Escola de Camins
Escola Tècnica Superior d'Enginyeria de Camins, Canals i Ports
UPC BARCELONATECH

Approximate Dirichlet boundary conditions in time-evolving physical domains

Treball realitzat per:
Guillermo Arregui Bravo

Dirigit per:
Ramon Codina Rovira
Joan Baiges Aznar

Màster en:
Enginyeria de Camins, Canals i Ports

Barcelona, 23 de Juny de 2016

Departament d'Enginyeria Civil i Ambiental

TREBALL FINAL DE MÀSTER

Abstract

In many coupled fluid-structure problems of practical interest the domain of at least one of the problems evolves in time. The most often applied approach to deal with such motion in numerical methods is the use of the Arbitrary Eulerian-Lagrangian (ALE) framework. However, the use of a pure fixed-mesh could allow us to get rid of choosing the arbitrary mesh velocity and is often more adapted for fluid motion. Nevertheless, several other problems arise from such a strategy when applied to the finite element method (FEM), especially regarding the application of Dirichlet boundary conditions (BCs).

The usual strategy to prescribe Dirichlet BCs in FEM is to define the boundary of the domain by placing nodes and facets on it, which allows us to strongly impose the condition by defining the unknown as such in the given nodes. However, it is quite straightforward to see that if a fixed-mesh strategy is to be used on, for example, a solid moving domain inside a fluid, non-matching or non-conforming grids will appear at every time step of calculation. In this cases, the boundary geometry intersects the boundary cells in an arbitrary way. To solve such a problem several techniques have been developed, such as the immersed boundary method, the penalty method, Nitsche's method, the use of Lagrange multipliers and other techniques that combine several of these strategies. These methods impose the BCs in an approximate way once the discretization has been carried out, either by modifying the differential operators near the interface (in finite differences) or by modifying the unknowns near the interface.

In this work we describe such numerical techniques for approximating Dirichlet BCs for the transient incompressible Navier-Stokes (N-S) equations. Some of these techniques have been programmed on FEMUSS (Finite Element Method Using Subscale Stabilitzation), one of the multiphysics Fortran code used at the International Centre for Numerical Methods in Engineering (CIMNE) and we applied them to study the flow of fluids around time-evolving prescribed solids.

Acknowledgements

I would like in this preamble to show my gratitude to the people that have helped me to give a meaningful content to this report and that have contributed to improve my knowledge in the field of numerical methods during these last four months.

I would like to sincerely thank Prof. Ramon Codina who, as the tutor of this thesis, was very attentive and helpful throughout the semester, always willing to cast some light on the non-trivial theoretical aspects of the incompressible Navier-Stokes equations. I wanted to show my strong gratitude to Dr. Joan Baiges, who co-tutored this Master Thesis and was my main reference for all aspects related to the existing code and to the implementation of new parts of it.

These acknowledgments are also for all the other researchers at CIMNE who were always ready to answer my questions with appropriate answers. I would specially like to thank the PhD student Arnau Pont for taking much of his time to help me run the simulations and understand many practical problems that arise when solving specific flow cases.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

Fluid mechanics is a large and important area of science and engineering due to the many phenomena that fall under its umbrella. The associated numerical simulation discipline, computational fluid dynamics, is an influential branch that leads to breakthroughs in designs and general understanding of how the world works. Modeling fluid behavior allows the obtention of very valuable results; from engineers that manage to design the most advanced aircraft to doctors that learn how biological substances move through the human body. At the heart of fluid mechanics and its modeling are the fundamental equations of motion, the Navier-Stokes (N-S) equations. The equations are known for over 150 years, yet their behavior is still not fully understood: mathematicians and physicists continue to search for the existence and uniqueness of its solution, with the goal of solving a Millenium prized problem [4].

In the last decade, wind energy has proved to be a promising building block for a future sustainable energy supply. Human-induced climate change and the evidence of the constraining political consequences of our strong dependence on every day less-available fossil fuels are at the forefront of this just-starting shift towards a renewable energy supply, where wind energy will definitely play a major role. Even though with a simple principle - just some well-designed blades that turn as a result of wind impacting on them and that make a shaft turn to produce energy- the assessment of wind's energy potential and impact requires solving the above-mentioned N-S problem coupled to the solid motion of the blades. The resolution of such a coupled problem requires the use of advanced numerical methods. Having a deep interest in renewable energies and in numerical schemes to solve the equations that rule real complex systems, the study of the problem of a solid, as for example a wind turbine, embedded in a flow was at the crossroad of two of my main interests. The broad goal of this Master Thesis being to learn numerical techniques to deal with time-evolving physical domains, the case of a wind turbine was always an inherent ambition.

Classical introductory courses to the Finite Element Method (FEM) as the ones I have taken stick to formulations which are hard to use while finding the solution of real complex problems. One of the usual limitations of this kind of courses is that they restrict to meshes that fit the domain of interest and, by consequent, use a strong imposition of Dirichlet boundary conditions (BCs) due to its algebraic simplicity. Nevertheless, in practical problems boundary conditions are very difficult to deal with, not only when it comes to their definition, but when it comes to implementation as well. If, for example, we have a solid with an externally imposed high-amplitude harmonic movement inside a fluid, it is easy to see that imposing Dirichlet BC in a classical/strong way will imply a lot of mesh distortion and recalculation around the position of the solid if we always want to have a fitted mesh. The possibility to use a fixed mesh appears as very appealing. However, it is quite straightforward to see that if a fixed-mesh strategy is to be used non-conforming grids will appear. In this cases, the boundary geometry intersects the boundary cells in an arbitrary way and we need to use what we call approximate BCs. Through my short academic background in numerical methods I did not have the chance to go abroad the limits of these FE introductory courses and to work on this project constituted a very interesting opportunity to do so.

In addition to its usefulness for moving domain problems, approximate BCs strategies, often called immersed boundary methods, are of extreme interest for computational cost reduction. The classical way to represent complex boundaries is the employment of unstructured meshes, where the boundary is represented by the mesh facets. Generating an unstructured mesh generates both a computational and storage overhead, especially in parallel computations requiring load-balanced domains and having a distributed memory system. This practical bottleneck constitutes another important reason for the development of ways to physically describe (or impose) boundary conditions on physical problems[5]. Being interested in computer science and complexity issues related to large-scale calculations, this work also provided a good frame for exploration.

We also wanted to emphasize that the interest of weak imposition of BCs goes beyond unfitted meshes. As it is discussed in [6, 7] strongly imposing, for example, no-slip conditions on high Reynolds number problems can be problematic since the sharp transition from the free-stream flow to the adherence boundary generates large amounts of vorticity which (at sufficient levels) can cause detachment of boundary layers. In some cases all vorticity is generated in the boundary and diffused into the fluid's interior, something that becomes very difficult to capture and very likely not resolvable on a given mesh, leading to nonphysical vortex structures [8]. Therefore, relaxation of BC imposition is sometimes important even in fitted meshes to avoid having non-physical results on FEM. It is important to have this idea in head for the further results since a very good approximation of the BC through parameter control, i.e. a *stronger* imposition, can lead to worse results in the rest of the domain.

Despite our interest to go very deep in the subject, in this study we will restrict ourselves to studying the flow of fluids across prescribed rigid solids, which is definitely not enough to fully understand the physics and mechanics of the fluid-structure interaction problem that arises from, for example, wind energy production, but can give us a first insight into the distortion of the flow by a moving solid or the fluid flow generated by a moving solid. Flow problems are usually numerically solved using finite volume techniques [9], because of their conservative nature. Nevertheless, we will be using FEM for this project.

1.2 Outline

To get a clear picture of all the numerical elements needed to solve such a problem, a progressive approach will be used. This work will be divided in six chapters, the first being this Introduction.

The structure of the report to come will be the following:

- Chapter 2 will introduce the reader to the theory necessary to understand the code as it was before the beginning of this project. We will first briefly present the incompressible Navier-Stokes equations as derived from the basic equations of Continuum Mechanics. After presenting its weak formulation and some main aspects of the existance of a solution, the standard Galerkin FEM for *space* discretization and a typical finite-difference time scheme will be introduced. Finally the Variational Multiscale Method (VMS) used to solve the problems that arise from the classical Galerkin FEM approach will be presented.

- Chapter 3 will go to the core of this work. After briefly commenting on the difficulties that arise when dealing with non-matching grids and its advantages, the chapter will be devoted to introducing the different approximate boundary condition (BC) techniques that were programmed, seeing what elements should be added or changed on the final formulation to solve and what terms need to be added on an elementary level when computing the matrices of the system at the element level. The convergence order and the error on the approximation of the BC of the different methods will be assessed numerically. Some benchmark will be used to confirm the validity of the methods.

- Chapter 4 will show the results of applying the diverse methods to a 2D problem consisting of a fan rotating on an cavity and to a rotating fan embedded on a piped-flow. This case was used for two main purposes; first, it is in some way a 2D version of the geometrically more complicated problems that we would like to deal with by using these approximate BC techniques and, secondly, it provides a quicker and visually easier way to check the added parts to the code than large-scale 3D simulations. Since FEMUSS is a vast FEM code used by a lot of researchers at CIMNE, a lot of precautions need to be taken before uploading a new version of the code and this case was used with such a purpose. The idea was to make a simple test that the future uploads of the code should pass.

- Chapter 5 will be devoted to draw some conclusions and to discuss further possible work.

Chapter 2

Theoretical background

2.1 The incompressible Navier-Stokes equations

The beginning of this report is going to be devoted to present the most general case of the problem we want to deal with, which is solving the flow of a fluid in a given domain, in our case the flow of air around a prescribed solid. Since the beginning we will assume already some basic knowledge in continuum mechanics from the reader. Usually three equations including energy balance are considered to solve mechanical problems in a domain, but we will here consider isothermal problems. The limit of this hypothesis will be discussed by the end of this report. Let's then first of all recall the two basic conservation laws we will need to deal with in its differential form, where we already considered temperature independent properties [10]:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho\, \mathbf{u}) = 0 \qquad \text{Conservation of mass} \tag{2.1a}$$

$$\frac{\partial (\rho\, \mathbf{u})}{\partial t} + \nabla \cdot (\rho\, \mathbf{u} \otimes \mathbf{u}) - \nabla \cdot \underline{\underline{\boldsymbol{\sigma}}} = \rho\, \mathbf{f} \qquad \text{Conversation of linear momentum} \tag{2.1b}$$

with ρ the density of the continuum, $\mathbf{u}$ its velocity, $\mathbf{f}$ the volume body forces acting on the continuum and $\underline{\underline{\boldsymbol{\sigma}}}$ the Cauchy tensor representing the internal stresses.

For the work developed in this report, these equations can be further simplified. First of all we will have to deal with a Newtonian fluid, whose Cauchy stress tensor $\underline{\underline{\boldsymbol{\sigma}}}$ can be written as $\underline{\underline{\boldsymbol{\sigma}}} = -p\, \underline{\underline{\mathbf{I}}} + 2\mu\, \nabla^{\mathbf{S}}\mathbf{u}$, where μ is the kinematic viscosity of the fluid, that we will assume as constant. We will also consider incompressibility, which implies a constant ρ in space and time. Therefore, the incompressible Navier-Stokes (N-S) problem for a Newtonian fluid consist of finding a velocity $\mathbf{u}$ and a pressure p solution of the non-linear

Cauchy and boundary value problem[1]

$$\rho \frac{\partial \mathbf{u}}{\partial t} + \rho \mathbf{u} \cdot \nabla \mathbf{u} - \mu \boldsymbol{\Delta} \mathbf{u} + \nabla p = \rho \mathbf{f} \qquad \text{in } \Omega_t, \quad \text{t>0} \tag{2.2a}$$

$$\nabla \cdot \mathbf{u} = 0 \qquad \text{in } \Omega_t, \quad \text{t>0} \tag{2.2b}$$

$$\mathbf{u} = \mathbf{u}_d \qquad \text{on } \Gamma_d, \quad \text{t>0} \tag{2.2c}$$

$$\mathbf{n} \cdot [2\mu \nabla^{\mathbf{S}} \mathbf{u} - p\mathbf{I}] = \mathbf{g}_n \qquad \text{on } \Gamma_n, \quad \text{t>0} \tag{2.2d}$$

$$\mathbf{u} = \mathbf{u}_0 \qquad \text{in } \Omega_t, \quad \text{t=0} \tag{2.2e}$$

where $\mathbf{u}_0$ is the initial condition that satisfies the incompressibility condition [2], Γ_d represents the part of the boundary where Dirichlet boundary conditions (BCs) apply, Γ_n the part where Neumann[3] BCs apply and the subindex t in Ω_t is used to show that the domain can *change* over time. More general BCs can be used to solve the incompressible N-S equations by dividing velocity and stress into their normal and tangent components or using combined boundary conditions often called Robin-type conditions. However the extension to those is straightforward.

The first equation imposes Newton's fundamental dynamics law in a continuum observed in an Eulerian referential, which means looking at the fields of interest at given points in space. The first term in the equation represents the local variation of the velocity in a point, while the second arises from the convective term that appears when computing the material derivative of the velocity. The other terms represent the different forces acting over a point in the fluid, which are internal forces related to pressure or viscosity and external volume forces which are grouped under term **f**. The second equation imposes the incompressible nature of the given flow, which can result from both a pure physical material property or from low Mach number flows. The latter explains the incompressible nature of the standard flow of air around a wind turbine in typical wind conditions that we would like to end up explaining and therefore justifies the hypothesis of working with the incompressible N-S equations throughout this Master Thesis. These two equations with the given BCs are the problem that we will like to solve, despite the fact of being unable to guarantee the well-posedness of the boundary value problem[4].

Let us assume, for the sake of simplicity, that $\Gamma_n = 0$ and $\Gamma_d \equiv \Gamma$. The extensions to the most general case require slight modifications, but the general methodology is easier to

[1] It is important to note now that first equation could be treated numerically under the given form or explicitly treating the divergence of the symmetric gradient of $\mathbf{u}$, since the incompressibility condition (2.2b) that allows us to write equation (2.2a) with only the $\boldsymbol{\Delta}\mathbf{u}$ term is not perfectly satisfied once the problem is discretized.

[2] A commonly applied initial condition is the still fluid, which implies that several time steps need to be computed before having a fully-developed flow.

[3] As it can be seen using the definition of the stress tensor for Newtonian fluids, those are imposed traction conditions.

[4] It is as mentioned in Chapter 1 one of the most well-known unsolved mathematical problems. It has in fact been proven for two dimensions but not still for a general 3D problem.

understand doing these assumptions and adapts better to the goal of this project, which is to define methods to weakly impose Dirichlet BCs.

We know that the first step to construct valid numerical methods in the framework of the Finite Element Method (FEM) is to find the weak or variational formulation of the problem in question by testing with a given function $\mathbf{v}$. Let us first define the following spaces [11]:

$$H^1(\Omega) = \{v \in L^2(\Omega) \,|\, \frac{\partial v}{\partial x_i} \in L^2(\Omega), \forall i = 1, ..., d\} \tag{2.3a}$$

$$H_0^1(\Omega) = \{v \in H^1(\Omega) \,|\, v = 0 \text{ in } \Gamma = \partial\Omega\} \tag{2.3b}$$

$$Q = L_0^2(\Omega) = \{q \in L^2(\Omega) \,|\, \int_\Omega q \, d\Omega = 0\} \tag{2.3c}$$

where the derivatives $\frac{\partial}{\partial x_i}$ need to be understood as weak derivatives.

Let us now introduce $V = H^1(\Omega)^d$, $V_0 = H_0^1(\Omega)^d$ the generalization of the spaces given to vector fields in $\mathbb{R}^d$. Let us also introduce $L^2(0,T;V)$ the set of functions whose V-norm in space is $L^2((0,T])$ in time, $L^\infty(0,T;L^2)$ the functions whose L^2-norm in space is $L^\infty((0,T])$ in time and $\mathcal{D}'(0,T;Q)$ the set of *functions* whose Q-norm in space is a distribution in time. The natural solution space for the variational problem to be well-written is $L^2(0,T;V) \times \mathcal{D}'(0,T;Q)$, but due to the equation itself it can be shown that $\mathbf{u}$ needs also to belong to $L^\infty(0,T;L^2)$. Then the weak form of the incompressible Navier-Stokes equations can be written as follows: find $[\mathbf{u}, p] \in (L^2(0,T;V) \cap L^\infty(0,T;L^2)) \times \mathcal{D}'(0,T;Q)$ such that

$$\rho(\partial_t \mathbf{u}, \mathbf{v}) + \rho\langle \mathbf{u} \cdot \nabla \mathbf{u}, \mathbf{v}\rangle + \mu(\nabla \mathbf{u}, \nabla \mathbf{v}) - (p, \nabla \cdot \mathbf{v}) = \rho\langle \mathbf{f}, \mathbf{v}\rangle \qquad \forall\, \mathbf{v} \in V_0 \tag{2.4a}$$

$$(q, \nabla \cdot \mathbf{u}) = 0 \qquad \forall\, q \in Q \tag{2.4b}$$

$$\mathbf{u} = \mathbf{u}_d \qquad \text{in } \Gamma \tag{2.4c}$$

where $\langle \cdot, \cdot \rangle$ is understood as the integral of the component-wise product of the two given functions and $(\cdot, \cdot)$ applies when both functions are in $L^2(\Omega)$. Some integration by parts have been carried out and the property of $\mathbf{v} = \mathbf{0}$ on Γ coming from $V_0 = H_0^1(\Omega)^d$ has been used.

The problem with this well-defined formulation is that no proof of uniqueness has been found yet for a general 3D case. Some steps have already been done towards the proof of existence, uniqueness and smoothness of a solution to the N-S equations, but still some very important issues need to be resolved. The goal of this work is obviously not to deal with a theoretical analysis of the N-S equations, but we wanted to give some of the standard ideas that anyone facing their numerical resolution should know. Even though the problem is not linear, giving the linear framework of existence and uniqueness of the solution can be interesting as we will see. Let's use here the typical notation to explain the conditions that are needed in the generalized Lax-Milgram's theorem or Banach-Nečas-Babuška (BNB) theorem[5] [12]. With W and V Hilbert spaces with respect to norms $\|\cdot\|_W$ and $\|\cdot\|_V$, let us suppose we have an abstract variational problem as follows:

$$\text{find } u \in W \text{ such that } B(u,v) = L(v) \quad \forall v \in V \tag{2.5}$$

and the following conditions are satisfied

1. $L(\cdot)$ is a continuous linear form, which implies that $v \to L(v)$ is linear from V to $\mathbb{R}$ and that $\exists\ C > 0$ such that
$$|L(v)| \leq C \, \|v\|_V, \quad \forall v \in V \tag{2.6}$$
2. $B(\cdot,\cdot)$ is a continuous bilinear form, which implies that $v \to B(u,v)$ is linear from V to $\mathbb{R}$ $\forall u \in W$, that $u \to B(u,v)$ is linear from W to $\mathbb{R}$ $\forall v \in V$ and that there $\exists$ $M > 0$ such that
$$|B(u,v)| \leq M \|u\|_W \|v\|_V, \quad \forall (u,v) \in W \times V \tag{2.7}$$
3. There exists $\gamma > 0$ such that the following inf-sup condition (often called W-stability) is satisfied:
$$\inf_{u\in W} \sup_{v \in V} \frac{B(u,v)}{\|u\|_W \, \|v\|_V} \geq \gamma \tag{2.8}$$
4. The following condition -on the adjoint operator- is satisfied:
$$\{B(u,v) = 0, \quad \forall u \in W\} \implies \{v = 0\} \tag{2.9}$$

Then there exists a unique solution to Problem (2.5).

