \tag{5.26}$$

where Ω_T is the entire computational domain. Straightforward are the relations:

$$\begin{cases} F_d = -\int_{\Omega_T} \left(\mu \frac{\partial u}{\partial x}\frac{\partial \alpha}{\partial x} + \frac{\partial u}{\partial y}\frac{\partial \alpha}{\partial y} - p\frac{\partial \alpha}{\partial x}\right) d\Omega, \\ F_l = -\int_{\Omega_T} \left(\mu \frac{\partial v}{\partial x}\frac{\partial \alpha}{\partial x} + \frac{\partial v}{\partial y}\frac{\partial \alpha}{\partial y} - p\frac{\partial \alpha}{\partial y}\right) d\Omega. \end{cases} \tag{5.27}$$

Therefore, a volume integral over the whole domain can be applied to calculate the drag and lift forces, a more convenient way for the *FBM* and *PM*. The table below presents results for drag and lift coefficients obtained from simulating flow around cylinder by using the grid *BENCH1_FBM* and for $Re = 20$. The reference values are according to [9] $C_d = 0.55795$ and $C_l = 0.010618$.

	C_d		C_l	
Level	FBM	full-λ	FBM	full-λ
6	5.5808	5.4191	0.0947	0.0702
7	5.5799	5.4352	0.1019	0.0826
8	5.5799	5.4506	0.1032	0.0830

Table 5.21: Drag and Lift coefficients for *FBM* and full-lambda *PM* 10^{-3}

	C_d		C_l	
Level	FBM	frac-λ	FBM	frac-λ
6	5.5808	5.2753	0.0947	0.0727
7	5.5799	5.3653	0.1019	0.0803
8	5.5799	5.4187	0.1032	0.0824

Table 5.22: Drag and Lift coefficients for *FBM* and fractional-lambda *PM* 10^{-3}

	C_d		C_l	
Level	FBM	adapt-λ	FBM	adapt-λ
6	5.5808	5.4352	0.0947	0.0826
7	5.5799	5.4506	0.1019	0.0830
8	5.5799	5.4801	0.1032	0.0905

Table 5.23: Drag and Lift coefficients for *FBM* and adaptive-lambda *PM* 10^{-3}

All the values from table (5.21, 5.22, 5.23) are multiplied by a factor of 10^{-1}. We have tabulated results for only $\varepsilon = 10^{-3}$. For higher values of ε, the drag and lift coefficients calculated by the proposed volume integration technique are becoming worse due to the fact that the pressure, the main constitutive part of the coefficients,

is strongly oscillating (5.13) if the penalty parameter becomes stronger. To emphasis this situation, we show in the the following table (5.24) the results of drag and lift coefficients for the main method full-lambda *PM* with the penalty parameter 10^{-6}, specifying that the same phenomenon happens also for the other two *PM* methods.

	C_d		C_l	
Level	FBM	adapt-λ	FBM	adapt-λ
6	5.5808	4.2365	0.0947	0.0688
7	5.5799	4.4931	0.1019	0.0733
8	5.5799	4.8587	0.1032	0.0871

Table 5.24: Drag and Lift coefficients for *FBM* and adaptive-lambda $PM10^{-3}$

All the above results were obtained with the volume integration method and using the *BENCH1_FBM* body-alligned mesh. We notice that the best results are obtained for the adaptive-lambda *PM*, but still not satisfactory since the relative error is bigger about 1%, while all other cases the relative error is bigger. The increasing of the level of the mesh improves the values for drag and lift coefficients such that, a more accurate alligned mesh around the interface of the solid object will provide even closer results to the reference one. We also claim that there is an optimal value for the ε-penalty parameter for which the calculation are the most good, but we can not figure a logical relation to determine this value. However, with smoothing techniques for the velocity and pressure solution around the solid object, it is espected that the results are getting closer to the reference. The calculations of drag and lift coefficients in a more proper way for the penalty methods and more accurate has to be a future task for the plans of applying penalty method in more complex configurations, of fluids with moving objects, where the movement is governed by the physical laws.

Chapter 6

Applications

6.1 Oscillating cylinder benchmark

To demonstrate the ability of the presented *PM* to handle flows with complex moving boundaries, we have simulated one oscillating cylinder in a channel problem. The movement of the solid object is prescribed by a periodical function, depending on an amplitude, a frequency and time. The computational domain is the channel 2.2×0.41, triangulated by the mesh from figure 6.1 and with the statistical data from table 6.1.