It could be surprising that the general conditions of existence and uniqueness of a solution to a linear stationary variational problem are introduced when we are dealing with a well-known non-linear transient problem. The reason is that it happens to be true that to proof existence -not possible for uniqueness- of the solution to the variational problem (2.4), the specific form of the inf-sup condition (2.8) for the Stokes problem needs also to be satisfied [13]. The interpretation of such a condition is not straightforward, but it could eventually be seen as follows. Let us rewrite (2.4) as the abstract problem: find $[\mathbf{u}, p] \in L^2(0,T;V) \times \mathcal{D}'(0,T;Q)$ such that

$$\left(\rho \frac{\partial \mathbf{u}}{\partial t}, \mathbf{v}\right) + B([\mathbf{u},p],[\mathbf{v},q]) = L([\mathbf{v},q]), \quad \forall [\mathbf{v},q] \in V_0 \times Q \tag{2.10}$$

[5] Several versions of those conditions exist depending on the field to which they apply. They can also be called Ladyzhenskaya-Babuška-Brezzi (LBB) conditions when written on finite element spaces or just Babuška-Brezzi conditions.

where

$$B([\mathbf{u},p],[\mathbf{v},q]) = \rho\langle \mathbf{u}\cdot\nabla\mathbf{u},\mathbf{v}\rangle + \mu(\nabla\mathbf{u},\nabla\mathbf{v}) - (p,\nabla\cdot\mathbf{v}) + (q,\nabla\cdot\mathbf{u}) \tag{2.11a}$$

$$L([\mathbf{v},q]) = \rho\langle \mathbf{f},\mathbf{v}\rangle \tag{2.11b}$$

If a time discretization and a non-linearity iterative algorithm are introduced (as it will be done on Section 2.2), the resulting infinite dimension space problem can be treated in the framework described in (2.5) and will inherit many of the features of a Stokes and a convection-diffusion-reaction problem. It *makes sense* then that the conditions given need to be satisfied. This heuristic approach is far from being mathematically justified but gives the reader an idea of why the inf-sup condition, that will appear later, needs to be tackled for the N-S problem despite being a transient non-linear problem.

2.2 The Galerkin Finite Element Method for the Navier-Stokes equations

With the weak formulation derived in (2.4) the numerical problem starts. The goal of this section is to present how we discretize the different elements appearing in this problem: the physical domain, the function spaces and the variational/weak formulation. Again for the sake of simplicity, what we call triangular Lagrange finite elements of order one will be used to show how the standard 2D Galerkin FEM is constructed and what is the system that we finally have to solve. The extension to higher order elements and 3D problems follows the same idea but will make the notation harder to understand. The reader with previous knowledge on FEM will be able to skip this section if necessary.

The idea of Galerkin methods in general is very simple: we choose a functional space $V_h \times Q_h \subset V \times Q$ of finite dimension and solve for a solution $[\mathbf{u}_h, p_h]$ in the new functional space tested against functions in $V_{h,0} \times Q_h \subset V_0 \times Q$. This allows us to transform our infinite dimension problem into a finite dimension -and thus computationally resolvable- problem due to the possibility of defining functions in the space as a linear combination of the elements in a basis of the finite dimension functional space chosen. The two most well-known Galerkin methods are the Ritz method [14] and FEM, which we will be using here.

In the framework of FEM the typical approach consists in decomposing the physical domain in which the weak formulation is written and to impose a given shape to a function $u(\mathbf{x})$[6] inside this partition, that we will call $\mathcal{P}_h$, where h stands for the *size* of the mesh (the smaller h, the more refined the mesh). In 2D, a triangulation of Ω_t is a subdivision of

[6]Herein u will represent any of the components of the solution: the three components of the velocity field u_x, u_y, u_z or the pressure p.

this domain into triangles. The standard choice is to choose triangles that cover all Ω_t but no more and that fulfill the following rule: triangles having some intersection, this should be either on common vertex or a common full edge. In particular, two different triangles do not overlap. This implies, more technically said, that $\mathcal{P}_h$ is a simplical complex [11]. The partition must also respect the division of the boundary into Dirichlet or Neumann boundaries. A typical partition, or what we call mesh, for an arbitrary polygonal domain is found in Figure 2.1. Partitioning the domain represents a first approximation to the problem, since the boundary's shape can be arbitrary and not be captured by the edges of the triangles, as it can be seen in the inner circle of Figure 2.1.

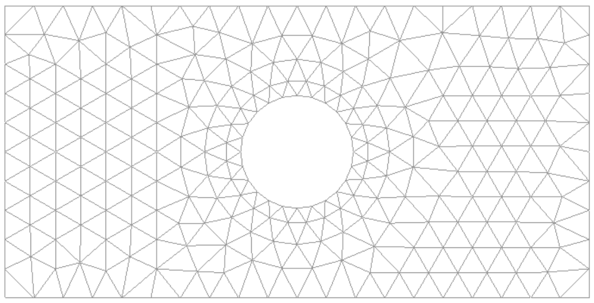


Figure 2.1: Triangulation $\mathcal{P}_h$ of a physical domain Ω

Once the partitioning $\mathcal{P}_h$ is done, what we need to do is to find the right subspace of functions V_h where we want to solve our discretized physical problem. For the sake of constructing such a space, let us for example think about polynomial functions of degree at most one in a 2D space:

$$h(x,y) = a_0 + a_1 x + a_2 y, \qquad a_0, a_1, a_2 \in \mathbb{R} \tag{2.12}$$

which are uniquely determined by its values on three different non-aligned points or the vertices of a non-degenerate triangle. The three values of the function in the vertices -that we call nodes- will be called from now on the local degrees of freedom. Another important property of (2.12) is that the value of $h \in \mathbb{P}_1$ on the edge that joins two vertices of the triangle depends only on the values of h in these two nodes. Therefore, we can think of taking two triangles K and K' sharing an edge and constructing the function that is linear in each triangle and has the given values in the vertices. Since the value on the common edge depends only in the values in the nodes, this function will be continuous. This can be done triangle by triangle until we end up with a piecewise linear function that is overall continuous. We have ended up constructing the following space:

$$V_h = \{u_h \in \mathscr{C}(\overline{\Omega}) \mid u_h|_K \in \mathbb{P}_1, \forall K \in \mathcal{P}_h \}$$

If the values on the set of vertices/nodes are fixed there exists a unique $u_h \in V_h$ with those values in the vertices. Even though the construction of the space has been carried out for triangular linear elements, the procedure done can be understood for higher order elements if the proper points are added in the elements.

Let $\mathbf{p}_i$ denote the nodes of our domain, with i going from 1 to the number of nodes N_{nodes}. Due to what has just been explained, if a node is fixed and we associate the value one to this node and zero to all other nodes there exists a unique function $\phi_i \in V_h$ that has these values, that is,

$$\phi_i(\mathbf{p}_i) = \delta_{i,j} = \{ \begin{array}{ll} 1, & j = i \\ 0, & j \neq i \end{array} \tag{2.13}$$

as can be seen in Figure 2.2. Let us define N_{nodes} as the total number of nodes and $n_{Dirichlet}$ as the nodes where Dirichlet boundary conditions for u are imposed. With the help of the set of functions $\boldsymbol{\phi} = \{\phi_i \ , i = 1, ..., N_{nodes}\}$ ordered in such a way that we have put at the end all Dirichlet-prescribed nodes, we have constructed a basis of our finite element space that has the advantage of letting us write the solution in the following way

$$u_h = \sum_{i=1}^{N_{free}} u_h(\mathbf{p}_i)\phi_i + \sum_{j=N_{free}}^{N_{nodes}} u_d(\mathbf{p}_j)\phi_j \tag{2.14}$$

where $N_{free} = N_{nodes} - n_{Dirichlet}$. This means that the coefficients of the solution on the basis $\boldsymbol{\phi}$ are nodal values of the function. A possible generalization of this basis to our finite dimension solution space $V_h \times Q_h$ for the incompressible N-S equations in 2D could be as follows:

$$\boldsymbol{\phi} = \{(\phi_i, 0, 0), (0, \phi_i, 0), (0, 0, \phi_i) \, , i = 1, ..., N_{nodes}\} \tag{2.15}$$

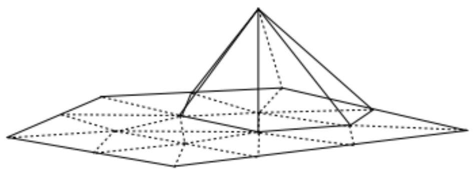


Figure 2.2: Basis function ϕ_i for linear triangular elements

A Galerkin FEM method for the 2D incompressible N-S equations consists in replacing in (2.4) $\mathbf{u}$ and p by

$$\mathbf{u}_h = (\sum_{i=1}^{N_{free}} u_i^x\phi_i + \sum_{j=N_{free}}^{N_{nodes}} u_d^x(\mathbf{p}_j)\phi_j, \sum_{i=1}^{N_{free}} u_i^y\phi_i + \sum_{j=N_{free}}^{N_{nodes}} u_d^y(\mathbf{p}_j)\phi_j) \tag{2.16}$$

$$p_h = \sum_{i=1}^{N_{nodes}} p_i\phi_i \tag{2.17}$$

where we have supposed as in (2.4) that Dirichlet BCs are imposed on all components of the velocity $\mathbf{u}_d$. If it were not the case we would need to take different $n_{Dirichlet}$ depending on the direction.

Since the test functions $[\mathbf{v}_h, q_h]$ are taken in $V_{h,0} \times Q_h$ we can use the functions in the already mentioned basis, except those centered in nodes in the Dirichlet boundaries where we already now the value of the function, to test the weak form (2.4) of our problem once $\mathbf{u}_h$ and p_h have been inserted. We are finally left with a set of $(N_{dimension} + 1) \cdot N_{nodes} - N_{dimension} \cdot n_{Dirichlet}$ time-dependant non-linear equations. For the system of equations to be written algebraically as a function of the unknown nodal values of fields $\mathbf{u}_h$ and p_h a discretization in time needs to be used. Here many choices are possible, but for the sake of simplicity in this quick review on how to numerically solve the incompressible N-S equations, we will use a one step first order Euler backward difference time discretization which consists in saying that

$$\frac{\partial \mathbf{u}_h^{n+1}}{\partial t} = \frac{\mathbf{u}_h^{n+1} - \mathbf{u}_h^n}{\Delta t} + \mathcal{O}(\Delta t) \tag{2.18}$$

and taking all other terms in the equation at step $n+1$. This scheme is unconditionally stable but is in reality rarely used since it is only first order. Nevertheless, it allows for a simple and clear writing of the system. Other schemes taking the $\mathbf{u}$-terms in the convective part on different time steps could lead to avoiding non-linearity but those are very seldom used and we decided to deal with non-linearity in an explicit way.

Once this simple time discretization performed, the system to solve has the following form

$$\frac{1}{\Delta t}\mathbf{D}\mathbf{U}^{n+1} - \frac{1}{\Delta t}\mathbf{D}\mathbf{U}^{n} + \mathbf{S}\mathbf{U}^{n+1} + \mathbf{N}(\mathbf{U}^{n+1}) - \mathbf{L}^{\mathbf{T}}\mathbf{P}^{n+1} = \mathbf{F}^{n+1} \tag{2.19a}$$

$$\mathbf{L}\mathbf{U}^{n+1} = \mathbf{0} \tag{2.19b}$$

where $\mathbf{U}^{n+1}$ denotes the vector of unknowns composed of u_i^x and u_i^y for all unknown nodes at time $(n+1)\Delta t$ and $\mathbf{P}^{n+1}$ denotes the vector of unknowns p_i at the same time step. The first terms represent terms coming from the time derivation, $\mathbf{S}\mathbf{U}^{n+1}$ the discretization of viscous terms, $\mathbf{N}(\mathbf{U}^{n+1})$ the discretization of the non-linear convective terms, $\mathbf{L}\mathbf{U}^{n+1}$ stands for the discretization of the divergence of $\mathbf{u}$ and $-\mathbf{L}^{\mathbf{T}}\mathbf{P}^{n+1}$ discretizes the gradient of p. On the other side, the right hand side comprises contributions from the source term, from the Dirichlet prescribed nodes and eventually terms from Neumann prescribed BCs if the assumption on Γ_n had not been done in Section 2.1. We want to find ourselves with a linear system of equations, therefore the non-linear term $\mathbf{N}(\mathbf{U}^{n+1})$ needs to be treated. Several ways of dealing with non-linearity independently of time integration exist, among which the most well known are Newton, quasi-Newton and Picard type iterative methods. The general solving of the non-linear system of equations consists mainly in using an iterative procedure which linearizes the system according to one of the mentioned methods. We will here restrict ourselves to show how Picard's method works for Equation (2.19a). The non-linear term $\mathbf{u} \cdot \nabla \mathbf{u}$ at iteration $k+1$ can render the system linear if one of the two **u**-dependant terms in or both are taken from the previous iteration. The following possibilities are available,

$$(\mathbf{u} \cdot \nabla \mathbf{u})^{k+1} \simeq \mathbf{u}^{k+1} \cdot \nabla \mathbf{u}^{k} \tag{2.20a}$$
$$\simeq \mathbf{u}^{k} \cdot \nabla \mathbf{u}^{k+1} \tag{2.20b}$$
$$\simeq \mathbf{u}^{k} \cdot \nabla \mathbf{u}^{k} \tag{2.20c}$$

Several numerical experiments [15] have shown that the only constructor showing good convergence is (2.20b). Picard's method has the advantage that the initial estimate does not need to be in a neighborhood of the final solution in comparison with Newton method, which is in fact a linearization technique that uses a linear combination of the terms (2.20). Therefore a simple Stokes-like problem with convection can be solved in order to produce an initial guess for the iterative procedure even though this can lead to convergence problems for high Reynolds number flows where convection predominates. The final system of linear equations to solve reads as follows: find $\mathbf{U}_{n+1}^{k+1}$ and $\mathbf{P}_{n+1}^{k+1}$ such that,

$$\frac{1}{\Delta t}\mathbf{D}\mathbf{U}_{n+1}^{k+1} - \frac{1}{\Delta t}\mathbf{D}\mathbf{U}_{n} + \mathbf{S}\mathbf{U}_{n+1}^{k+1} + \mathbf{N}(\mathbf{U}_{n+1}^{k})\mathbf{U}_{n+1}^{k+1} - \mathbf{L}^{\mathbf{T}}\mathbf{P}_{n+1}^{k+1} = \mathbf{F}_{n+1} \tag{2.21a}$$
$$\mathbf{L}\mathbf{U}_{n+1}^{k+1} = \mathbf{0} \tag{2.21b}$$

It is important to note that in Equation (2.21b) the number of rows and columns that define the matrix is imposed by the order of the element that is chosen to interpolate velocity and pressure respectively. The reader will have noted that the writing in (2.15) implies that we have implicitly decided to choose same interpolation for velocity and pressure, however this choice is arbitrary. To have a non-singular matrix out of (2.21b) we see that not all elements are capable of giving a well-posed system since we need to ensure that the number of pressure unknowns never exceeds the number of velocity unknowns. This is automatically satisfied for the interpolation orders chosen. Nevertheless, our same order interpolation scheme leads to some complications that will be studied later.

Since the methods that have been programmed and that will be later discussed in Chapter 3 happen to add terms to the above matrices, we will introduce here the notation that we will be using in this report. The system to be solved is:

$$\begin{pmatrix} \mathbf{K_{uv}} & \mathbf{K_{pv}} \\ \mathbf{K_{uq}} & \mathbf{0} \end{pmatrix} \cdot \begin{pmatrix} \mathbf{U}_{n+1}^{k+1} \\ \mathbf{P}_{n+1}^{k+1} \end{pmatrix} = \begin{pmatrix} \hat{\mathbf{F}}_{n+1} \\ \mathbf{0} \end{pmatrix} \tag{2.22}$$

where the 4th block matrix is $\mathbf{0}$ since we have no terms for p_h coming from testing with

q_h. Obviously we have:

$$\mathbf{K_{uv}} = \frac{1}{\Delta t}\mathbf{D} + \mathbf{S} + \mathbf{N}(\mathbf{U}^k_{n+1}) \tag{2.23a}$$

$$\mathbf{K_{pv}} = \mathbf{L^T} \tag{2.23b}$$

$$\mathbf{K_{uq}} = \mathbf{L} \tag{2.23c}$$

$$\hat{\mathbf{F}}_{n+1} = \mathbf{F}_{n+1} + \frac{1}{\Delta t}\mathbf{D}\mathbf{U}_n \tag{2.23d}$$

for the simple time integration and non-linearity schemes that have been used. Extension to more complicated schemes only requires some simple algebraic manipulation. In (2.23) no subscript for iteration is used for $\mathbf{F}$ because the integrals defining the matrices do not depend on the iteration. If the following arrangement of the degrees of freedom (DOFs) is chosen:

$$\begin{pmatrix} \mathbf{K}_{u^x v^x} & \mathbf{K}_{u^y v^x} & \mathbf{K}_{pv^x} \\ \mathbf{K}_{u^x v^y} & \mathbf{K}_{u^y v^y} & \mathbf{K}_{pv^y} \\ \mathbf{K}_{u^x q} & \mathbf{K}_{u^y q} & \mathbf{0} \end{pmatrix} \cdot \begin{pmatrix} \mathbf{U}^{k+1}_{x,n+1} \\ \mathbf{U}^{k+1}_{y,n+1} \\ \mathbf{P}^{k+1}_{n+1} \end{pmatrix} = \begin{pmatrix} \hat{\mathbf{F}}_{v^x,n+1} \\ \hat{\mathbf{F}}_{v^y,n+1} \\ \mathbf{0} \end{pmatrix} \tag{2.24}$$

then the definition of the matrices for the introduced numerical scheme correspond to the ones in Table 2.1.

Block UV
$[\mathbf{K}_{u^x v^x}]_{ij} = \frac{\rho}{\Delta t}\int_\Omega \phi_i\phi_j + \rho\int_\Omega u^x_{k,n+1}\frac{\partial\phi_j}{\partial x}\phi_i - \mu\int_\Omega \nabla\phi_j\cdot\nabla\phi_i$ $[\mathbf{K}_{u^y v^y}]_{ij} = \frac{\rho}{\Delta t}\int_\Omega \phi_i\phi_j + \rho\int_\Omega u^y_{k,n+1}\frac{\partial\phi_j}{\partial y}\phi_i - \mu\int_\Omega \nabla\phi_j\cdot\nabla\phi_i$ $[\mathbf{K}_{u^y v^x}]_{ij} = \rho\int_\Omega u^y_{k,n+1}\frac{\partial\phi_j}{\partial x}\phi_i$ $[\mathbf{K}_{u^x v^y}]_{ij} = \rho\int_\Omega u^x_{k,n+1}\frac{\partial\phi_j}{\partial y}\phi_i$
Block PV
$[\mathbf{K}_{pv^x}]_{ij} = \int_\Omega \phi_j\frac{\partial\phi_i}{\partial x}$ $[\mathbf{K}_{pv^y}]_{ij} = \int_\Omega \phi_j\frac{\partial\phi_i}{\partial y}$
Block UQ
$[\mathbf{K}_{u^x q}]_{ij} = \int_\Omega \phi_i\frac{\partial\phi_j}{\partial x}$ $[\mathbf{K}_{u^y q}]_{ij} = \int_\Omega \phi_i\frac{\partial\phi_j}{\partial y}$
RHS $\mathbf{F_V}$
$[\hat{\mathbf{F}}_{v^x,n+1}]_j = \int_\Omega (f^x + u^x_n)\phi_j - \sum_{m=N_{free}}^{N_{nodes}} \frac{\rho}{\Delta t}\int_\Omega u^x_d(\mathbf{p}_m)\phi_m\phi_j +$

$\rho \int_\Omega (\mathbf{u}_{k,n+1}\cdot\mathbf{u}_d(\mathbf{p}_m))\frac{\partial\phi_m}{\partial x}\phi_j - \mu\int_\Omega u_d^x(\mathbf{p}_m)\nabla\phi_m\cdot\nabla\phi_j$
$[\hat{\mathbf{F}}_{v^y,n+1}]_j = \int_\Omega (f^y + u_n^y)\phi_j - \sum_{m=N_{free}}^{N_{nodes}} \frac{\rho}{\Delta t}\int_\Omega u_d^y(\mathbf{p}_m)\phi_m\phi_j$
$+\ \rho \int_\Omega (\mathbf{u}_{k,n+1}\cdot\mathbf{u}_d(\mathbf{p}_m))\frac{\partial\phi_m}{\partial y}\phi_j - \mu\int_\Omega u_d^y(\mathbf{p}_m)\nabla\phi_m\cdot\nabla\phi_j$

Table 2.1: Definition of the matrices and vectors required to construct the linear system of equations to be solved at time step t_{n+1} in iteration $k+1$.

One of the various strengths of FEM is its capacity to construct the matrices in Table 2.1 in a very straightforward way. Due to the choice of functions (2.15), the integrals defining the coefficients of the above matrices vanish in most of the cases since no function overlap between ϕ_i and ϕ_j exists unless the nodes are connected. This allows the matrix to be computed by what is called an *assembly* process which uses shape functions $N_i(\mathbf{x})$ at an element level, those being the restrictions to element K of the test functions ϕ_i which have K in their support. Afterwards the matrix of connectivity is used to place the elements in the elemental matrix $\mathbf{K_{elem}}$ in the appropriate place in $\mathbf{K}$. We will not get herein in the details of such construction since it is a very standard process and we refer the reader to reference [16] or any other basic FEM introductory course if the notion is unknown. Furthermore, the computation of the integrals that define $\mathbf{K_{elem}}$ also raise a problem. We need either and effective way of evaluating the shape functions $N_i(\mathbf{x})$ or a closed form for the resulting integrals. Both possibilities are done usually by moving to the so-called reference element and assuming an isoparametric transformation [17], which means a change of variables using the same interpolation order as the order of the FE method itself. Once the integral is written in the reference variables they need to be calculated numerically and we have in this work used the standard Gauss quadratures.

With the system of equations to solve written in (2.22) and a way to calculate all terms in Table 2.1 we could think now that the problem is ready to be solved using any linear system solver, except that one of the steps at the beginning of our system construction was not appropriate. The standard Galerkin approximation as defined above does not ensure stability of our system. Instability sources are the following:

- The convective term in singularly perturbed problems, which leads to oscillating solutions when the Reynolds cell number is too large and requiring extremely fine meshes to be overcomed.
- The requirement of compatibility conditions between the velocity and the pressure approximation spaces due to the BNB inf-sup conditions discussed in Section 2.1 cannot be guaranteed with the Galerkin formulation using same interpolation order for pressure and velocity.

We already stated back in Section 2.2 that the inf-sup condition would appear later and already mentioned that the N-S transient incompressible equations are in some way a generalization of the Stokes equations and the convection-diffusion-reaction equations. Therefore, in terms of the conditions that the first need to satisfy for solutions to exist, the latter will naturally give a lot of information. The condition (2.8) applied to the continuous Stokes problem can be written as the two following conditions:

$$\|\nabla \mathbf{v}\|_{L^2}^2 \geq \eta \|\mathbf{v}\|_V^2, \quad \forall \mathbf{v} \in V \tag{2.25a}$$

$$\inf_{q\in Q} \sup_{\mathbf{v}\in V} \frac{(q, \nabla \cdot \mathbf{v})}{\|q\|_Q \, \|\mathbf{v}\|_V} \geq \gamma_* > 0 \tag{2.25b}$$

where the first condition is the coercivity of the bilineal form $a(\mathbf{u}, \mathbf{v}) = (\nabla \mathbf{u}, \nabla \mathbf{v})_\Omega$ and the second a new inf-sup condition that arises from the most general one. The spaces V and Q are the same ones as for the N-S problem.

Now we see that the choice of FE spaces for the pressure p and velocity $\mathbf{u}$ is in fact of extreme importance since the inf-sup condition in (2.25) needs to be satisfied on the discrete space:

$$\inf_{q_h\in Q_h} \sup_{\mathbf{v}_h\in V_h} \frac{(q_h, \nabla \cdot \mathbf{v}_h)}{\|q_h\|_{Q_h} \, \|\mathbf{v}_h\|_{V_h}} \geq \gamma_* \tag{2.26}$$

the problem being that the condition above is not inherited from the infinite dimensional counterpart in (2.25) except for some special circumstances, while the coercivity is directly inherited. One, therefore, has to prove the non-trivial discrete inf-sup condition (2.26). When conforming spaces are chosen such that $V_h \subset V$ and $Q_h \subset Q$ the problem that arises is that the supreme and the infimum can live in different spaces and therefore the conditions do not hold. For the Galerkin FEM used on the incompressible N-S equations this is the case except for some very special interpolation spaces, whose pairs $V_h - Q_h$ will be called *compatible* and will satisfy the condition by construction. For the simple case of equivalent interpolation space we have chosen to work with, this condition does not hold and therefore something needs to be done to guarantee condition (2.26) [18].

Two possible approaches exist to solve the mentioned problem. The most straightforward is obviously to choose one of the aforesaid *compatible* pairs. Nevertheless, the choice for same interpolation has not been arbitrary; it allows for much simplicity in implementation and reduces computational cost. Therefore another approach will be chosen here, which consists on writing the variational problem (2.11) in a way that the form $B([\mathbf{u}, p], [\mathbf{v}, q])$ can guarantee stability of all the terms. This will allow us as well to have control of the convective term, since for the standard form of the problem no specific condition is known to guarantee the stability with respect to the advection velocity $\mathbf{a} = \mathbf{u}_{h,n}^k$. Even more, we do not know any error estimate for the convection-diffusion-reaction equation not exploding for $|\mathbf{a}| \gg \nu$ [12], which *shows* the necessity of writing the variational problem for the N-S equations in a different way.

In the next section we will present the Variational Multiscale Method (VMS) used to stabilize the standard Galerkin approximation of the incompressible N-S weak formulation, which will ensure both stabilizing the convective term and satisfying the resulting inf-sup condition. Even though the underlying mathematical content will not be assessed in depth and the reader could sometimes have the impression of lack of justification, it is important to get an insight into the formulation since it is the basis of the formulation programmed in FEMUSS that we will be using later on. The interested reader can refer to the chapter accorded to this issue in [3] for a more general and detailed explanation.

2.3 The Variational Multiscale Method (VMS)

This section presents the Variational Multiscale (VMS) framework via its application to the problem of interest. Introduced in the 90s [19], this framework has been applied to design stabilized FEM in problems in which stability of the standard Galerkin method is not ensured, as are the incompressible N-S equations with equal interpolation. Most of what will be said in this section is taken from the chapter accorded to the issue in [3], written precisely by the tutors of this thesis.

The key idea of the VMS approach is to split the space where the solution lives into a finite element space and a subscale space. For our space $V \times Q$ this writes as:

$$V \times Q = (V_h \oplus V^{'}) \times (Q_h \oplus Q^{'}) \tag{2.27}$$

where spaces with subindex h are the FE space and spaces with superindex $'$ are any complement to them in $V \times Q$. A particular VMS method is precisely defined by the way the subscale space $V^{'}$ and $Q^{'}$ are approximated from the FE spaces V_h and Q_h. For the moment we let $V^{'}$ and $Q^{'}$ be the complementary spaces, but later on the notation will be used for an approximation of the same.

The obvious consequence of splitting the functional space will be the associated splitting of the unknowns and tests functions. For the sake of simplicity and for practical reasons[7] we take $Q^{'} \simeq 0$ and we will only perform a splitting on velocity. If we take $\mathbf{u} = \mathbf{u}_h + \mathbf{u}^{'}$ and $\mathbf{v} = \mathbf{v}_h + \mathbf{v}^{'}$ in (2.11), the problem can be written as follows:

$$\begin{aligned} \rho(\partial_t \mathbf{u}_h, \mathbf{v}_h) + \rho(\partial_t \mathbf{u}^{'}, \mathbf{v}_h) + \rho\langle \mathbf{u} \cdot \nabla \mathbf{u}_h, \mathbf{v}_h \rangle + \rho\langle \mathbf{u} \cdot \nabla \mathbf{u}^{'}, \mathbf{v}_h \rangle + \mu(\nabla \mathbf{u}_h, \nabla \mathbf{v}_h) + \mu(\nabla \mathbf{u}^{'}, \nabla \mathbf{v}_h) \\ -(p_h, \nabla \cdot \mathbf{v}_h) + (q_h, \nabla \cdot \mathbf{u}_h) + (q_h, \nabla \cdot \mathbf{u}^{'}) = \rho\langle \mathbf{f}, \mathbf{v}_h \rangle, \quad \forall\, (\mathbf{v}_h, q_h) \in V_{0,h} \times Q_h \end{aligned} \tag{2.28}$$

$$\begin{aligned} \rho(\partial_t \mathbf{u}_h, \mathbf{v}^{'}) + \rho(\partial_t \mathbf{u}^{'}, \mathbf{v}^{'}) + \rho\langle \mathbf{u} \cdot \nabla \mathbf{u}_h, \mathbf{v}^{'} \rangle + \rho\langle \mathbf{u} \cdot \nabla \mathbf{u}^{'}, \mathbf{v}^{'} \rangle + \mu(\nabla \mathbf{u}_h, \nabla \mathbf{v}^{'}) \\ +\mu(\nabla \mathbf{u}^{'}, \nabla \mathbf{v}^{'}) - (p_h, \nabla \cdot \mathbf{v}^{'}) = \rho\langle \mathbf{f}, \mathbf{v}^{'} \rangle, \qquad \forall\, \mathbf{v}^{'} \in V_0^{'} \end{aligned} \tag{2.29}$$

[7]Instability is mostly associated to the convective term, therefore a splitting only in velocity can both ensure stability for singularly perturbed problems and for same interpolation order spaces for pressure and velocity

where the reader will note that in the convective term we have not developed explicitly the splitting in the advection velocity. This is because we are going to do the assumption that the advection velocity $\mathbf{u} = \mathbf{u}_* \simeq \mathbf{u}_h$. This assumption will simplify the following developments and makes sense if we recall that the equations will need to be linearized at some point using (2.20b). The original VMS formulation [19] was in fact developed having linear problems in mind and its extension to the N-S equations was implicitly based on a *linearization*, fixing the advection velocity and applying the multiscale splitting to the rest of the terms. We will now use Equation (2.29) to construct an approximation of $\mathbf{u}^{'}$ as a function of $\mathbf{u}_h$, which will require the definition of $V^{'}$ as a function of V_h and by prolongation a definition of $V_0^{'}$ as a function of $V_{0,h}$. We will first decompose the domain Ω with a FE partition $\mathcal{P}_h$ and rewrite the inner-products in the equations above as sums over elements $K \in \mathcal{P}_h$ of inner products in K, i.e. $(\cdot,\cdot)_\Omega = \sum_K (\cdot,\cdot)_K$. Equation (2.29) now reads the following way:

$$\begin{split}\rho(\partial_t \mathbf{u}^{'}, \mathbf{v}^{'}) + \sum_K \Big[\rho(\partial_t \mathbf{u}_h, \mathbf{v}^{'})_K + \rho\langle \mathbf{u}_* \cdot \nabla \mathbf{u}_h, \mathbf{v}^{'}\rangle_K + \rho\langle \mathbf{u}_* \cdot \nabla \mathbf{u}^{'}, \mathbf{v}^{'}\rangle_K + \overbrace{\mu(\nabla \mathbf{u}_h, \nabla \mathbf{v}^{'})_K}^{A} \\ + \overbrace{\mu(\nabla \mathbf{u}^{'}, \nabla \mathbf{v}^{'})_K}^{B} - \overbrace{(p_h, \nabla\cdot \mathbf{v}^{'})_K}^{C}\Big] = \rho\langle \mathbf{f}, \mathbf{v}^{'}\rangle, \qquad \forall \mathbf{v}^{'} \in V_0^{'}\end{split} \tag{2.30}$$

where we are going to do K-wise integration by parts on terms A, B and C in order to get an equation which is fully written again with $\mathbf{v}^{'}$ on the second argument of the inner-products. After i.b.p. the equation leads to

$$\begin{split}\overbrace{\rho(\partial_t \mathbf{u}^{'}, \mathbf{v}^{'})}^{1} + \overbrace{\sum_K (\rho\partial_t \mathbf{u}_h + \rho \mathbf{u}_* \cdot \nabla \mathbf{u}_h - \mu \boldsymbol{\Delta} \mathbf{u}_h + \nabla p_h, \mathbf{v}^{'})_K}^{2} + \overbrace{\sum_K (\rho \mathbf{u}_* \cdot \nabla \mathbf{u}^{'} - \mu \boldsymbol{\Delta} \mathbf{u}^{'}, \mathbf{v}^{'})_K}^{3} \\ + \overbrace{\sum_K (\mu \mathbf{n} \cdot \nabla \mathbf{u}_h - p_h \mathbf{n}, \mathbf{v}^{'})_{\partial K}}^{4} + \overbrace{\sum_K (\mu \mathbf{n} \cdot \nabla \mathbf{u}^{'}, \mathbf{v}^{'})_{\partial K}}^{5} = \overbrace{\rho\langle \mathbf{f}, \mathbf{v}^{'}\rangle}^{6}, \qquad \forall \mathbf{v}^{'} \in V_0^{'}\end{split} \tag{2.31}$$

Now, some assumptions can be done in order to further simplify Equation (2.31). Those are the following:

- We will deal with what we call *quasi-static* subscales, which mean neglecting term 1 [20].
- For every element K, we choose $\mathbf{u}^{'}$ and $\mathbf{v}^{'}$ to be zero in the element boundary ∂K [21]. Therefore terms 4 and 5 can be neglected. This is usually stated as choosing $V^{'}$ as a *bubble* functions space.
- As already stated, we choose $\mathbf{u}_*$ to be equal to $\mathbf{u}_h$

With these assumptions we can group terms 2 and 6 to make the residual $\mathcal{R}([\mathbf{u}_h, p_h])$ of the momentum equation for $[\mathbf{u_h}, p_h]$ appear. The reader will note that now we have a set of N_{elem} independent problems that read:

$$\rho \mathbf{u}_* \cdot \nabla \mathbf{u}^{'} - \mu \boldsymbol{\Delta} \mathbf{u}^{'} = \mathcal{R}([\mathbf{u}_h, p_h]) + \mathbf{v}_{\perp}{}^{'} \text{ in } K, \forall K \tag{2.32}$$

for any $\mathbf{v}_{\perp}{}^{'} \in V_{\perp}^{'}$. The next step to reach the desired expression $\mathbf{u}^{'} = f(\mathbf{u}_h)$ would be to invert the operator in the left side of Equation (2.32). This inversion cannot be accomplished exactly, and an approximation of the inverse operator is needed. This is exactly what we do in the VMS framework, choosing to write

$$\mathbf{u}^{'}|_K \simeq \boldsymbol{\tau}_K \cdot (\mathcal{R}([\mathbf{u}_h, p_h]) + \mathbf{v}_{\perp}{}^{'})|_K \tag{2.33}$$

where the inversion is done on each element K and $\boldsymbol{\tau}_K$, called the *matrix of stabilization parameters* is an approximate of the inverse of the differential operator in (2.32). Several ways exist to appropriately construct matrices $\boldsymbol{\tau}_K$, but going through the details of these constructions is out of the scope of this work. The interested reader could refer to [3] for a construction through approximate Green functions and to [22] if an approximate Fourier analysis is more appealing.

The actual form of matrix $\boldsymbol{\tau}_K$ that we used for the simulations is the following:

$$\boldsymbol{\tau}_K = \left(\frac{c_1 \mu}{h_K^2} + \frac{c_2 \rho \|\mathbf{u}_*\|_{L^\infty(K)}}{h_K} \right)^{-1} \mathbf{I} \tag{2.34}$$

where c_1 and c_2 are constants which only depend on the degree of the FE approximation, and that try to ensure that numerical dissipation is not excessive. The reader will note that the two terms that appear in the definition of $\boldsymbol{\tau}_K$ could be interpreted as the characteristic diffusion time in the element K and the time it takes for a particle to go across an element or advection characteristic time, both weighted by constants c_1 and c_2 respectively. As already mentioned $\|\mathbf{u}_*\|_{L^\infty(K)}$ is taken from the previous iteration when constructing the linear system and we have precisely used the fact that it is known to do the approximation of the inverse operator.