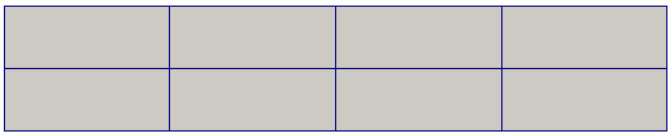


Figure 6.1: Mesh channel LEVEL = 1

LEVEL	NVT	NMT	NEL	NEQ
2	45	76	32	184
3	153	280	128	688
4	561	1072	512	2656
5	2145	4192	2048	10432
6	8385	16576	8192	41344

Table 6.1: Statistical data for the cartezian grid channel mesh

The movement of the solid particle is described by the x-translational movement of the center point:

$$\begin{cases} x_c = x_0 + Asin(2\pi ft), \\ y_c = y_0, \end{cases} \tag{6.1}$$

with the initial position $x_0, y_0 = 1.1, 0.2$ in the center of the channel. Here, $A = 0.25$ is the amplitude, $f = 0.25$ is the frequency and t is the time parameter. The density of the solid particle is set to $\rho = 1$ and the kinematic viscosity of the fluid to $\nu = \frac{\mu}{\rho} = 10^{-3}$. The cylinder has the diameter $D = 0.1$ and the fluid is initially at rest. No-slip boundary conditions are set for the entire computational domain. A non-stationary model of the incompressible Navier-Stokes problem implemented with the full-lambda *PM* is solved. We have only chosen the full-lambda candidate since from the validation point of view it provides good accuracy and efficiency with minimum implementation effort. However, both fractional- and adaptive-lambda *PM*s can be used to simulate and solve this problem. For the time discretization we only applied a *BE* method, but any fractional-θ scheme presented in the previous chapters can be used.

The flow is initially at rest while the solid particle has already its own velocity. It perturbs the flow by starting to translate, such that the flow becomes very complex when the solid object is in the positions at $t = t_0 + \frac{2n}{4}T$, $n \in \{0, 1, 2, \dots\}$. The figures below show some vorticity contour plots in different time point of the periodical move: $t = t_0, t = t_0 + \frac{1}{4}, t = t_0 + \frac{2}{4}, t = t_0 + \frac{3}{4}$. As said before, the critical points are the moment of changing the direction of the movement of the solid object, when it has zero velocity and starts afterwards to translate in opposite direction. These snapshots show that the fluid is perturbed by the oscillating cylinder and vortex is generated periodically in the wake of the cylinder. The last snapshots show also the velocity vector field at three essential moments of the movement.