Finally, we need to give a certain form to $\mathbf{v}_{\perp}{}^{'}$ to achieve our goal. This means choosing a space $V^{'}$. For such a purpose we would like to impose the orthogonality condition to (2.33) by testing it with a given $\mathbf{w}^{'\perp}$, which yields:

$$\mathbf{v}^{'\perp} = -\mathbf{P}_{\boldsymbol{\tau}}^{'\perp} \mathcal{R}([\mathbf{u}_h, p_h]) \tag{2.35}$$

where $\mathbf{P}_{\boldsymbol{\tau}}^{'\perp}$ is the projection onto $V^{'\perp}$ associated to the special weighted inner-product $(\cdot,\cdot) = \sum_K \langle \boldsymbol{\tau_K} \cdot, \cdot \rangle_K$. Using $\mathbf{P}_{\boldsymbol{\tau}}^{'} = \mathbf{I} - \mathbf{P}_{\boldsymbol{\tau}}^{'\perp}$, we finally get:

$$\mathbf{u}^{'}|_K = \boldsymbol{\tau}_K \cdot \mathbf{P}_{\boldsymbol{\tau}}^{'} (\mathcal{R}([\mathbf{u}_h, p_h]))|_K \tag{2.36}$$

Two typical choices exist for the subscales space, which are:

- $\mathbf{P}'_{\boldsymbol{\tau}} = \mathbf{I}$, which consists in taking $\mathbf{v}_{\perp}{}' = \mathbf{0}$ and is called the Algebraic Subgrid-Scale formulation (ASGS)
- $\mathbf{P}'_{\boldsymbol{\tau}} = \mathbf{I} - \mathbf{P}_h$, $\mathbf{P}_h$ being the $L^2(\Omega)$-projection onto the FE space, the resulting VMS formulation being called the Orthogonal Subscales Stabilization (OSS)

We would like to note here that since V' has been approximated we have no longer $V = V_h \oplus V'$ but we are looking for a solution on $V_* = V_h \cup V'_{approx}$ which may not be a conforming space of V. This is in fact obvious for example for the ASGS formulation when using linear elements to build space V_h since the residual comprises the gradient of $\mathbf{u}_h$ which is obviously discontinuous across element boundaries. Therefore $\mathbf{u}' \notin H^1(\Omega)^d$ and consequently we could have $V_* \not\subset V$ the space where we are looking for the approximated solution.

In what follows, we have precisely chosen the ASGS formulation to write the final stabilized formulation because it is easier to read since no projection operators appear. The additional ingredientt added by the OSS formulation is that the projection is done using information from the last iteration. With some rewriting of Equation (2.28), notably using $(\mathbf{u}_*\cdot\nabla\mathbf{u}', \mathbf{v}_h) = (\nabla\cdot(\mathbf{u}_* \otimes \mathbf{u}'), \mathbf{v}_h) - (\mathbf{u}'(\nabla \cdot \mathbf{u}_*), \mathbf{v}_h)$ and integrating by parts the first term while neglecting the second one due to incompressibility, we find:

$$\begin{aligned} \rho(\partial_t \mathbf{u}_h, \mathbf{v}_h) + \rho(\partial_t \mathbf{u}', \mathbf{v}_h) + \rho\langle \mathbf{u}_* \cdot \nabla \mathbf{u}_h, \mathbf{v}_h\rangle + \mu(\nabla \mathbf{u}_h, \nabla \mathbf{v}_h) \\ -(p_h, \nabla \cdot \mathbf{v}_h) + (q_h, \nabla \cdot \mathbf{u}_h) - \sum_K \langle \mathbf{u}', \rho \mathbf{u}_* \cdot \nabla \mathbf{v}_h + \mu \boldsymbol{\Delta} \mathbf{v}_h + \nabla q_h \rangle_K \\ + \sum_K \langle \mathbf{u}', \mu \mathbf{n} \cdot \nabla \mathbf{v}_h + q_h \mathbf{n}\rangle_{\partial K} = \rho\langle \mathbf{f}, \mathbf{v}_h\rangle, \quad \forall [\mathbf{v}_h, q_h] \in V_{0,h} \times Q_h \end{aligned} \tag{2.37}$$

We recall now that we already did some hypothesis on $\mathbf{u}'$ and $\mathbf{u}_*$. Using them and replacing (2.36) into (2.37), we end up with a final equation only written in terms of $\mathbf{u}_h$, $\mathbf{v}_h$, p_h and q_h,

$$\begin{aligned} \rho(\partial_t \mathbf{u}_h, \mathbf{v}_h) + \rho\langle \mathbf{u}_h \cdot \nabla \mathbf{u}_h, \mathbf{v}_h\rangle + \mu(\nabla \mathbf{u}_h, \nabla \mathbf{v}_h) - (p_h, \nabla \cdot \mathbf{v}_h) + (q_h, \nabla \cdot \mathbf{u}_h) \\ - \sum_K \langle \boldsymbol{\tau}_K \cdot \mathcal{R}(\mathbf{u}_h, p_h), \rho \mathbf{u}_h \cdot \nabla \mathbf{v}_h + \mu \boldsymbol{\Delta} \mathbf{v}_h + \nabla q_h \rangle_K = \rho\langle \mathbf{f}, \mathbf{v}_h\rangle, \quad \forall [\mathbf{v}_h, q_h] \in V_{0,h} \times Q_h \end{aligned} \tag{2.38}$$

With such an equation, we can do again all that we did on Section 2.2 regarding time integration and non-linearities, the main difference being that now, due to the terms that come from stabilization, block matrix $\mathbf{0}$ in the left-hand-side (LHS) and vector $\mathbf{0}$ in the right-hand-side (RHS) in the system in Equation (2.22) are no longer $\mathbf{0}$, since we have terms combining q_h and p_h.

Therefore the system to solve is the following:

$$\begin{pmatrix} \tilde{\mathbf{K}}_{\mathbf{UV}} & \tilde{\mathbf{K}}_{\mathbf{UQ}} \\ \tilde{\mathbf{K}}_{\mathbf{PV}} & \tilde{\mathbf{K}}_{\mathbf{PQ}} \end{pmatrix} \cdot \begin{pmatrix} \mathbf{U}^{k+1}_{n+1} \\ \mathbf{P}^{k+1}_{n+1} \end{pmatrix} = \begin{pmatrix} \tilde{\mathbf{F}}^{k+1}_{n+1} \\ \tilde{\mathbf{F}}_{\mathbf{q}}{}^{k+1}_{n+1} \end{pmatrix} \tag{2.39}$$

where the ˜ is used to show that the current matrices are not the same as the ones in (2.22) since they contain a lot of new terms that ensure the stability of the numerical method. The interested reader can refer to Table A.1 in Annex A, where the system is fully defined.

Even though we do not want to fill this project with code[8], we will here show the main structure of the algorithm that is used to solve the incompressible N-S equations under the assumptions done. The meaning of the *Hooks* that appear on the code will be understood later, but it corresponds to parts of the main routine where subroutines can and would be added when the new approximate boundary condition methods will be introduced. The only important idea now is that in hook *Hook.PreDirichlet* the rows and columns associated to nodes with strong Dirichlet BCs are eliminated and the proper RHS is added. We want precisely to find alternatives to this way of proceeding when it is not available due to the nature of our mesh, as we will see on next chapter.

Algorithm 1 Stabilized formulations ASGS and OSS to solve the incompressible NSE

```
1: read $\mathbf{U}_{h,0}$ (initial condition)
2: set $\mathbf{P}_{h,0} = 0$
3: for $j = 0, ..., N-1$ do
4:     set $i = 0$
5:     set $\mathbf{U}^0_{h,j+1} = \mathbf{U}_{h,j}$, $\mathbf{P}^0_{h,j+1} = \mathbf{P}_{h,j}$
6:     while not converged do
7:         $i \leftarrow i + 1$
8:         set $\mathbf{U}^i_{*,j+1} = \mathbf{U}^{i-1}_{h,j+1}$
9:         for ielem=1,....,nelem do
10:            elem ← elems(ielem)
11:            load $\mathbf{U}^{i-1}_{h,j+1}|_{elem}$
12:            load $\mathbf{P}^{i-1}_{h,j+1}|_{elem}$
13:            load $\mathbf{U}_{h,j}|_{elem}$
14:            load $\mathbf{f}_{h,j+1}|_{elem}$
15:            compute $\underline{\underline{\tau_{elem}}}$
16:            set $\tilde{\mathbf{K}}_{\mathbf{UV}}|_{elem} = \mathbf{0}, \tilde{\mathbf{K}}_{\mathbf{UQ}}|_{elem} = \mathbf{0}, \tilde{\mathbf{K}}_{\mathbf{PV}}|_{elem} = \mathbf{0}, \tilde{\mathbf{K}}_{\mathbf{PQ}}|_{elem} = \mathbf{0}$
17:            $\tilde{\mathbf{F}}_{\mathbf{u}}|_{elem} = \mathbf{0}, \tilde{\mathbf{F}}_{\mathbf{q}}|_{elem} = \mathbf{0}$
18:            compute $Hook.PreGauss$
19:            if elem ∈ BCelems then
20:                if Neumann BC then
21:                    for igaus=1,...,elem.boundary.gauss do
22:                        elem.bound.igaus ← igaus
23:                        compute $\tilde{\mathbf{F}}_{\mathbf{u},\mathbf{N}}|_{elem.bound.igaus}$
```

[8]It would in fact be useless to do it since the complicated and multiphysics structure of the code will impede a clear reading

```
                        $\tilde{\mathbf{F}}_{\mathbf{u}}|_{elem} = \tilde{\mathbf{F}}_{\mathbf{u}}|_{elem} + \tilde{\mathbf{F}}_{\mathbf{u},\mathbf{N}}|_{elem.bound.igaus}$
                    **end for**
                **end if**
            **end if**
            **for** igaus=1,...,elem.ngauss **do**
                elem.igaus $\leftarrow$ igaus
                **if** OSS case **then**
                    compute $P_h(\mathcal{R}(\mathbf{U}^{i-1}_{h,j+1}|_{elem}, \mathbf{P}^{i-1}_{h,j+1}|_{elem})$
                **end if**
                compute $\tilde{\mathbf{K}}_{\mathbf{UV}}|_{elem.igaus}, \tilde{\mathbf{K}}_{\mathbf{UQ}}|_{elem.igaus}, \tilde{\mathbf{K}}_{\mathbf{PV}}|_{elem.igaus}, \tilde{\mathbf{K}}_{\mathbf{PQ}}|_{elem.igaus}$
                compute $\tilde{\mathbf{F}}_{\mathbf{u}}|_{elem.igaus}, \tilde{\mathbf{F}}_{\mathbf{q}}|_{elem.igaus}$
                set $\tilde{\mathbf{K}}_{\mathbf{UV}}|_{elem} = \tilde{\mathbf{K}}_{\mathbf{UV}}|_{elem} + \tilde{\mathbf{K}}_{\mathbf{UV}}|_{elem.igaus}$
                set $\tilde{\mathbf{K}}_{\mathbf{UQ}}|_{elem} = \tilde{\mathbf{K}}_{\mathbf{UQ}}|_{elem} + \tilde{\mathbf{K}}_{\mathbf{UQ}}|_{elem.igaus}$
                set $\tilde{\mathbf{K}}_{\mathbf{PV}}|_{elem} = \tilde{\mathbf{K}}_{\mathbf{PV}}|_{elem} + \tilde{\mathbf{K}}_{\mathbf{PV}}|_{elem.igaus}$
                set $\tilde{\mathbf{K}}_{\mathbf{PQ}}|_{elem} = \tilde{\mathbf{K}}_{\mathbf{PQ}}|_{elem} + \tilde{\mathbf{K}}_{\mathbf{PQ}}|_{elem.igaus}$
                set $\tilde{\mathbf{F}}_{\mathbf{u}}|_{elem} = \tilde{\mathbf{F}}_{\mathbf{u}}|_{elem} + \tilde{\mathbf{F}}_{\mathbf{u}}|_{elem.igaus}$
                set $\tilde{\mathbf{F}}_{\mathbf{q}}|_{elem} = \tilde{\mathbf{F}}_{\mathbf{q}}|_{elem} + \tilde{\mathbf{F}}_{\mathbf{q}}|_{elem.igaus}$
                compute $Hook.InGauss$
            **end for**
            compute $Hook.PreDirichlet$
            assembly $\tilde{\mathbf{K}}_{\mathbf{UV},\mathbf{UQ},\mathbf{PV},\mathbf{PQ}}|_{elem}$ to $\underline{\underline{\mathbf{K}}}$
            assembly $\tilde{\mathbf{F}}_{\mathbf{u},\mathbf{q}}|_{elem}$ to $\underline{\mathbf{F}}$
        **end for**
        solve $\underline{\underline{\mathbf{K}}} \cdot [\mathbf{U}^{i}_{h,j+1}\mathbf{P}^{i}_{h,j+1}] = \underline{\mathbf{F}}$
        check convergence
    **end while**
    set $\mathbf{U}_{h,j+1} = \mathbf{U}^{i}_{h,j+1}$,
**end for**
```

Until this point a lot of theoretical background on the way we are going to solve the incompressible N-S equations using FE has been given. Even if this is not absolutely necessary to assess the real core of this project, we believe it is important for the reader to understand what was implemented in FEMUSS (Finite Element Method Using Subgrid Scales) before the beginning of this Master Thesis. The overall idea of the work itself will be to add new terms to the matrices that compose (2.39) that will help to weakly impose Dirichlet BCs when the domain of interest evolves in time and a fixed-mesh is used for computing the solution to the problem. The algorithm was given to explain where the newly created code was introduced, always choosing the structure that minimizes both structural change of the code and repetition of code lines.

Chapter 3

Approximate imposition of Dirichlet boundary conditions

In the numerical simulation of physical phenomena many times one is faced with the need of simulating problems where the physical domain evolves in time. Such is the case for the numerical simulation of structural problems involving large deformations, multifluid flow problems, fluid-structure interaction or problems involving a free surface, amongst others. The question that arises is how to tackle with the movement of the domain(s). Even though some meshless techniques exist [23], we will focus here on assessing the problems for mesh-based methods, in our case Finite Element Method (FEM).

The computational meshes that we use to numerically solve large and computationally expensive problems obviously need to deal with the displacement of the physical domain. The most straightforward approach is to set a mesh that adapts to the shape of the domain by using Lagrangian formulations [24] which can deal with large displacements and geometric non-linearities, or Arbitrary Lagrangian-Eulerian (ALE) formulations [3], which will be discussed on Chapter 4. When we deal with anchored solids in both pure solid dynamics or fluid-structure interaction these two approaches can perform well because element distortion or folding can be avoided [25], but in some low viscosity fluid problems displacements and deformations are too large to have a mesh that adapts to it.

The solution seems *a priori* pretty evident: remeshing. This is obviously one possibility; when the mesh is excessively deformed we can compute a new mesh which covers the deformed configuration and project the previous results from the original deformed mesh to a well-suited mesh. The main problem is that meshing is by itself a time and memory consuming labor and it is rarely an integral part of the FE code. Very often external software is used for meshing and this can be a complicated task, mostly when running in parallel environment, which is the most likely case when running vast numerical experiments. In addition, for large deformations this could be necessary at nearly every time step, thus becoming computationally unachievable.

We already said that the goal of this project is to deal with a very different approach: embedded or fixed mesh methods. In fact, the equations we wrote in Chapter 2 already assumed a purely Eulerian point of view, thus fixed meshes in space. In fixed mesh methods the boundary of the physical domain does not need to match with the boundary of the mesh[1]. In this case boundary conditions (BCs) need to be applied in a boundary which is immersed, and the discrete version of the variational problem does only have physical meaning over the part of the mesh which is occupied by the physical domain. The reader will note that another difficulty when dealing with fixed meshes is that time and space integration may need nodal information that we may not necessarily have. To have a better understanding of the problem we are dealing with, let's imagine the solid domain in Figure 3.1 to be, first of all, still and fixed inside a flow domain. Let's imagine that far from what we see we have some imposed BCs that give rise to a flow. We are aware that if the solid is fixed the use of approximate BCs might not required, but, since it is as well a possibility to apply approximately Dirichlet BCs in fixed interfaces and it allows for clarity we will first consider this case.

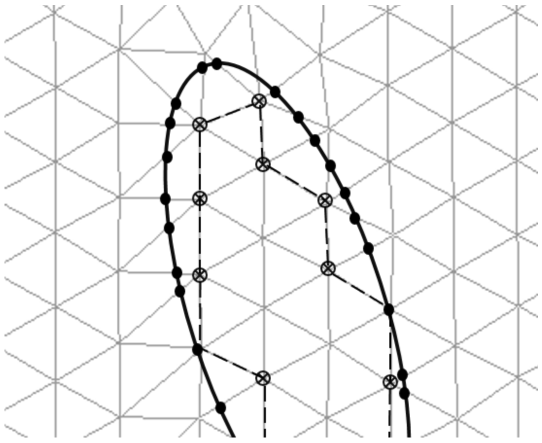


Figure 3.1: Fixed solid domain inside a fluid domain Ω

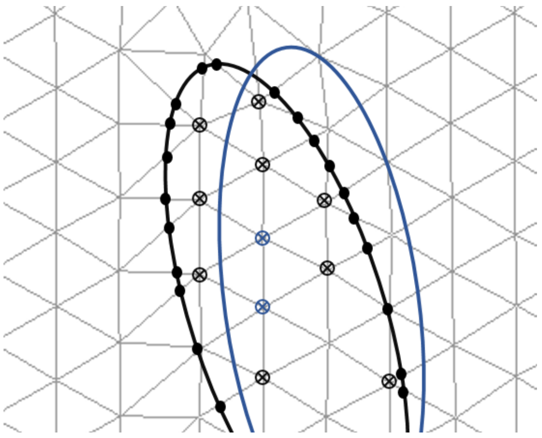


Figure 3.2: Rotating solid domain inside a fluid domain Ω

Imagine that we want to solve at a given time $t = t_n$ the system that we end up having at Section 2.3. The problem stated as it is now, Dirichlet BCs cannot be directly applied by building function $\mathbf{u}_h$ in the way we did in (2.16) because the physical boundary where we impose the velocity condition does not coincide with nodes. This is precisely the reason why this Chapter 3 is called *Approximate imposition of Dirichlet boundary conditions*; we are going to change the system we are solving to weakly impose the BCs. To account for the presence of the boundary -where the condition needs to be imposed in a weak way- we see that the domain where we want to solve the incompressible N-S equations is composed of all the points in the mesh outside of the domain occupied by the solid and the nodes that are in the first layer inside the same, that we have marked in Figure 3.1 with a $\otimes$. We will call this layer from now on layer L_{-1}. The way in which the different methods

[1]We obviously need to have the domain inside the domain covered by the mesh

impose the condition will be seen in the following sections, but they all need to include such nodes. Now, imagine that the solid is a rigid solid turning at a given speed ω and that at $t = t_{n+1}$ the solid is occupying the physical space defined in blue in Figure 3.2. What we see is that, when we want to solve for $[\mathbf{u}_h, p_h]$ at time t_{n+1} we will need information from at least the previous time step t_n from nodes that were not a part of the computed domain on the previous time step. Those are marked in Figure 3.2 as $\otimes$. Therefore we also need some strategy that deals with how to have some information about these nodes in order to calculate time derivatives.

Existing families of fixed mesh methods are precisely often classified depending on those two strategies [26]. Firstly, on the way each method deals with the imposition of BCs, and secondly, on the way they deal with time and sometimes space integration over the fixed mesh. Even though we will comment on the additional ingredients added by the latter at the beginning of Chapter 4, we are focusing here on the former. Let us before getting into the specifics describe the notation used for the problem to be solved. Consider the situation depicted in Figure 3.3. A domain $\Omega \subset \mathbb{R}^d$, d=1,2,3, and with boundary $\Gamma = \partial\Omega$ -red curve in the figure- is covered by a mesh that occupies a domain $\Omega_h = \Omega_{in} \cup \Omega_\Gamma$. $\Omega_{in} \subset \Omega$ is composed by the elements interior to Ω and Ω_Γ by the set of elements of the mesh cut by Γ. Let us also split $\Omega_\Gamma = \Omega_{\Gamma,in} \cup \Omega_{\Gamma,out}$, where part $\Omega_{\Gamma,in} = \Omega \cap \Omega_\Gamma$ and $\Omega_{\Gamma,out} = \Omega_\Gamma \ \Omega_{\Gamma,in}$.

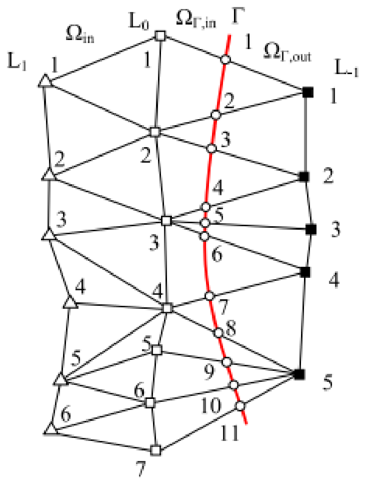


Figure 3.3: Setting of the mesh near the boundary for the class of problems of interest. Taken from [1].

One assumption that we will do in the following is to force the intersection of Γ with the element domains to be piecewise polynomial of the same order of interpolation as the FE interpolation. This will simplify much the calculations since the operations will be performed on a straight line in the reference element if isoparametric elements are used [5], allowing for an easy integration rule. Suppose that we want to solve a boundary value problem for an unknown $\mathbf{u}$ in Ω with the mesh of Ω_h given and boundary conditions $\mathbf{u} = \mathbf{u}_d$. The reader's first reaction would certainly be the following:

- Obtain the nodes of Γ, that we will call from now on $\mathbf{x}_{\Gamma,i}$ and that are the white circles in Figure 3.3, from the intersection with the element edges
- Split the elements in Ω_Γ in order to obtain a grid matching the boundary Γ. This would be possible due to the piecewise polynomial assumption we just did.
- Prescribe the boundary condition on the discrete system $\mathbf{u}_h = \mathbf{u}_d$ in the classical way explained in Section 2.2

We already see that the strategy above leads to a local remeshing. We already discussed the problems related to remeshing from a computational point of view. In addition, the method does not apply for Cartesian meshes and would imply a permanent change in the sparsity of the connectivity matrix. It is actually a *rare* practice to impose BCs this way. In reality, several approaches exist such as the use of penalty terms as in the original immersed boundary method [27], using Lagrange multipliers [28, 6], which may require the use of additional unknowns and the proper choice of a new inf-sup stable space, or the well-known Nitsche's method [29].

Before starting the description of the methods, we wanted to note that since the equations are solved only in Ω the elements cut by Γ will not be triangular elements and we need to set the numerical integration points in an adapted way. This subintegration is shown in Figure 3.4 for the case of 2D linear elements that we discussed before. The numerical integration points (red points) required in each triangle resulting from this splitting were set before the numerical integration points loop (*Hook.PreGauss* in 1) in order to keep the structure of the code. The degrees of freedom (DOFs) of the problem remain the same since the splitting is only for integration purposes.

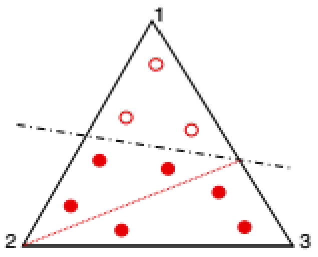


Figure 3.4: Integration splitting of the cut elements. Taken from [2].

Nevertheless, arbitrary cutting of the interface on the background mesh and the subsequent subintegration may cause instabilities. The nature of this instabilities is due to an ill-conditioning of the matrix $\mathbf{K}$ in (2.39) defining the resulting linear system. This ill-conditioning appears when the boundary Γ cuts the elements close to the interior fluid nodes and far away from the *external* solid/other fluid nodes. This causes the contribution of the element integral in the external nodes to be small compared to contributions in other nodes, and instabilities appear in those due to a bad matrix conditioning. Such nodes are depicted in Figure 3.5. The solution to this ill-conditioning was introduced by Burman with what he called a *ghost penalty stabilization* technique [30] for general finite element problems. The idea is to add terms in order for the condition number of the matrix to be upper bounded independently of how the domain boundary intersects the computational mesh. We now extend to the present setting the ideas presented on the cited article by adding the following to (2.38):

$$\begin{aligned} S_{ghost}([\mathbf{v}_h, q_h]; [\mathbf{u}_h, p_h]) = &\sum_{K_{\Omega_{cut}}} \gamma_{1K} \langle \nabla \mathbf{v}_h, \mathbf{P}_h^{\perp}(\nabla \mathbf{u}_h) \rangle_K \\ &+ \sum_{K_{\Omega_{cut}}} \gamma_{2K} \langle \nabla q_h, \mathbf{P}_h^{\perp}(\nabla p_h - \rho \mathbf{f}) \rangle_K \end{aligned} \tag{3.1}$$

where Ω_{cut} is a patch that extends over the elements cut by Γ and the first neighbour layer in the fluid as can be depicted in Figure 3.6 and the sums only extend for $K \in \Omega_{cut}$. The constants γ_{1K} and γ_{2K} were defined as:

$$\gamma_{1K} = c_3\, h^2\, \tau_K^{-1} \tag{3.2a}$$

$$\gamma_{2K} = c_4\, \tau_K \tag{3.2b}$$

where constants c_3 and c_4 are algorithmic constants that we set to $c_3 = c_4 = 1$.

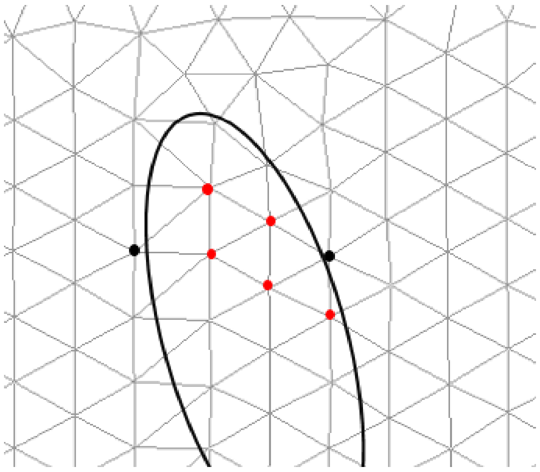


Figure 3.5: Interface cut leading to instabilities in the red nodes

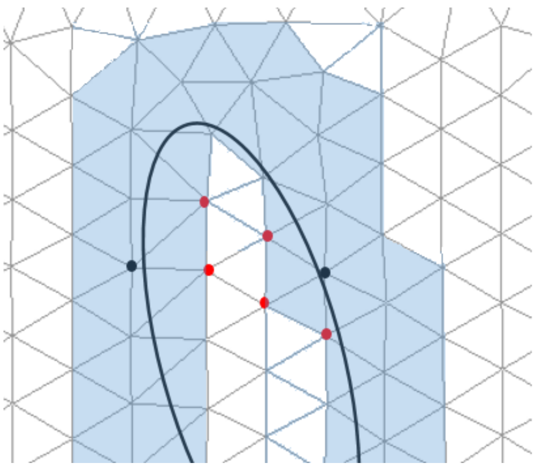


Figure 3.6: Domain Ω_{cut} defined to stabilize the system of equations

For the orthogonal projections involved in these non-consistent stabilization terms [2], we rely on $L^2(\Omega_{cut})$-projections which are evaluated after each non-linear iteration in order to optimize the sparsity pattern of the linear system matrix. These terms were by

[2]The terms are in fact weakly consistent, which means that the error induced converges to zero optimally with the mesh size h.

definition not active in our simulations and we only decided to use them in case some local instabilities as the ones that can be seen in Figures 3.7 - 3.8 appeared. The resulting terms added to the system felt under *Hook.PreGauss* in pseudo-code (1) in order to perform integration using the basic integration points due to integration over all the element area in elements K in Ω_{cut}, in opposition to the other terms that should be integrated only over the physical domain Ω with the subsequent subintegration mentioned above. The calculation of the projections was done at the end of each iteration in a routine not explicitly included in the pseudo-code and using a lumped-mass matrix for simplicity purposes. We will later in this work see that the use of the stabilizing terms used to solve such issues led as well to some problems on the interface that would require further work.

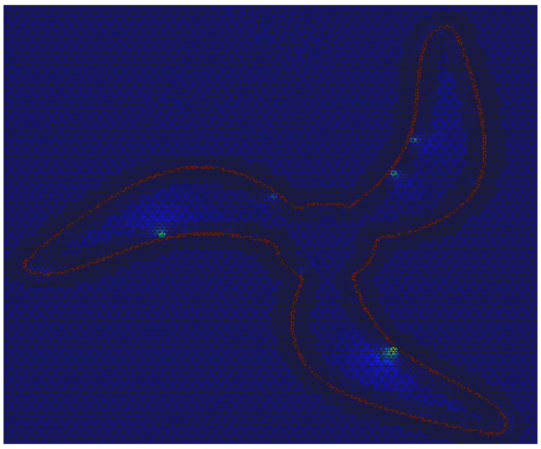


Figure 3.7: Example of instabilities due to interface cut with the background mesh

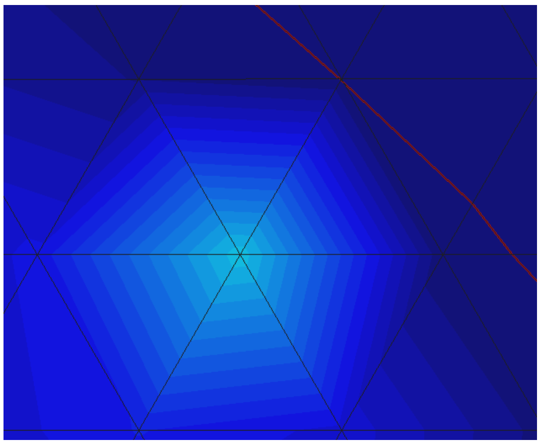


Figure 3.8: Detail of instability from Figure 3.7

Once this subintegration and the *ghost stabilization* mentioned the common ingredients to all the methods presented have been given and we need now to go to the specific methods that allow us to weakly impose Dirichlet BCs at a given time step. Every single method will be presented separately, even though penalty-based methods have been grouped. The numerical and/or physical reasoning behind them will be first explained and then the notion of convergence will be assessed. We decided in this work to do convergence tests towards an exact solution $[\mathbf{u}_{ex}, p_{ex}]$ using only space norms and we will not deal with norms in time. More precisely, we will use the following norm-induced errors in the rest of the chapter:

$$e_{\mathbf{u}, L^2(\Omega)} = \frac{\|\mathbf{u}_{ex} - \mathbf{u}_h\|_{L^2(\Omega)}}{\|\mathbf{u}_{ex}\|_{L^2(\Omega)}} = \frac{\left(\int_\Omega (\mathbf{u}_{ex} - \mathbf{u}_h)^2\right)^{\frac{1}{2}}}{\left(\int_\Omega \mathbf{u}_{ex}^2\right)^{\frac{1}{2}}} \tag{3.3a}$$

$$e_{p, L^2(\Omega)} = \frac{\|p_{ex} - p_h\|_{L^2(\Omega)}}{\|p_{ex}\|_{L^2(\Omega)}} = \frac{\left(\int_\Omega (p_{ex} - p_h)^2\right)^{\frac{1}{2}}}{\left(\int_\Omega p_{ex}^2\right)^{\frac{1}{2}}} \tag{3.3b}$$

$$e_{\overline{\mathbf{u}}, L^2(\Gamma)} = \frac{\|\mathbf{u}_{ex} - \mathbf{u}_h\|_{L^2(\Gamma)}}{\|\mathbf{u}_{ex}\|_{L^2(\Gamma)}} = \frac{\left(\int_\Gamma (\mathbf{u}_{ex} - \mathbf{u}_h)^2\right)^{\frac{1}{2}}}{\left(\int_\Gamma \mathbf{u}_{ex}^2\right)^{\frac{1}{2}}} \tag{3.3c}$$

where $\overline{\mathbf{u}} = \mathbf{u}|_\Gamma$. Therefore, in order to assess the convergence properties an exact solution $[\mathbf{u}_{ex}, p_{ex}]$ is required. Two possible options exist. The first simple option is to solve a given problem of interest with a very fine mesh and to set the solution for this mesh, that we will note by h_*, to be an *analytical* solution to our problem and then compare the solution with solutions obtained from other coarse meshes. This is obviously not the best option since the problem could be converging to a wrong or non stable solution for a stationary problem. The most adapted is to set a simple problem on a simple geometry and to insert on the momentum equation a divergence-free solution verifying some given BCs, which will give us the body forces required for the particular solution to be a *feasible* solution. Then a simulation with such forces has to be performed. Even though we said the interest of weakly imposing Dirichlet BCs is mainly to deal with moving subdomains, since we want to assess the *space* convergence of the methods we will use the stationary N-S equations to test the methods.

It is not obvious how to practically define such a problem since if we, for example, aimed at creating a divergence-free solution on a square with a fixed rotating sphere in the middle, it would be extremely hard to find a solution such that it satisfies that the velocity is what it needs to be at the contact with the sphere. If, otherwise, the domain of a given immersed solid changed over time a time independent solution would be impossible. What we finally did was to take a divergence-free field in a square and impose the velocity given by the field in a immersed boundary directly, without using the velocity boundary condition obtained from a solid motion as we will do later. Still, a *solid* case was used in order to solve and project a level-set function tracking the interface and used to identify cut elements on the background fixed mesh. More detail on the level-set problem will be given on Chapter 4. The problem that we actually used to test the methods is depicted in Figure 3.9, where the divergence-free velocity field $\mathbf{u}$ and the pressure p are given by:

$$u(x, y) = 2x^2 y(x-1)^2(y-1)(2y-1) \tag{3.4a}$$

$$v(x, y) = -2y^2 x(x-1)(y-1)^2(2x-1) \tag{3.4b}$$

$$p(x, y) = \mu \sin(2\pi x) \cos(2\pi y) \tag{3.4c}$$

$u(x, y)$ and $v(x, y)$ being the x- and y-velocity components respectively, where premultiplying the pressure by μ turned out to be necessary if we wanted the existing error estimates to be low. It is straightforward to verify that this field satisfies $\nabla \cdot \mathbf{u} = 0$. The computational domain is the unit square, discretized using simple unstructured meshes of linear triangular elements, the range of element sizes being $0.003 \leq h \leq 0.05$. The problem will be analyzed for various values of the dynamic viscosity μ in order to guarantee robustness of the different methods as we are aware that high Reynolds number flows can lead to some numerical complications regarding the convergence of the non-linearity iterative algorithm.

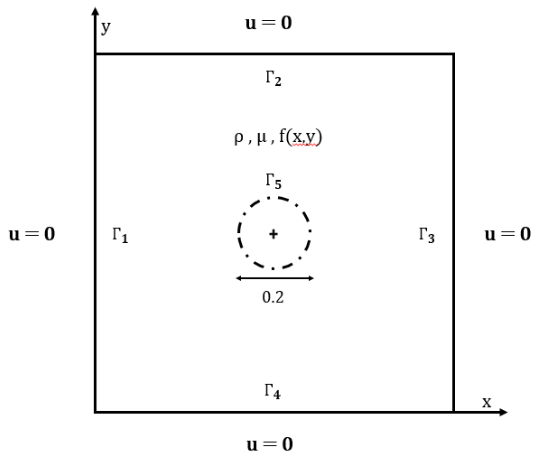


Figure 3.9: Geometry and boundary conditions of the problem used for convergence analysis.

The Dirichlet BCs that we imposed are also depicted in Figure 3.9, and are obviously inherited from the solution itself:

$$\mathbf{u} = \mathbf{0}, \qquad \text{on } \Gamma_{1,2,3,4} \tag{3.5a}$$

$$u(x,y) = 2x^2y(x-1)^2(y-1)(2y-1), \qquad \text{on } (x,y) \in \Gamma_5 \tag{3.5b}$$

$$v(x,y) = -2y^2x(x-1)(y-1)^2(2x-1), \qquad \text{on } (x,y) \in \Gamma_5 \tag{3.5c}$$

where BCs in (3.5a) will be strongly imposed and (3.5b-3.5c) will be weakly imposed.