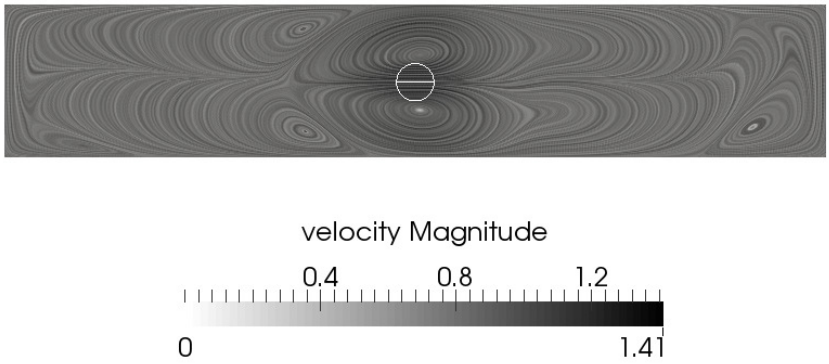


Figure 6.2: $t = t_0$

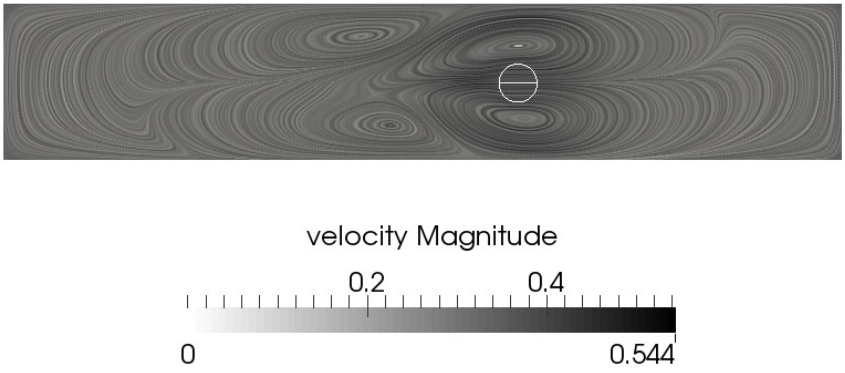


Figure 6.3: $t = t_0 + \frac{1}{4}T$

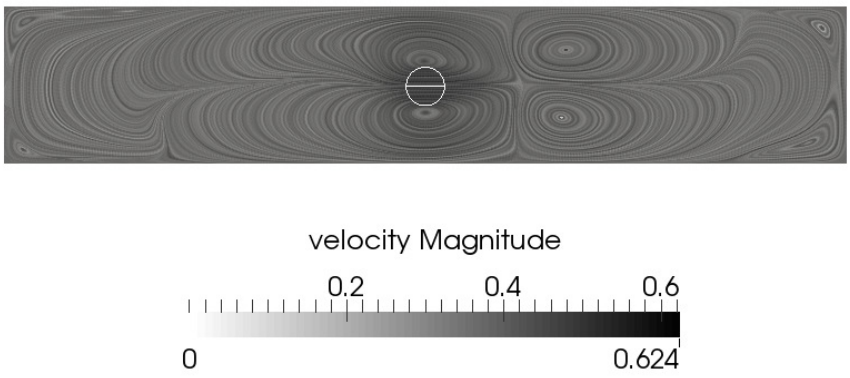


Figure 6.4: $t = t_0 + \frac{2}{4}T$

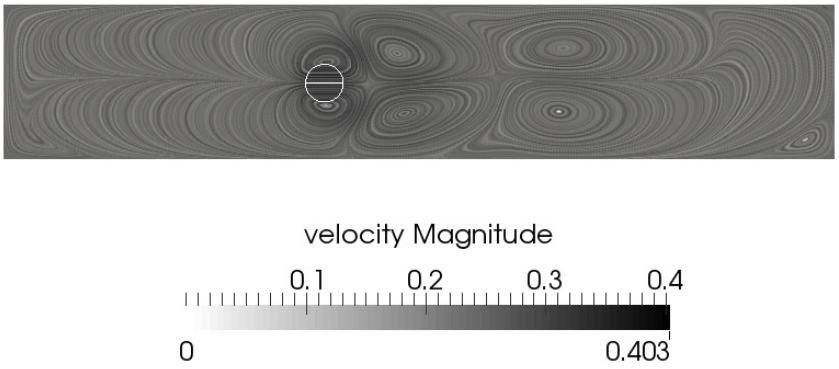


Figure 6.5: $t = t_0 + \frac{3}{4}T$

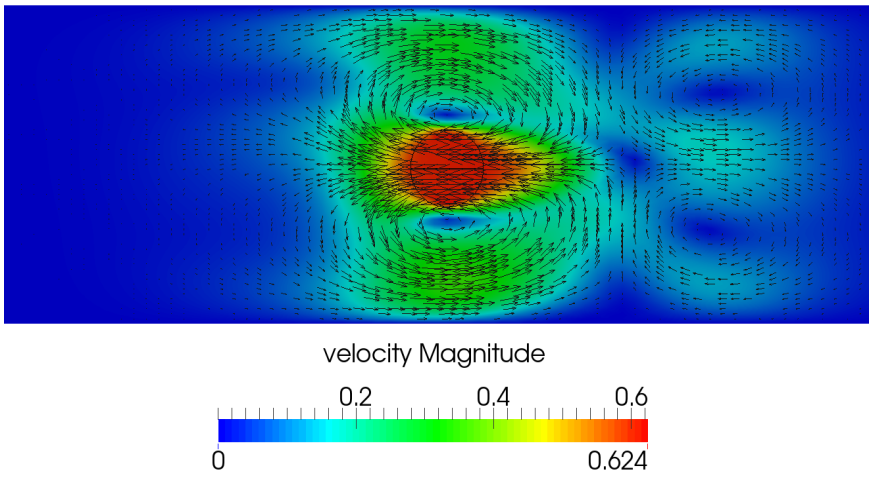


Figure 6.6: $t = t_0 + \frac{2}{4}T$

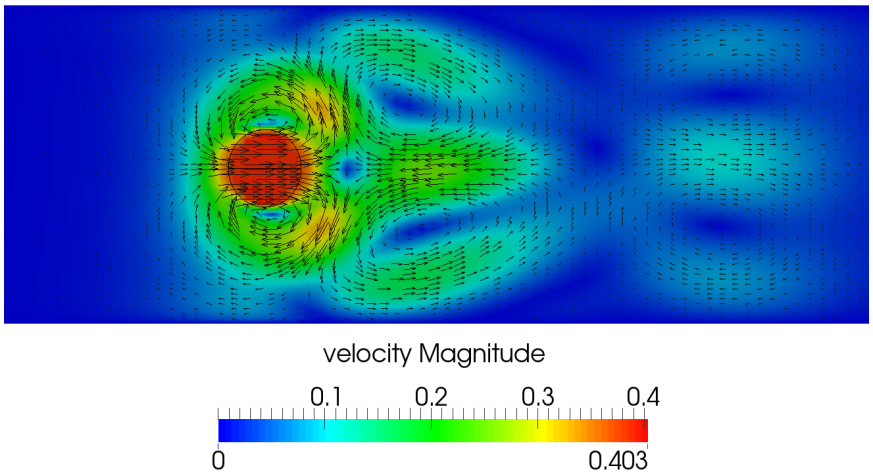


Figure 6.7: $t = t_0 + \frac{3}{4}T$

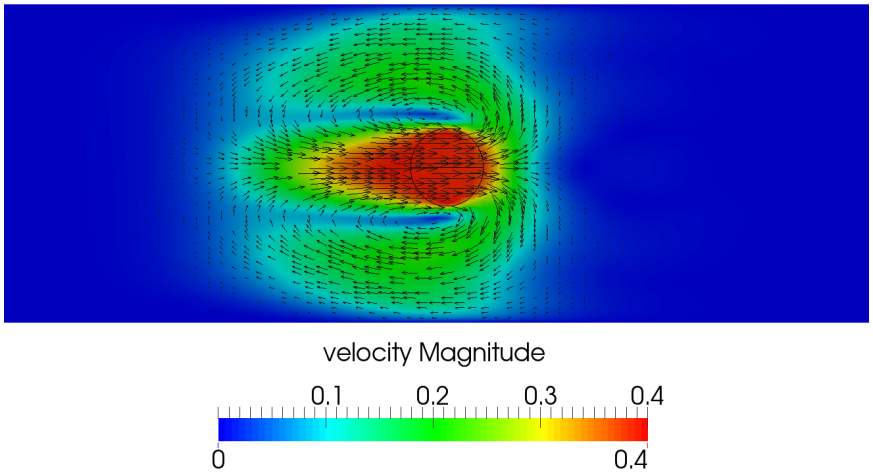


Figure 6.8: $t = t_0 + T$

6.2 Rigid object with complex shape geometry

This subsection intend to show that the proposed *PM* can deal successfully complex geometries of the rigid body, not only the standard cases of circles/particles, ellipses,

squares/rectangles, for which the distant function is easy to compute. For this purpose, we simulate flow around a stationary complex object to prove the ability of the method of capturing any kind of boundary shape. We have choose a stationary trivial 2D fusiform representation, which can be interpreted as a simplified fish-contour. The length of the shape is chosen to be $L \in \{0.5, 0.75, 1.0\}$, the head is fixed in the point $(x_0, y_0) = (0.2, 0.2)$ and the maximum thickness is at point $x = 0.25$ and set to 0.1. For simplicity, we imposed that mid-line of the shape is parallel to the velocity profile set at the inflow. The described shape was obtained by using cubic spline interpolation for the points $(x_i, y_i) \in \{(0.2, 0.2), (0.25, 0.25), (0.2 + L, 0.2)\},\ i = \overline{1,3}$. Solving the system of equations obtained by imposing the conditions of the cubic spline interpolation, we can define four functions which together describe a continuous and $\mathcal{C}^2$ closed curve. For the case of $L = 0.5$ the fish-contour reads:

$$\begin{cases} l_1(x) = 2y_0 - \left(\dfrac{1}{4} - \dfrac{20}{81}(x - x_1)^2 + \dfrac{3200}{81}(x - x1)^3\right),\ x \in [x_0, x_1] \\ l_2(x) = 2y_0 - \left(\dfrac{1}{4} - \dfrac{20}{81}(x - x_1)^2\right),\ x \in [x_1, x_0 + L] \\ s_2(x) = \dfrac{1}{4} - \dfrac{20}{81}(x - x_1)^2,\ x \in [x_1, x_0 + L] \\ s_1(x) = \dfrac{1}{4} - \dfrac{20}{81}(x - x_1)^2 + \dfrac{3200}{81}(x - x1)^3,\ x \in [x_0, x_1] \end{cases} \tag{6.2}$$

For different length of the mid-line of the shape, corresponding coefficients will be obtained. For the moment we do not care that the head and tail part are not smooth, since the purpose is to simulate fluid around solid object with complex shape. More over, this are third order singularities and are parallel to the flow field, hence it does not perturb the flow. Further on, the technique of capturing the boundary of such a shape by using PM or FBM is still based on *in/out* check, but in the reference coordinate system by projection methods. We provide in the following part snapshots of the solution for 3 different length (0.5, 0.75 and 1.0) presenting the streamline contours and the fluid velocity vector field.