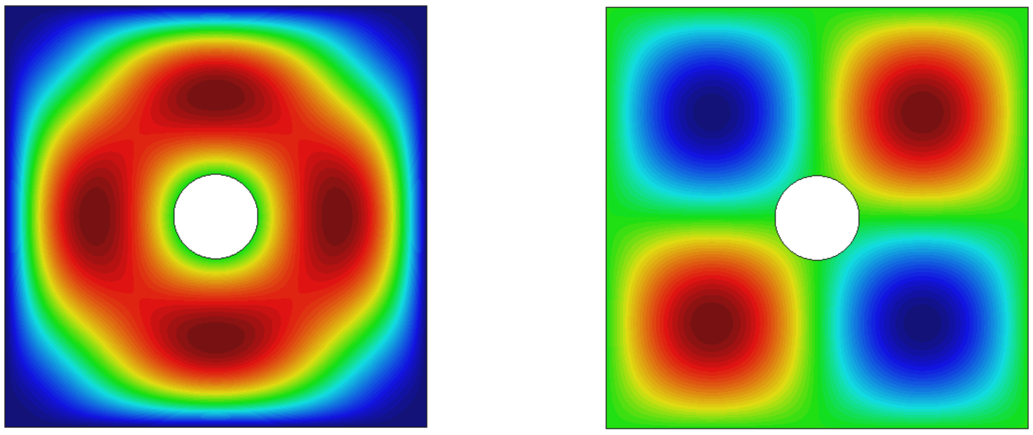

(a) $\|\mathbf{u}_h\|$ (b) p_h

Figure 3.10: Plots of the analytical solution for the problem used for convergence analysis (using strong Dirichlet BC imposition)

We show in Figures 3.10a-3.10b the solution expected, the precise values in pressure depending on the choice of μ. In this case the solution was obtained using strong Dirichlet BCs on all boundaries for a fine mesh of size h=0.003. The optimal convergence rate of the errors expected when the mesh size is reduced using linear triangular elements is two in velocity for the $L^2(\Omega)$-norm and at least one in $L^2(\Omega)$-norm for the pressure field for the incompressible N-S equations [12]. From now on, all the convergence plots given will correspond to the described problem in this section. Since it could also be useful to compare the convergence curves of the methods we will be describing with the convergence curves if Dirichlet BCs were imposed in a classical way in the inner circle, we have plotted these in Figure 3.11, where curves exhibit slopes higher than two for both velocity and pressure.

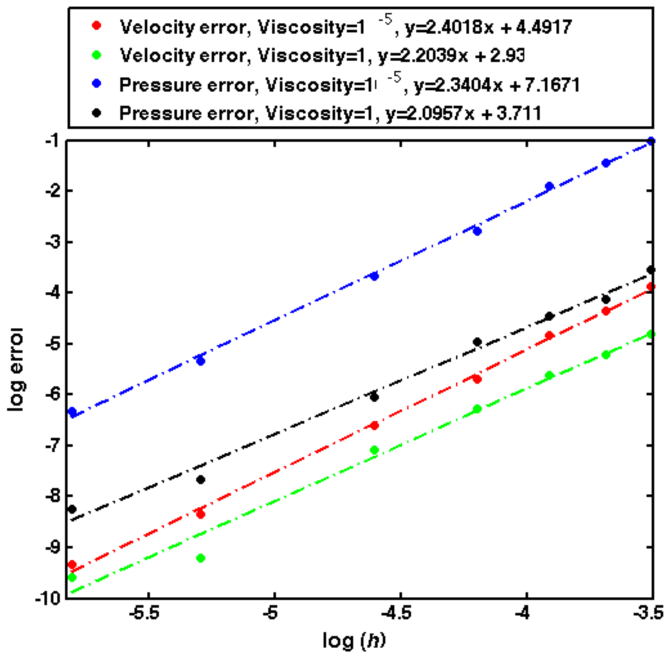


Figure 3.11: Log-log convergence curves (strong Dirichlet BC imposition).

3.1 Penalty-based methods

Penalty-based methods' main idea is, as its name says, to penalize distance of the real solution to the imposed value in the boundary. Even though the nature of this penalization has been interpreted differently, all the methods have a similar form. Throughout this section some of the methods that could lie inside this family will be presented.

Their origin can be found in the *immersed boundary method* [27], that is due to C. Peskin in 1972. The key point of the original immersed boundary method is the introduction of a force in the momentum equation. In principle this force is introduced only in the fluid-solid interface by means of a Dirac-delta function, but in practice this function has to be extended to the nodes surrounding the interface due to the discrete nature of finite element meshes, and the Dirac-delta function is smoothed. The simpler option and the most commonly used [31] is to consider this force as proportional to the deviation of the boundary value of the velocity to the boundary condition, in a spring-like model. This is actually done in a very large class of constrained optimization problems as well. For this case, k being the constant of proportionality, the punctual force added to the N-S equations is

$$\mathbf{f} = \sum_{i}^{P} k\,(\mathbf{u} - \mathbf{u}_d)\,\delta(|\mathbf{x} - \mathbf{x}_{\Gamma,i}|) \tag{3.6}$$

which in weak form adds the term $\langle\, \mathbf{v}_h\,,\, \sum_i^P k(\mathbf{u}_h - \mathbf{u}_d)\,\overline{\delta}(|\mathbf{x} - \mathbf{x}_{\Gamma,i}|)\,\rangle_\Omega$ to the left-hand side (LHS) of the equation. We have herein smoothed the Dirac-delta function due to discrete nature of the FEM. Here k has to be chosen large enough to correctly impose the BC. The problem is that large values of k can render the linear system highly ill-conditioned or diminish the accuracy in the surrounding solution. We will see that this user-defined parameter is a feature of penalty methods. We will study the sensibility of the solution to it, since the variability of the performance of the method with respect to this parameter is said to be one of their main drawbacks. The first group of methods that were implemented in FEMUSS look very similar to the one just described, but the reasoning behind is far more numerical than physical. Two possible formulations arising from different interpretations are going to be associated to the penalty method and we will just call them *penalty method* and *penalty method with boundary tractions.*

3.1.1 Penalty method

Let us imagine that the problem to solve is now the following: find $[\mathbf{u}, p] \in L^2(0, T; V) \times \mathcal{D}'(0, T; Q)$ satisfying

$$\rho\frac{\partial \mathbf{u}}{\partial t} + \rho\mathbf{u}\cdot\nabla\mathbf{u} - \mu\boldsymbol{\Delta}\mathbf{u} + \nabla p = \rho\mathbf{f} \qquad \text{in} \quad \Omega_t, \quad t > 0 \tag{3.7a}$$

$$\nabla\cdot\mathbf{u} = 0 \qquad \text{in} \quad \Omega_t, \quad t > 0 \tag{3.7b}$$

$$\mathbf{n}\cdot[2\mu\nabla^{\mathbf{S}}\mathbf{u} - p\mathbf{I}] = \frac{1}{\epsilon}(\mathbf{u}_d - \mathbf{u}) \qquad \text{on} \quad \Gamma_d, \quad t > 0 \tag{3.7c}$$

where Γ_d is the boundary where Dirichlet boundary conditions were imposed on the original problem and where we have now imposed a Robin-type boundary condition. It has been shown [32] that by applying this same transformation to a Poisson problem with Dirichlet BCs, the solution of the Robin-type BC converges to the solution of the standard Poisson problem with Dirichlet BC when $\epsilon \to 0$. Although no formal proof has been found to show that the solution to Problem (3.7) converges to the solution to Problem (2.2) with ϵ for the incompressible N-S equations, we will numerically show that the method produces good solutions. The idea of the penalty method is to solve this *penalized* problem by FEM instead of the original problem. It is therefore obvious that ϵ needs to converge to 0 at least with the same rate as h converges to 0; i.e $\epsilon = O(h)$. In the most general case and with a stronger background in numerical analysis it is possible to show for the Poisson problem that if $\epsilon = C\,h^\lambda$, an optimal convergence rate of the method can be achieved. We will, as a departure point, choose $\lambda = 1$, and analyse what happens with the convergence rate, hoping that optimal convergence could be ensured for the method. For an interesting approach to this subject, which is far more general than how it is presented here, and to the relation between differential problems and their variational form the reader could refer to Courant's analysis of vibrational problems [33]. The main idea that the penalty method takes from the issues discussed in [33] is that Dirichlet BCs should be seen as a limit case of the natural boundary conditions of the given problem.

The advantage of solving Problem 3.7 is that when the weak form is computed we find ourselves with a new term on $\mathbf{u}$ coming from using the Robin boundary condition (3.7c) after integration by parts, which will approximate the Dirichlet BC when the mesh size h is small enough. Now, the simplest penalty method consists in adding

$$\int_\Gamma \frac{\alpha}{h}(\mathbf{u}_h - \mathbf{u}_d)\cdot \mathbf{v}_h \tag{3.8}$$

to the LHS of the discrete weak form[3], where we have set $\epsilon = \frac{h}{\alpha}$ for the reason already mentioned. In order to ensure consistency (invariance to unit changes) without requiring further characteristic lengths the parameter α has been chosen for every element K as follows:

$$\alpha = s_1\mu + s_2 h \|\mathbf{u}_*\|_{L^\infty(K)} \tag{3.9}$$

which obviously leads to a consistent system and where parameters s_1 and s_2 (or making it short, the penalty parameter α) depend in the geometry of the mesh, mostly in the geometry of the cut elements. It is precisely these mesh-dependent parameters that control the stability of the method [34]. The problem is that the value of the parameters that guarantee stability are not known *a priori*, and they have therefore been introduced as a degree of freedom set by the user in order to both decide the degree of enforcement of the boundary condition in diffusion-dominated problems and convective-dominated problems and to ensure stability. The natural choice of s_1 and s_2 will be to take them as high as possible to be in the *sure side* of stability; nevertheless, a trade-off could exist between accuracy in the boundary and ill-conditioning of the system if they are chosen too high. We have here decided to add the term (3.8) once the stabilization discussed in Section 2.3 has been carried out. This means that in the weak form resulting from (3.7) we have considered $\mathbf{u} = \mathbf{u}_h$ without splitting on the boundary term that results from integrating by parts the divergence of the stress tensor $\underline{\underline{\boldsymbol{\sigma}}}$ and that yields (3.8). The reader will see that from now on in the methods presented we assume that the terms to *add* are added after the stabilization and directly on the equation only depending on $\mathbf{u}_h$.

3.1.2 Penalty method with boundary tractions

Now we will explain what we have called *penalty method with boundary tractions*, which is what people usually describe as the penaly method. It yields a very similar formulation

[3]The term itself can be split on a part on the LHS and a part on the RHS

than the previous one and consists in *directly* imposing the boundary condition in a weak way on the original problem. What we mean by this is that the boundary condition $\mathbf{u} = \mathbf{u}_d$ in Γ can be directly tested by $\mathbf{v}$, or more specifically by the trace of $\mathbf{v}$ on Γ. After multiplying the BC by $a\mathbf{v}$ ($a \in \mathbb{R}$) and integrating, we get in the discrete form:

$$\int_\Gamma a\left(\mathbf{u}_h - \mathbf{u}_d\right) \cdot \mathbf{v}_h \tag{3.10}$$

It turns out to be necessary, here for stability reasons, to have $a = \frac{\alpha}{h}$ [35]. The geometry-dependent stability similarly arises in this case, so we decided to set α with the same form (3.9). However, the problem that arises from adding term (3.10) to the original discrete problem with a boundary Γ not fitting the mesh boundary is that there is no reason why the test function associated to the boundary nodes must vanish. As a consequence the contribution from the boundary integral coming from integration by parts of the constitutive tensor $\underline{\underline{\boldsymbol{\sigma}}}$ must be accounted for, otherwise a poor approximation of the unknowns close to the interface is obtained even if the boundary condition is well approximated [4]. By defining the operator [1]:

$$[\mathbf{n}\cdot\underline{\underline{\boldsymbol{\sigma}}}](\mathbf{u}, p) = -2\mu\mathbf{n} \cdot \nabla^{\mathbf{S}}\mathbf{u} + p\mathbf{n} \tag{3.11}$$

we need to add as well to the LHS term $\langle[\mathbf{n}\cdot\underline{\underline{\boldsymbol{\sigma}}}](\mathbf{u}_h, p_h), \mathbf{v}_h\rangle_\Gamma$. These boundary tractions give its name to the method.

3.1.3 Nitsche's method

Even though the method we just introduced usually performs well, it has one main drawback. Mainly, it is neither symmetric nor skew-symmetric even for the Stokes case. Nitsche's method is precisely an extension of the penalty method with boundary tractions that renders the system antisymmetric or symmetric for symmetric problems, allowing for an important computational cost reduction. The choice depends on how the sign for the incompressibility equation is chosen in the formulation. In our case, to render the system antisymmetric what we do is that we weight the boundary condition $\mathbf{u}_h = \mathbf{u}_d$ by the skew-symmetric counterpart of the operator (3.11). This adds the following new terms to the LHS of the discrete moment equation:

$$-\int_\Gamma (2\mu\mathbf{n} \cdot \nabla^{\mathbf{S}}\mathbf{v}_h + q_h\mathbf{n}) \cdot \mathbf{u}_h \tag{3.12}$$

and the following to the RHS

$$-\int_\Gamma (2\mu\mathbf{n} \cdot \nabla^{\mathbf{S}}\mathbf{v}_h + q_h\mathbf{n}) \cdot \mathbf{u}_d \tag{3.13}$$

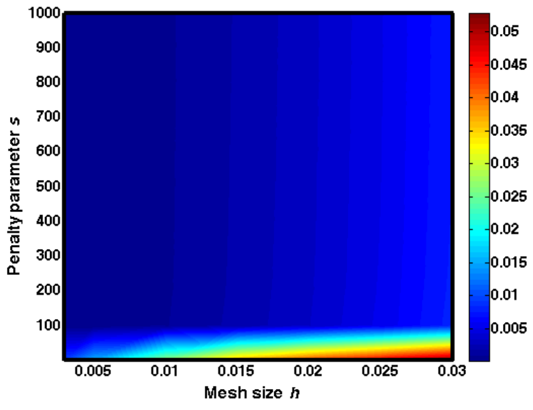
These terms produce contributions that can be added to the matrices in system (2.39).

[4]We would like the poor approximation to be irrelevant, because those tractions are precisely the only practical difference between this method and the simple penalty method

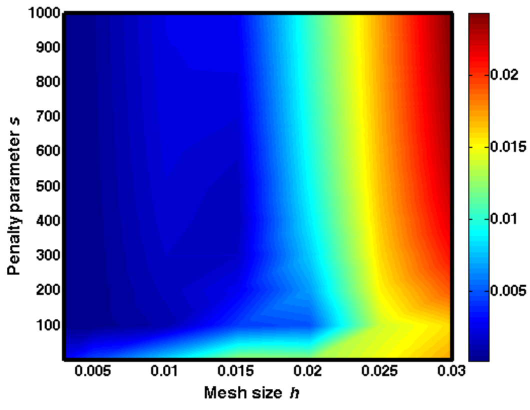
Since all terms imply integration over Γ they were performed previous to the main numerical integration points loop, where only volume integrals are performed in FEMUSS. The routines required where therefore placed under hook *Hook.PreGauss* in the pseudo-code (1). The integration points for the surface integration were set to be the intersection points for simplicity purposes.

3.1.4 Convergence analysis

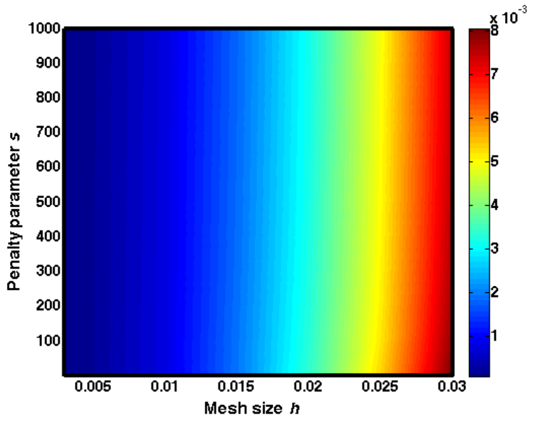
We will here present how the three methods introduced in this section perform on the problem presented at the beginning of this chapter and depicted in Figure 3.9. For the three methods, we will present convergence curves, separately on velocity, pressure and on the boundary error. Let us first present how the errors on velocity and pressure behave as a function of the penalty parameters $s_{1,2}$ introduced in the previous subsections and the mesh size h. The color plots shown in Figure 3.12 correspond to a viscosity $\mu = 1$ and use s_1 as varying parameter since the flow is diffusion-dominated.



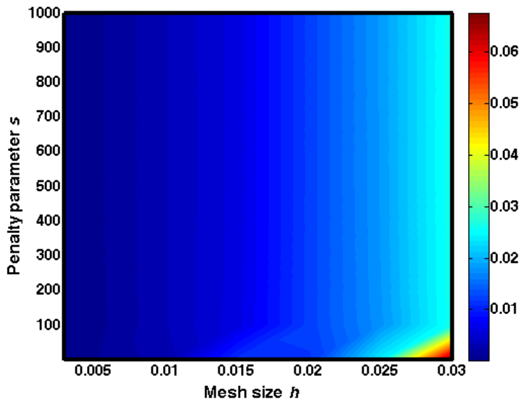
(a) $e_{\mathbf{u},\,L^2(\Omega)}$, Penalty



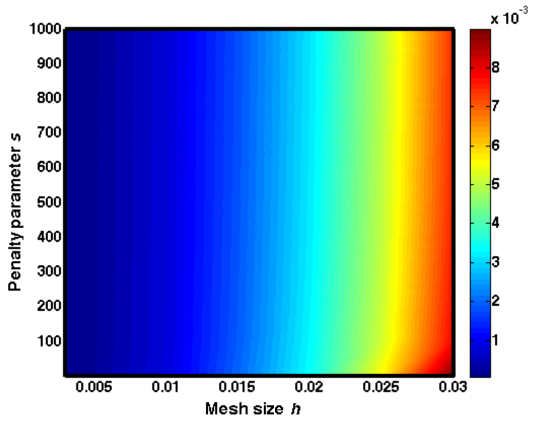
(b) $e_{p,\,L^2(\Omega)}$, Penalty



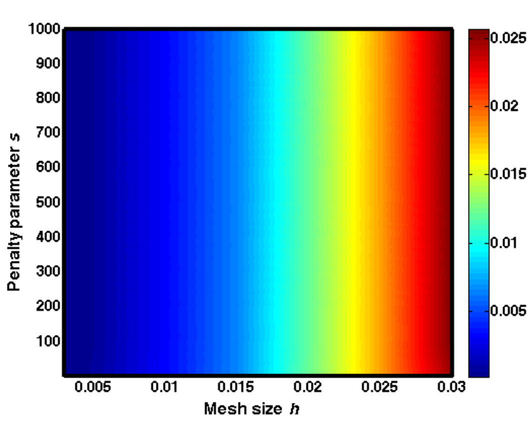
(c) $e_{\mathbf{u},\,L^2(\Omega)}$, Penalty with tractions



(d) $e_{p,\,L^2(\Omega)}$, Penalty with tractions



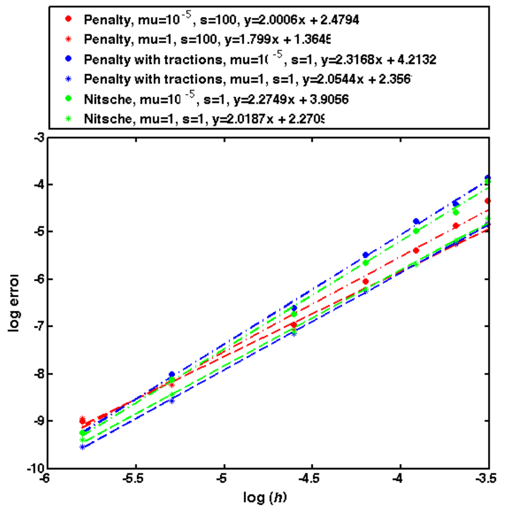
(e) $e_{\mathbf{u},\,L^2(\Omega)}$, Nitsche



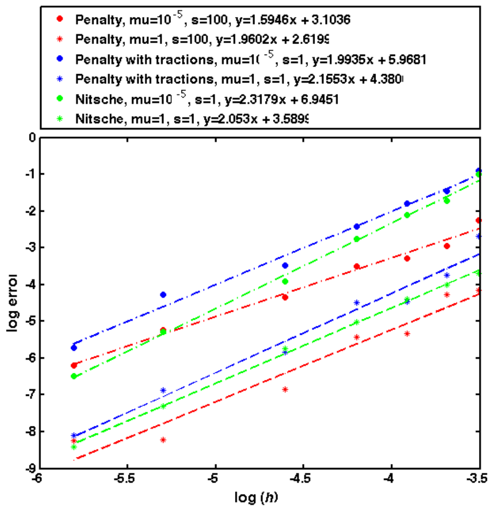
(f) $e_{p,\,L^2(\Omega)}$, Nitsche

Figure 3.12: Plots of the numerical errors (penalty-based methods).

The main conclusions that can be drawn from Figure 3.12 are that choosing s_1=1 may lead to results which show low-order convergence for the simplest penalty method and higher error values in pressure for the penalty method with tractions. As soon as the penalty parameter is high enough (in our case for s_1=100 it is the case) the results do nearly not depend on the penalty parameter and are accurate (Figure 3.13). Nevertheless, we are dealing with a pretty smooth analytical solution (see Figure 3.11 for the convergence orders of the same case with strong imposition of Dirichlet BC) which can help us make the numerical performance more s-independant. We also see that Nitsche's method is even more robust since for s_1=1 the results on pressure have the same absolute error than for higher values of the penalty parameter.



(a) $e_{\mathbf{u},\,L^2(\Omega)}$

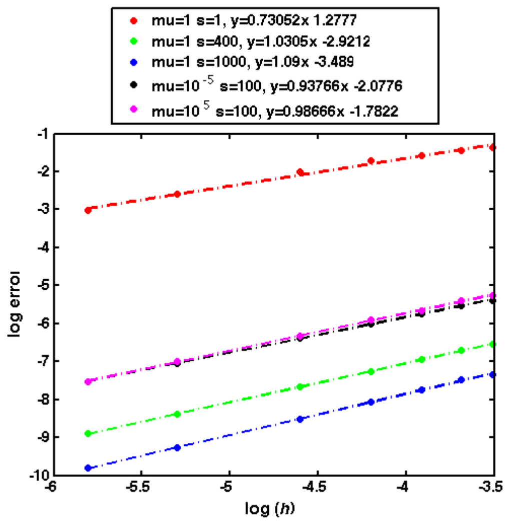


(b) $e_{p,\,L^2(\Omega)}$

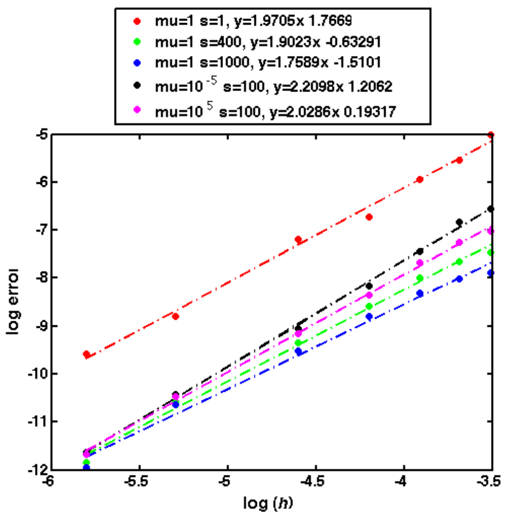
Figure 3.13: Log-log convergence curves with linear fits (penalty-based methods).

We also show the log-log convergence plots of the penalty-based methods. The curve using a value s_1=100 for the simplest penalty method is used due to the very bad performance of the same with s_1=1. For the other two methods the same curves are depicted in Figure 3.13 but using s_1=1 as parameter. We can see that the methods are performing as expected and even better than expected. We get a slope considerably higher than 1 for the pressure error, approaching 2. We were first extremely surprised to get slopes even higher than two for the velocity error but those can be assigned, as said, to the smoothness of the solution itself. The slopes achieved are of the same order as the ones depicted in Figure 3.11 for strong Dirichlet BC on the inner circle, which confirms the validity of those to effectively solve the N-S incompressible equations in non-matcing grids. This similarity is even stronger for Nitsche's method for which even the absolute value of the errors is the same. Additionally, the penalty method without boundary tractions, where an unreal module and direction of the tractions in the boundary are imposed due to (3.7c), performed only slightly worse than the other two penalty-based methods. Nevertheless, the linear fits are not perfect for this method, and the scatter of the points shows some convexity, which would imply lower convergence orders.

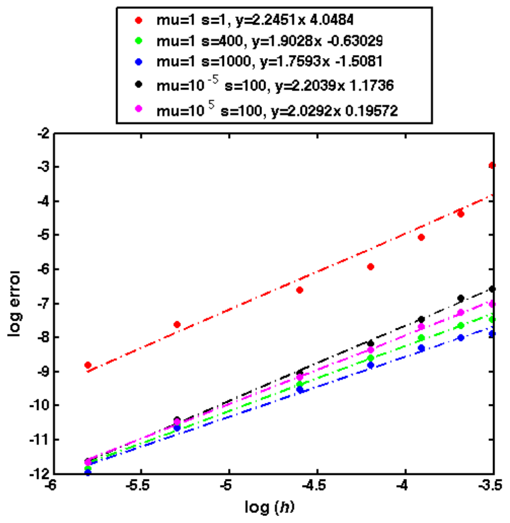
Even though the slopes of the convergence curves for all the methods are similar, the convergence of the error on the boundary $e_{\overline{\mathbf{u}},\,L^2(\Gamma)}$ could present some considerable differences since it is precisely on how those impose the condition that they differ. So let us now show the same convergence plots shown above for the boundary error for the three methods and with various penalty parameter (s_1 or s_2 as a function of the nature of the case) and viscosity μ values.



(a) $e_{\overline{\mathbf{u}},\,L^2(\Gamma)}$, Penalty



(b) $e_{\overline{\mathbf{u}},\,L^2(\Gamma)}$, Penalty with tractions



(c) $e_{\overline{\mathbf{u}},\,L^2(\Gamma)}$, Nitsche

Figure 3.14: Log-log convergence curves of the boundary error (penalty-based methods).

We see that the slopes of the curves for the simple penalty method differ considerably from the slopes achieved by the penalty method with tractions and Nitsche's method. Nevertheless, a convergence order of approximately 1 on the former is still satisfying due to the boundary nature of the integrals. Another interesting aspect is the dependence of both the absolute errors and the slopes of the curves as a function of the penalty parameter. We were surprised to see that higher values of $s_{1,2}$ yielded lower convergence orders, with the exception of the simple penalty method.

The main difference between the methods is the distinct nature of the approximation done with the penalty method. This method is not a consistent method since $[\mathbf{u}, p]$, solution of the original problem with Dirichlet BCs, does not satisfy the weak formulation used to derive the discrete system of equations, so we are not surprised that the convergence order on the boundary differs from the other two penalty-based methods. In addition, the method showed a strong under-performance of the convergence order with low viscosities, which is another sign of such feature. The difference between the methods was predictable since with the simplest penalty method the convergence of the discrete solution to the *approximated analytical solution* and the convergence of the approximated analytical solution to the real analytical solution need to be coupled. Further studying the different approach used with the penalty method should be undertaken to fully understand the method, which, at least to our knowledge, has not been used for the resolution of the N-S equations with approximate Dirichlet BCs. Nevertheless, the practical interest of the method is relative, since adding the corresponding tractions to the formulation is, from a computational point of view, not complicated or time-consuming.

3.2 A *linked* Lagrange multiplier symmetric method

The method to be exposed in this section is a method theoretically developed and implemented in another code at CIMNE in 2011, and everything concerning the formulation can be consulted in Reference [34], which will here be used as a source.

The main goal of the development of this method is to find a solution to the drawbacks of the methods presented in the last section. Having a user-defined parameter whose value determines the stability of the method is, as we said, the main drawback, since the choice of this stabilization parameter is not straightforward: if the parameter is not large enough, the problem becomes unstable; if it is too large, the resulting system of equations becomes ill-conditioned. This drawback could be addressed by doing a stability analysis of the formulation using the inverse estimates in order to define the minimum value for the stabilization parameter. However the inverse estimates still have some non-dimensional constants that need to be defined and those are hard to find for non-symmetric problems.

Therefore, the method we would like to construct for weakly imposing Dirichlet boundary conditions has the following desired properties:

- No additional degrees of freedom should be introduced in the final system of equations[5]. Traditional Lagrange multiplier techniques make the system of equations become larger, and the space for the Lagrange multipliers has to be carefully chosen so that the final formulation remains stable.
- No user-defined penalty parameters upon which the stability of the method relies should be part of the formulation. This will help us avoid as well ill-conditioning of the matrix defining the resulting system of equations.
- A symmetric system should be constructed for symmetric problems, but being able to deal with non-symmetric problems
- Rate of convergence should approach the optimal

The last of this requirements will not be tackled theoretically, but will be tested numerically as it has been done in the previous section for penalty-based methods. Let us therefore see how the other requirements are achieved.

The main idea of what we have called the *linked* Lagrange multiplier method is to introduce a third field representing an additional element-wise discontinuous flux field. However, this is only required in the elements which are cut by the immersed boundary, and because it is discontinuous across interelement boundaries, it can be condensed prior to solving the resulting system of equations. Let us consider the next three-field

[5]This is mainly a requirement imposed by the existing code itself. Everything added should be able to minimize the number of changes to make in the general structure of the code

incompressible Navier-Stokes problem, consisting of finding $\mathbf{u} : \Omega \to \mathbb{R}^d$, $p : \Omega \to \mathbb{R}$ and $\underline{\underline{\boldsymbol{\sigma}}} : \Omega \to \mathbb{R}^{d\times d}$ such that:

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u} - \nu\boldsymbol{\Delta}\mathbf{u} + \nabla p = \mathbf{f} \qquad \text{in } \Omega, \qquad t > 0 \tag{3.14a}$$

$$\nabla\cdot\mathbf{u} = 0 \qquad \text{in } \Omega, \qquad t > 0 \tag{3.14b}$$

$$\frac{1}{\nu}\underline{\underline{\boldsymbol{\sigma}}} = \nabla\mathbf{u}, \qquad \text{in } \Omega_{\Gamma,in}, \quad t > 0 \tag{3.14c}$$

$$\mathbf{u} = \mathbf{u}_d \qquad \text{on } \Gamma_d, \qquad t > 0 \tag{3.14d}$$

$$\mathbf{u} = \mathbf{u}_0 \qquad \text{in } \Omega, \qquad t = 0 \tag{3.14e}$$

Now, to construct a weak formulation for the problem, let's multiply the first equation by $\mathbf{v}$, (3.14b) by q, (3.14c) by $\frac{\nabla\mathbf{v}}{n}$ and by $\frac{\underline{\underline{\boldsymbol{\tau}}}}{n}$, and (3.14d) by $\underline{\underline{\boldsymbol{\tau}}}\cdot\mathbf{n}$ and by $\frac{a}{2}\mathbf{v}$, where $a \in \mathbb{R}$ will be defined later. To obtain a symmetric method, let us also test the normal component of the boundary condition (3.14d) by q_h. We will consider the finite element spaces $V_h \subset H^1(\Omega_h)^d$, $Q_h \subset L^2(\Omega_h)$ and $S_h \subset L^2(\Omega_h)^{d\times d}$, where functions in S_h will be considered piecewise discontinuous in the elements cut by Γ and zero elsewhere.

We get, after grouping and introducing the ASGS stabilization terms described in Section 2.3, the following discrete problem:

$$\begin{aligned}
&(\delta_t\mathbf{u}_h,\mathbf{v}_h) + \langle\mathbf{u}_h\cdot\nabla\mathbf{u}_h,\mathbf{v}_h\rangle + \nu(\nabla\mathbf{u}_h,\nabla\mathbf{v}_h) - (p_h,\nabla\cdot\mathbf{v}_h) - \langle\underline{\underline{\boldsymbol{\sigma}_h}}\cdot\mathbf{n},\mathbf{v}_h\rangle_\Gamma \\
&+ \langle\mathbf{n}\cdot\mathbf{v}_h,p_h\rangle_\Gamma + \frac{1}{n}(\nabla\mathbf{v}_h,\underline{\underline{\boldsymbol{\sigma}_h}})_{\Omega_{\Gamma,in}} - \frac{\nu}{n}(\nabla\mathbf{v}_h,\nabla\mathbf{u}_h)_{\Omega_{\Gamma,in}} + \langle\frac{a}{2}\mathbf{v}_h,\mathbf{u}_h\rangle_\Gamma + \\
&\sum_K \tau_K(\nu\boldsymbol{\Delta}\mathbf{v}_h + \mathbf{u}_h\cdot\nabla\mathbf{v}_h, \delta_t\mathbf{u}_h - \nu\boldsymbol{\Delta}\mathbf{u}_h + \mathbf{u}_h\cdot\nabla\mathbf{u}_h + \nabla p_h)_K = \langle\mathbf{f},\mathbf{v}_h\rangle \\
&+ \sum_K \tau_K(\nu\boldsymbol{\Delta}\mathbf{v}_h + \mathbf{u}_h\cdot\nabla\mathbf{v}_h,\mathbf{f})_K + \langle\frac{a}{2}\mathbf{v}_h,\mathbf{u}_d\rangle_\Gamma, \qquad \forall\,\mathbf{v}_h\in V_h
\end{aligned} \tag{3.15}$$

$$\begin{aligned}
&-(q_h,\nabla\cdot\mathbf{u}_h) - \sum_K \tau_K(\nabla q_h, \delta_t\mathbf{u}_h - \nu\boldsymbol{\Delta}\mathbf{u}_h + \mathbf{u}_h\cdot\nabla\mathbf{u}_h + \nabla p_h)_K + \langle\mathbf{n}\cdot\mathbf{u}_h,q_h\rangle_\Gamma \\
&= -\sum_K \tau_K(\nabla q_h,\mathbf{f})_K,\mathbf{v}_h\rangle + \langle\mathbf{n}\cdot\mathbf{u}_d,q_h\rangle_\Gamma, \qquad \forall\, q_h\in Q_h
\end{aligned} \tag{3.16}$$

$$-\frac{1}{n\nu}(\underline{\underline{\boldsymbol{\tau}}},\underline{\underline{\boldsymbol{\sigma}}})_{\Omega_{\Gamma,in}} + \frac{1}{n}(\underline{\underline{\boldsymbol{\tau}}},\nabla\mathbf{u}_h)_{\Omega_{\Gamma,in}} - \langle\underline{\underline{\boldsymbol{\tau}_h}}\cdot\mathbf{n},\mathbf{u}_h\rangle_\Gamma = \langle\underline{\underline{\boldsymbol{\tau}_h}}\cdot\mathbf{n},\mathbf{u}_d\rangle_\Gamma, \quad \forall\,\underline{\underline{\boldsymbol{\tau}}}_h\in S_h \tag{3.17}$$

where we already introduced $\delta_t\mathbf{u}_h$ the time discretization of our equation and where the VMS-stabilization splitting was only done for variable $\mathbf{u}$ in the momentum and incompressibility equations. For the moment the desired property of having no additional variables is not achieved and the method resembles a *traditional* Lagrange multiplier technique, so let us take a look at the obtained system to see how to condense it in order to have a system only written as a function of the initial variables $\mathbf{U}_{n+1}^{k+1}$ and $\mathbf{P}_{n+1}^{k+1}$.

The system reads as follows:

$$\begin{pmatrix} \mathbf{K_{uv}} & \mathbf{K_{pv}} & \mathbf{K_{\sigma v}} \\ \mathbf{K_{uq}} & \mathbf{K_{pq}} & \mathbf{0} \\ \mathbf{K_{u\tau}} & \mathbf{0} & \mathbf{K_{\sigma\tau}} \end{pmatrix} \cdot \begin{pmatrix} \mathbf{U}_{n+1}^{k+1} \\ \mathbf{P}_{n+1}^{k+1} \\ \boldsymbol{\sigma}_{n+1}^{k+1} \end{pmatrix} = \begin{pmatrix} \mathbf{F_v}{}_{n+1}^{k+1} \\ \mathbf{F_q}{}_{n+1}^{k+1} \\ \mathbf{F}_{\boldsymbol{\tau}}{}_{n+1}^{k+1} \end{pmatrix} \tag{3.18}$$

where the two **0** blocks result from the absence of crossed terms in pressure-pseudostress.

Now, in this system submatrices $\mathbf{K_{u\tau}}$, $\mathbf{K_{\sigma v}}$ and $\mathbf{K_{\sigma\tau}}$ are mostly sparse since the only non-zero rows and columns will be the corresponding to nodes in elements in Ω_Γ. This could be of interest for solving the system easily. In fact, the reason to choose S_h as a space of piecewise discontinuous functions in the elements cut by Γ and zero elsewhere is that it allows us to very easily deal with elements independently and condense in an elementary level for all elements K cut by Γ. This is practically very convenient since it allows for the elemental operations in FEMUSS described in Section 2.3 to remain mainly unchanged and solve a system with the initial degrees of freedom.

Since the integrals defining the third block of equations in (3.18) span only over each element due to the discontinuous nature of the tensor field, we can write the following on the element level:

$$\boldsymbol{\sigma}_{n+1}^{k+1} = \mathbf{K_{\sigma\tau}}^{-1} \cdot \left(\mathbf{F}_{\boldsymbol{\tau}}{}_{n+1}^{k+1} - \mathbf{K_{u\tau}} \cdot \mathbf{U}_{n+1}^{k+1} \right) \tag{3.19}$$

The matrix in system 3.18 can be rewritten as:

$$\begin{pmatrix} \mathbf{K_{uv}} - \mathbf{K_{\sigma v}}\mathbf{K_{\sigma\tau}^{-1}}\mathbf{K_{u\tau}} & \mathbf{K_{pv}} \\ \mathbf{K_{uq}} & \mathbf{K_{pq}} \end{pmatrix} \cdot \begin{pmatrix} \mathbf{U}_{n+1}^{k+1} \\ \mathbf{P}_{n+1}^{k+1} \end{pmatrix} = \begin{pmatrix} \mathbf{F_v}{}_{n+1}^{k+1} - \mathbf{K_{\sigma v}}\mathbf{K_{\sigma\tau}^{-1}}\mathbf{F}_{\boldsymbol{\tau}}{}_{n+1}^{k+1} \\ \mathbf{F_q}{}_{n+1}^{k+1} \end{pmatrix} \tag{3.20}$$

where only the arrays of interest $\mathbf{U}_{n+1}^{k+1}$ and $\mathbf{P}_{n+1}^{k+1}$ are kept as unknowns and everything can be assembled element-wise. The actual implementation of this method was done by adding the computation of the boundary terms in hook *Hook.PreGauss* and the volume terms in hook *Hook.InGaussElmats* in pseudo-code 1. The new terms were stored in independent arrays and later condensed as in (3.20) in hook *Hook.PreDirichlet*.