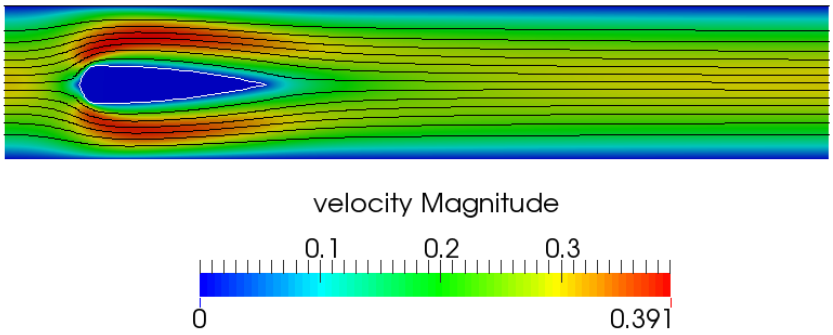


Figure 6.9: Streamline countours - flow around a fusiform shape $L = 0.50$

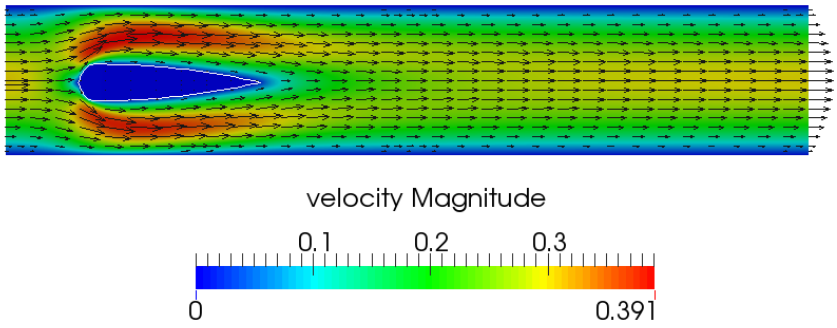


Figure 6.10: Velocity vector field - flow around a fusiform shape $L = 0.50$

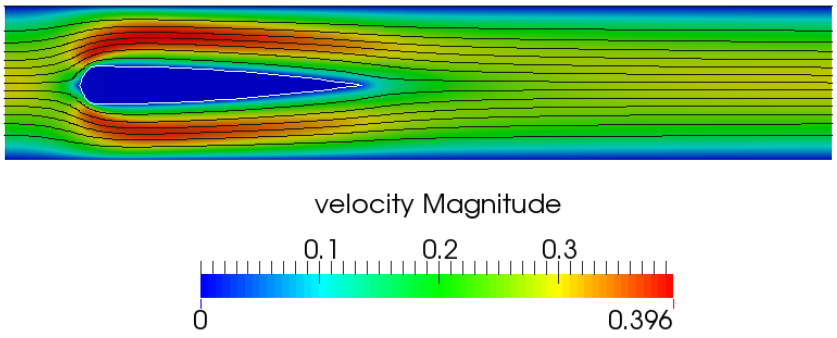


Figure 6.11: Streamline countours - flow around a fusiform shape $L = 0.75$

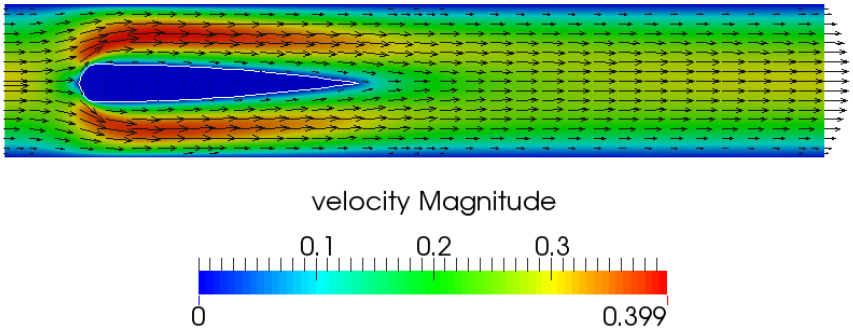


Figure 6.12: Velocity vector field - flow around a fusiform shape $L = 0.75$

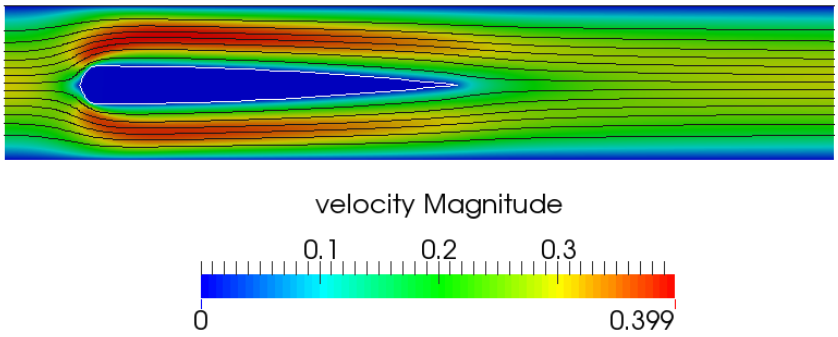


Figure 6.13: Streamline countours - flow around a fusiform shape $L = 1.0$

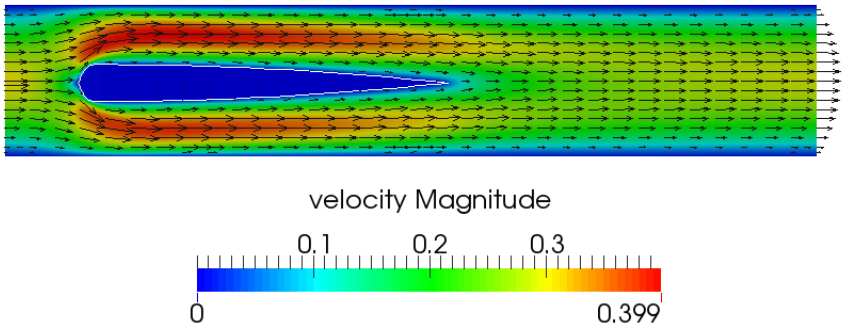


Figure 6.14: Velocity vector field - flow around a fusiform shape $L = 1.0$

For all these simulations the full-lambda PM method with the penalty parameter $\varepsilon = 10^{-3}$ was applied, using the cartezian mesh $BENCH2M55$ with level 8 of refinement. The figures above prove the capability of the presented penalty method to capture also different complex shape interfaces. We have calculated also the drag and lift coefficients that act on such an object and the values are presented in the table below:

Length	C_d	C_l
0.50	9.82189	1.16842
0.75	11.0277	1.67524
1.00	12.1265	1.74881

Table 6.2: Drag and Lift - flow around a fusiform shape full-lambda $PM10^{-3}$

The values from the table (6.2) show that only by means of penalty technique, the drag and lift coefficients modify. For this experiment we do not have any reference value, but the aim was to underline the aspect of the possibility of using arbitrary complex geometries of the solid object. In this sense, we considered just the stationary case. This opportunity opens the way of simulating moving complex shaped objects, like for instance a moving fish in a channel or even moving fishes in channel in opposite direction and meeting. This is a theme for further study: what happens with the flow when the fish perturbs it by its motion? what happens when the fishes are meeting: will they collide or swim past each other?

Chapter 7

Conclusion

This study focused on the investigation of a velocity penalization method of type Brinkman introduced for the first time by Angot, Bruneau and Fabrie [2]. The proposed *PM* methods benefits of the framework of *FEM* discretization techniques and a *MG* linear solver. The aim of the study was to develop a generalization method for the already validated *FBM* of Turek and Wan [50], with a proper mathematical support in effectively imposing a no-slip boundary condition on the velocity for solid/rigid objects inside a fluid domain. The method was applied to the Navier-Stokes model problem for incompressible flows, for laminar and transient flows (low and medium *Re* number), with solid objects and consists in adding a new term in the momentum equation which depends on a mask function $\chi(\mathbf{x})$ and a penalty parameter λ which tends to infinity (ε-penalty parameter tends to zero). Angot et al. [2] have analytically showed that the solution of the penalized Navier-Stokes equations converges to the actual Navier-Stokes solution if the penalization parameter λ is sufficiently large. Also, inside the solid object, the velocity component tends to 0, proving the capability of the method into imposing no-slip boundary condition. The immediate advantage is the possibility of introducing, in an easy way, solid boundaries without any condition over the penalty parameter or selected grid and without the need of changing the system matrix structures and entries. Moreover, the free choice of λ allows the error to be controlled precisely. The numerical realization of the methods is performed by using finite element schemes. The nonconforming $\widetilde{Q_1}/Q_0$ element was chosen for the space-discretization as part of the problem, $\widetilde{Q1}$ for velocity component u and Q_0 for

the pressure component p. The selected element satisfies unconditionally the inf-sup condition LBB without any additional stabilization, providing comparable accuracy like the first oder in the energy norm elements. Hence, the computations could run on a fixed grid, without need of re-meshing, which is in general cheaper. Cartesian meshes could be used, but for accuracy reasons, body-aligned meshes around the solid object are better candidates, without generating higher computational times or have negative effect on the efficiency of the linear and non-linear solvers. The non-linearity part of the Navier-Stokes model is treated by a fixed point defect correction iterative method, resulting linear subproblems in each non-linear step, which are finally solved by a Galerkin type direct coupled or splitting approach. The MG solver based on smoothers or preconditioners of type $VANKA$ solves the linear subproblems by recursively calculating the solution from one level of the grid to the next one, applying a restriction operator. After a V-, W- or F- cycle the obtained solution is prolongated to the final grid. For the case of non-stationary problems, a fractional-θ-scheme is used for the time discretization, obtaining in each time-step stationary problems. We have fully investigated and validated the proposed methods on the benchmark simulation flow around cylinder [45], presenting different implementation techniques of the PM with all details regarding implementation and presenting it as a generalization of the fictitious domain methods. We have also prove that the FBM is a special case of the proposed PM providing the best results. In this sense, for special configurations PM reproduces very accurately the results of the FBM, justifying the avatar of special case of the FBM for PM. The novelty in the proposed PM is the process of assembling the penalty matrix from the point of view of the quadrature formula and lumping process. Based on this aspects, we have implemented three variations of the PM, namely full-lambda, fractional-lambda and adaptive-lambda PM. The full-lambda method requires less resources to be implemented and uses a Dirac type of mask function. From our numerical investigations, it proves to be the best candidate regarding accuracy, efficiency and robustness. To eliminate the discontinuity of the mask function χ, we proposed the fractional-lambda PM. It treats all cubature points of one finite element in the same manner, setting the same penalty parameter value for all of them and taking care that the mass of the element is preserved. This supposes additional calculations to set the proper penalty parameter and the

advantage of having no discontinuity is overcome by the disadvantage of a less accurate interface capturing. The fractional-lambda does not provide better results then full-lambda, but they are still satisfactory. The adaptive-lambda *PM* deals with both aspects of interface capturing and mask function discontinuity, tending to improve both of them in the same time. From the validation process we can claim that the adaptive-lambda method is the best, but requires more implementation techniques and computational-time, while the gain in accuracy w.r.t to full-lambda method is not important. However, the discontinuity issue is reduced to very small local virtual elements and the interface of the rigid object is better approximated. Based on accuracy, efficiency, implementation and pre/postsmoothing processes, we consider full-lambda *PM* as the main implementation, because it does not require extra techniques to implement the penalty term, it is easy to incorporate into *CFD* codes and provides good results in comparison with *FBM* and other methods. We have used a *HRZ*-lumping technique, which can be applied for each *PM* methods, in order to improve the linear solver behavior. The reason of considering this lumping technique is due to the discretization element $\widetilde{Q_1}/Q_0$ which may generate non-positives entries for the off-diagonal entries of the penalty mass matrix. The *HRZ* method eliminates these kind of entries and preserves in the same time also the mass. The gain of using a penalty matrix *HRZ* lumped consists in a better efficiency of the linear solver. To settle a good base for the applications of *PM* to particulate flows, drag and lift hydrodynamical forces acting on the surface of the solid object were calculated by using a volume integration technique. However, the results for the drag and lift were not satisfactory due to the oscillation of the velocity and pressure solution around the interface of the rigid body. To improve the results, smoothing techniques especially for the pressure should be applied. Further on, to show the possibilities of the proposed method, also non-stationary flows with complex moving boundaries and flows with stationary complex shaped rigid objects were presented. The derived methods in this thesis can be successfully applied in simulating efficient and accurate flows with moving/steady complex geometries.

Bibliography

[1] D. Anca and S. Turek. A new immersed boundary and moving mesh method for particulate flow. Technical report, Fakultät für Mathematik, TU Dortmund, April 2007. Ergebnisberichte des Instituts für Angewandte Mathematik, Nummer 347.

[2] P. Angot, Bruneau C.-H., and Fabrie P. A penalization method to take into account obstacles in incompressible viscous flows. *Numer. math.*, 81:497–520, 1999.

[3] E. Arquis and J.P. Caltagirone. Sur le conditions hydrodynamiques au voisinage d'une interface milieu fluide - milieu poreux: application à la convection naturelle. 299(1), Série II, 1-4, 1984.

[4] G.K. Batchelor. *An Introduction to Fluid Mechanics*. Cambridge University Press, 1967.

[5] G.K. Batchelor. *An Introduction to Fluid Dynamics*. Cambridge University Press, 1988.

[6] C.-H Bruneau. Boundary conditions on artificial frontiers for incompressible and compressible navier-stokes equations. *ESAIM: Mathematical Modelling and Numerical Analysis*, 34:303–314, 2000.

[7] P.G. Ciarlet. *Handbook of numerical analysis*, volume II. Ciarlet, P.G. and Lions, J.L., Editors, 1991.

[8] R. Courant. Variational methods for the solution of problems of equilibrium and vibrations. *Bull. Amer. Math. Soc.*, 49:1–23, 1943.

[9] C. Duchanoy and T.R.G. Jongen. Efficient simulation of liquid-solid flows with rigid solids fraction in complex geometries. *Computers and Fluids*, 32:1453, 2003.

[10] R. Glowinski. *Handbook of numerical analysis*, volume IX. Ciarlet, P.G. and Lions, J.L., Editors, 2003.

[11] R. Glowinski, T.W. Pan, T.I. Hesla, and D.D. Joseph. A distributed lagrange multiplier/fictitious domain method for particulate flows. *Int. J. Multiphase Flow*, 25:755–794, 1999.

[12] R. Glowinski, T.W. Pan, T.I. Hesla, D.D. Joseph, and J. Perieux. A fictitious domain approach to the direct simulation of incompressible viscous flow past moving rigid bodies: Application to particulate flow. *J. Comput. Phy*, 169:363–426, 2001.

[13] R. Glowinski and J. Periaux. *Numerical Methods for nonlinear problems in fluid dynamics*. 1987.

[14] D. Goldsteain, R. Handler, and L. Sirovich. Modelling a no-slip flow boundary with an external force field. *J. Comput. Phys.*, 105(2):354–366, 1993.

[15] M. Grajewski, M. Köster, S. Kilian, and S. Turek. Numerical analysis and practical aspects of a robust and efficient grid deformation method in the finite element context. Technical report, Fakultät für Mathematik, TU Dortmund, 2005. Ergebnisberichte des Instituts für Angewandte Mathematik, Nummer 294.

[16] W. Hackbusch. Springer-Verlag, 1985. ISBN: 0-387-12761-5.

[17] P. Hansbo. An interior penalty finite element method for elasto-plasticity. 2008.

[18] John G. Heywood, R. Rannacher, and S. Turek. Artificial boundaries and flux and pressure conditions for the incompressible navier-stokes equations. *International Journal for Numerical Methods in Fluids*, 22.5:325–352, 1996.

[19] E. Hinton, T. Rock, and O.C. Zienkiewicz. A note on mass lumping and related processes in the finite element method. *Earthquake Engineering and Structural Dynamics*, 4:245–249, 1967.

[20] E. Hinton, T. Rock, and O.C. Zienkiewicz. A note on mass lumping and related processes in the finite element method. *Earthquake engineering and structural dynamics*, 4:245–249, 1976.

[21] H. H. Hu. Direct simulation of flows of solid-liquid mixtures. *Internat. J. Multiphase Flow 22*, 22:335–352, 1996.

[22] H. H. Hu, D.D Joseph, and M.J. Crochet. Direct simulation of fluid particle motions. *Theor. Comp. Fluid Dyn*, 3:285–306, 1992.

[23] S. Hysing. *Numerical simulation of immiscible fluids with FEM level set techniques.* PhD thesis, Technische Universität Dortmund, December 2007.

[24] V. John. Higher order finite element methods and multigrid solvers in a benchmark problem for the 3-d navier-stokes equations. 40:775–798, 2002.

[25] V. John. Higher order finite element methods and multigrid solvers in a benchmark problem for the 3d navier-stokes equations. *Int. J. Num. Meth. Fluids*, 134:775, 2002.

[26] L. Landau and E. Lifschitz. *Fluid Mechanics.* Addison-Wesley, 1953.

[27] L. Landau and E. Lifschitz. *Méchanique des Fluides.* Mir, 1971.

[28] J. Leray. Etude de diverses équations intégrales non linéarais et de quelques problèmes que pose l'hydrodynamique. *J. math. Pures Appl.*, 12:1–82, 1933.

[29] J. Leray. Essai sur les mouvements plans d'une liquide visquex emplissant l'esspace. *Acta Math.*, 63:193–248, 1934.

[30] J. Leray. Essai sur les mouvements plans d'une liquide visquex que limitent des parois. *J. math. Pures Appl.*, 13:331–418, 1934.

[31] M. Marion and R. Temam. *Handbook of numerical analysis*, volume VI. Ciarlet, P.G. and Lions, J.L., Editors, 1998.

[32] B. Maury and R. Glowinski. *Fluid-particle flow: a symmetric formulation*, volume Série I 324. C.R.Acad. Sci. Paris, 1997.

[33] Bertrand Maury. A fat boundary method for the poisson problem in a domain with holes. *Journal of scientific computing*, 16:319–339, 2001.

[34] C. Perskin. Numerical analysis of blood flow in the heart. *J. Comput. Phys. 25*, 1977:220–252, 1977.

[35] C. Perskin. The fluid dynamics of heart valves: Experimental, theoretical and computational method. *Annu. Rev. Fluid Mech.*, 14:235–259, 1982.

[36] W. Prager. *Introduction to Mechanics of Continua.* Ginn and Company, 1961.

[37] R. Rannacher and S. Turek. A simple nonconforming quadrilateral stokes element. *Numer. Meth. part. Diff. Equ.*, 8:97–111, 1992.

[38] E.M. Saiki. and S. Biringen. Numerical simulation of a cylinder in a uniform flow: application of a virtual boundary method. *J. Comput. Phys.*, 123:450–465, 1996.

[39] L. Schwartz. *Théorie des Distributions.* Hermann, 1966.

[40] J. Shen. On error estimates of the penalty method for unsteady navier-stokes equations. *SIAM J. Numer. Anal.*, 32:386–403, 1995.

[41] R. Temam. Une méthod d'approximation des solutions des équations de navier-stokes. *Bull. Soc. Math. France*, 18:49–73, 1968.

[42] S Turek. Springer, 1999.

[43] S. Turek, D. Anca, and D. Wan. Fictitious boundary and moving mesh methods for the numerical simulation of particulate flow, September 2006. Eccomas CFD 2006.

[44] S. Turek, S. Rivkind, J. Hron, and R. Glowinski. Numerical study of a modified time-steeping theta-scheme for incompressible flow simulations. 2006.

[45] S. Turek and M Schaefer. Benchmark computations of laminar flow around cylinder. *Notes on Numerical Fluid Mechanics*, 52:547–566, 1996.

[46] S. Turek and R. Schmachtel. Fully coupled and operator-splitting approaches for natural convection flows in enclosures. 2002.

[47] S. Turek, D.C. Wan, and L.S. Rivkind. An efficient multigrid fem solution technique for incompressible flow with moving rigid bodies. *Numerical Mathematics and Advanced Applications*, 35:844–853, 2003.

[48] S. Turek, D.C. Wan, and L.S. Rivkind. The fictitious boundary method for the implicit treatment of dirichlet boundary conditions with applications to incompressible flow simulations. *Lecture Notes in Computational Science and Engineering*, 35:37–68, 2003.

[49] S.P. Vanka. Implicit multigrid solution of navier stokes equations in primitive variables. *J. of Comp. Phys.*, 65:138–158, 1985.

[50] D. Wan, S. Turek, and L. Rivkind. An efficient multigrid FEM solution technique for incompressible flow with moving rigid bodies. In M. Feistauer, V. Dolejsi, P. Knobloch, and K. Najzar, editors, *Numerical Mathematics and Advanced Applications*, pages 844–853. Springer, 2003. Enumath 2003 Prague; ISBN-Nr. 3-540-21460-7.