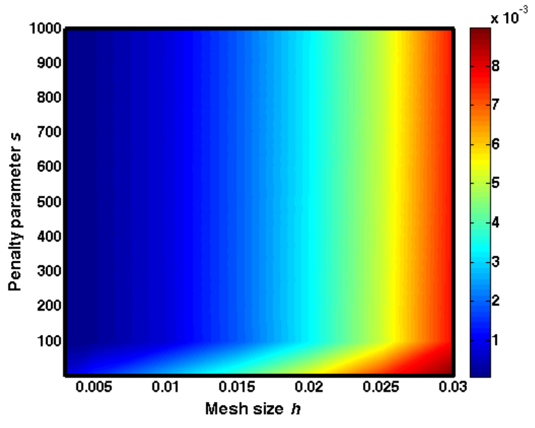
We still have not given any information on the shape of the test functions in S_h except from the fact that they are element-wise discontinuous, which allowed us to perform the condensation in (3.20) element-wise. It can be shown [34] that as soon as the interpolation order for the tensor field $\underline{\underline{\boldsymbol{\sigma}}}$ is at least one order lower than the order for the pressure and velocity fields, the formulation is stable if the following conditions are met:

$$n > 1 \tag{3.21a}$$

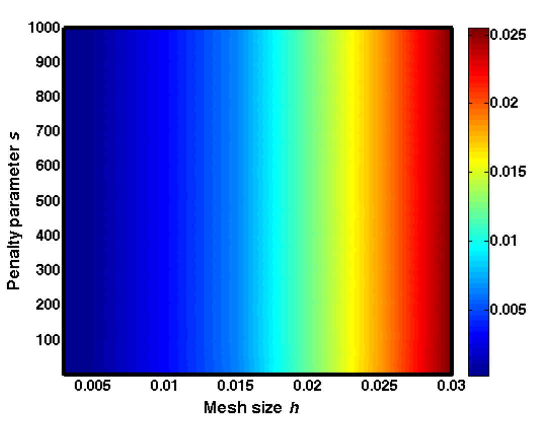
$$a \geq max(0, \mathbf{n} \cdot \mathbf{u}^k_{h,n+1}) \tag{3.21b}$$

independently of the the geometry of the cut mesh. Having an *a priori* estimate -since $\mathbf{u}^k_{h,n+1}$ is known from the previous iteration k- of the minimum value of the parameters involved in the formulation is clearly the main attribute of the *linked* Lagrange multiplier method in comparison to the penalty-based methods previously presented. However, ensuring stability does not necessarily ensure a proper imposition of the BC, and we will therefore also include two constants s_1 and s_2 that will be used to decide the strength of the BC imposition [6]. Therefore we will have $n = 2 \cdot s_1$ and $a = s_2 \cdot \|\mathbf{u}^k_{h,n+1}\|_{L_\Omega(K)}$, where the normal component has been replaced by the norm since stability is ensured due to positivity of the norm and it has been taken for each element K.

In the particular case of the code implemented, only constant piecewise discontinuous interpolation was considered. Therefore the *linked* Lagrange multiplier symmetric method being presented could only ensure stability when using linear elements in any dimension on FEMUSS. The reason for such an implementation was simplicity, since due to the double contraction nature of the terms in $\mathbf{K}_{\boldsymbol{\sigma}\boldsymbol{\tau}}$ and to this constant interpolation it makes the matrix diagonal. Computing the inverse required for condensation is therefore trivial. Nevertheless the changes to produce on the code in order to extend the method to higher-order interpolations is minimal.



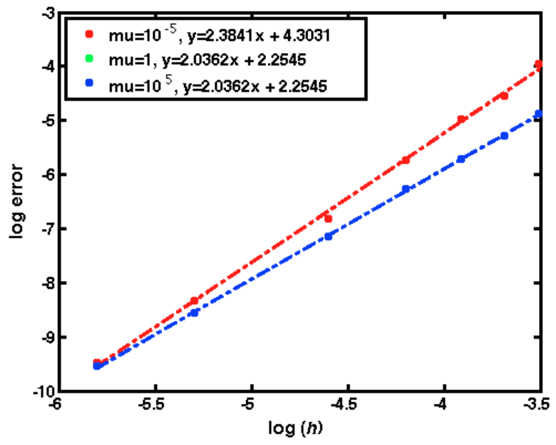
(a) $e_{\mathbf{u},\,L^2(\Omega)}$, *Linked* Lagrange



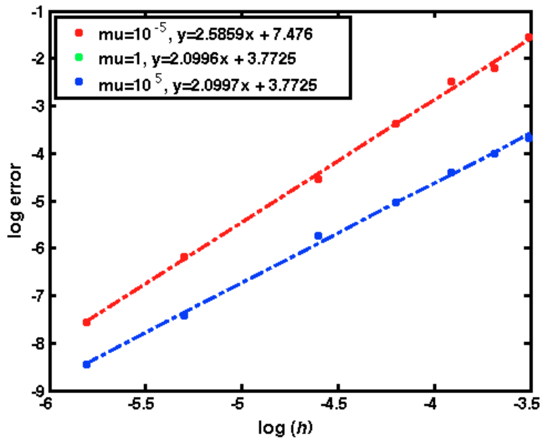
(b) $e_{p,\,L^2(\Omega)}$, *Linked* Lagrange

Figure 3.15: Plots of the numerical errors (*Linked* Lagrange multiplier method).

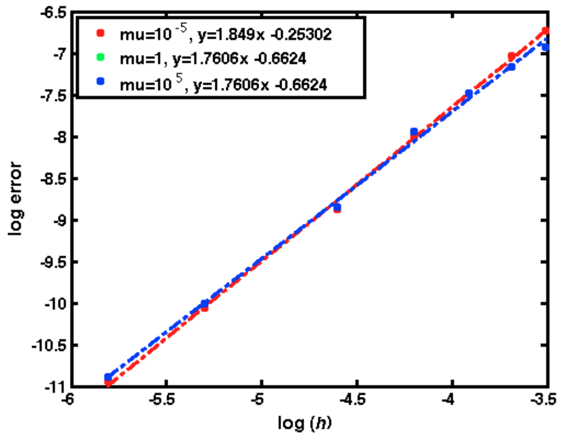
[6] In analogy with what has been done for the penalty-based methods, s_1 will multiply constant n or the diffusive term and s_2 the convective term (or a).



(a) $e_{\mathbf{u},\,L^2(\Omega)}$, *Linked* Lagrange



(b) $e_{p,\,L^2(\Omega)}$, *Linked* Lagrange



(c) $e_{\overline{\mathbf{u}},\,L^2(\Gamma)}$, *Linked* Lagrange

Figure 3.16: Log-log convergence curves with linear fits (*Linked* Lagrange multiplier method).

Having already presented the main idea of the method we are now able to show the results of te numerical tests that show convergence of the method and the influence of parameters n or a in the formulation, as well as its behaviour for various viscosities μ. For the color plots shown in Figure 3.15 s_1 is the varying parameter and a was hold constant as given above, since, for $\mu = 1$, what we want to compare is the effect of a user-set parameter on the diffusive part of the equation. The same plots than those explained for the penalty-based methods are presented on Figures 3.15 and 3.16, the log-log convergence plots given for s_1=100 or s_2=100 depending on the values of the viscosity μ. The method proves to be extremely robust and extremely invariant to a large range of viscosities since the curves for viscosities higher than 10^{-3} completely overlapped. Due to its robustness and to the stability guarantees, we decided to use this method in what follows as reference method to weakly impose Dirichlet BCs and figures showing results of simulations with embedded boundaries will be the result of using this method. Nevertheless, some results on the penalty-based methods could be given since, after all, we see that the absolute errors of those are nearly equivalent to the ones sown in this subsection.

3.3 Using external degrees of freedom

For the moment the methods we have presented are based on adding weak terms to the elemental formulation of our problem. Nevertheless other much more intuitive approaches exist. Another possibility in order to avoid the ill-conditioning due to penalty terms and the need of adding new degrees of freedom to the system of equations (which appears if pure Lagrange multipliers are used) is to use currently existing degrees of freedom in order to enforce BCs. What we mean by this is to directly replace momentum equations that result from testing with functions centered in the nodes adjacent to the fluid-solid interface -those in L_{-1}- with the equations that enforce BCs by minimizing the boundary error. Let's first of all indicate that no change on the matrices of the system to solve will occur on the equations due to testing with test functions on the component of the pressure in (2.15). Since the Dirichlet BC is a condition on $\mathbf{u}$ we will here only refer to these degrees of freedom (DOFs), but the assembly of these DOFs to the whole system is straightforward.

There are several methods which use pre-existing degrees of freedom to enforce BCs. We will focus herein in the method described in detail in [36] to prescribe Dirichlet BC on a generic immersed boundary. For the notation used we ask the reader to refer to previous Figure 3.3. Let us suppose that the solution $\mathbf{u}_h$ can be written in the following way:

$$\mathbf{u}_h = \left(\sum_i^{n_{in}} \phi_i u_i^x + \sum_j^{n_{out}} \phi_j u_j, \sum_i^{n_{in}} \phi_i u_i^y + \sum_j^{n_{out}} \phi_j u_j^y, \sum_i^{n_{in}} \phi_i u_i^z + \sum_j^{n_{out}} \phi_j u_j^z \right) \tag{3.22a}$$

$$= (\boldsymbol{\phi}_{in}^x \cdot \mathbf{U}_{in}^x + \boldsymbol{\phi}_{out}^x \cdot \mathbf{U}_{out}^x, \boldsymbol{\phi}_{in}^y \cdot \mathbf{U}_{in}^y + \boldsymbol{\phi}_{out}^y \cdot \mathbf{U}_{out}^y, \boldsymbol{\phi}_{in}^z \cdot \mathbf{U}_{in}^z + \boldsymbol{\phi}_{out}^z \cdot \mathbf{U}_{out}^z) \tag{3.22b}$$

where ϕ_i are the standard interpolation functions, n_{in} is the number of nodes in Ω_{in} and n_{out} the number of nodes in layer L_{-1}. We have here only split the solution in a sum of the terms associated to inner nodes and the sum associated to the nodes in layer L_{-1}. Suppose as well that the system of equations obtained when testing the weak discrete formulation by all the velocity FE test functions centered in the inner nodes n_{in} yields:

$$\mathbf{K}_{in,in}\mathbf{U}_{in} + \mathbf{K}_{in,out}\mathbf{U}_{out} = \mathbf{F}_{in} \tag{3.23}$$

where vectors $\mathbf{U}_{in}$ and $\mathbf{U}_{out}$ are made of the x, y and z components defined in (3.22). This system is undetermined and therefore needs to be completed with some equations imposing the BCs, the objective being to compute $\mathbf{U}_{out}$. As $\mathbf{u}_h$ needs to be equal to $\mathbf{u}_d$ in Γ, let us minimize the following functional with respect to $\mathbf{U}_{out}$

$$J(\mathbf{U}_{in}, \mathbf{U}_{out}) = \int_\Gamma (\mathbf{u}_h(\mathbf{x}) - \mathbf{u}_d(\mathbf{x}))^2 \, dS \tag{3.24a}$$

$$= \int_\Gamma \sum_{k=x,y,z} \left(\boldsymbol{\phi}_{in}^k(\mathbf{x}) \cdot \mathbf{U}_{in}^k + \boldsymbol{\phi}_{out}^k(\mathbf{x}) \cdot \mathbf{U}_{out}^k - u_d^k(\mathbf{x}) \right)^2 dS \tag{3.24b}$$

which yields

$$\frac{\partial J(\mathbf{U}_{in},\mathbf{U}_{out})}{\partial \mathbf{U}^k_{out}} = 0 \rightarrow \mathbf{N}^k_\Gamma\,\mathbf{U}^k_{in} + \mathbf{M}^k_\Gamma\,\mathbf{U}^k_{out} = \mathbf{F}^k_\Gamma, \quad \text{for } k{=}x{,}y \text{ or } z \tag{3.25}$$

where

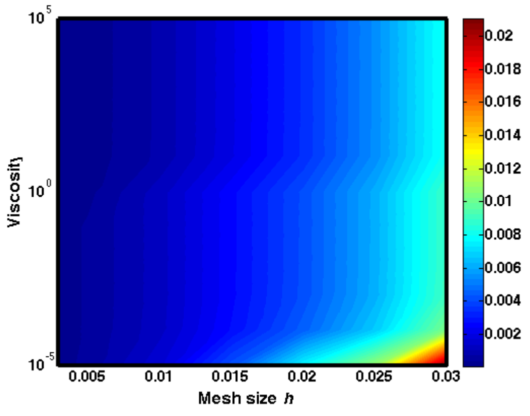
$$\mathbf{N}^k_\Gamma = \int_\Gamma \boldsymbol{\phi}^{k\,\mathbf{t}}_{out}\,\boldsymbol{\phi}^k_{in}\,dS, \quad \mathbf{M}^k_\Gamma = \int_\Gamma \boldsymbol{\phi}^{k\,\mathbf{t}}_{out}\,\boldsymbol{\phi}^k_{out}\,dS, \quad \mathbf{F}^k_\Gamma = \int_\Gamma \boldsymbol{\phi}^{k\,\mathbf{t}}_{out}\,u^k_{h,d}\,dS \tag{3.26}$$

By regrouping in the right order the terms in matrices $\mathbf{N}^k_\Gamma, \mathbf{M}^k_\Gamma$, and vectors $\mathbf{F}^k_\Gamma$, $\mathbf{U}^k_{in}$ and $\mathbf{U}^k_{out}$ we end with a system of the form:

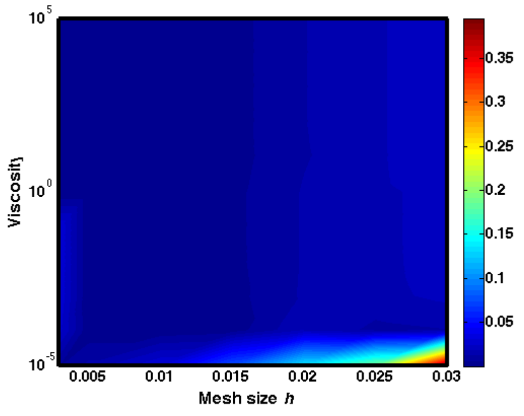
$$\begin{pmatrix} \mathbf{K}_{in,in} & \mathbf{K}_{in,out} \\ \mathbf{N}_\Gamma & \mathbf{M}_\Gamma \end{pmatrix} \cdot \begin{pmatrix} \mathbf{U}_{in} \\ \mathbf{U}_{out} \end{pmatrix} = \begin{pmatrix} \mathbf{F}_{in} \\ \mathbf{F}_\Gamma \end{pmatrix} \tag{3.27}$$

obviously taken at a given time step $n+1$ and at a given iteration $k+1$.

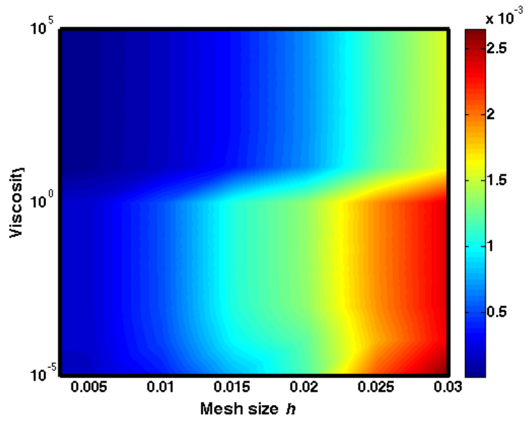
The construction of the above matrix can be computed at the elementary level and without changing the main structure of the code. After having computed the elemental matrix contributions in the main numerical integration points loop one can delete the rows associated to the testing of the equation with velocity shape functions in the nodes of the cut elements that lie inside the solid (L_{-1}) and add the proper terms defined by (3.26). All the code required for this method was therefore included in hook *Hook.PreDirichlet* in pseudo-code (1). The problem is that the ghost stabilization terms on such nodes are deleted and therefore stability issues could arise. In addition, since the equations are no more the same on those nodes, some other stabilization technique should be developed. Nevertheless, this method can be very accurate to impose the BC if the cutting of the boundary is not too close to the nodes inside the fluid in the cut elements, as it turned out to be the case for the errors shown here.



(a) $e_{\mathbf{u},\,L^2(\Omega)}$, Using external degrees of freedom

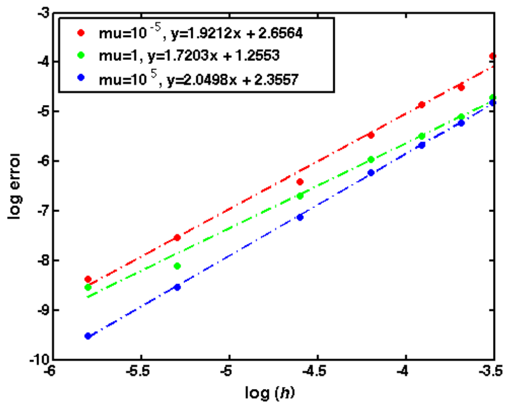


(b) $e_{p,\,L^2(\Omega)}$, Using external degrees of freedom

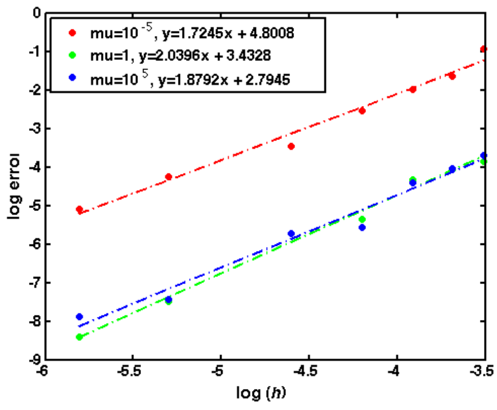


(c) $e_{u,\,L^2(\Gamma)}$, Using external degrees of freedom

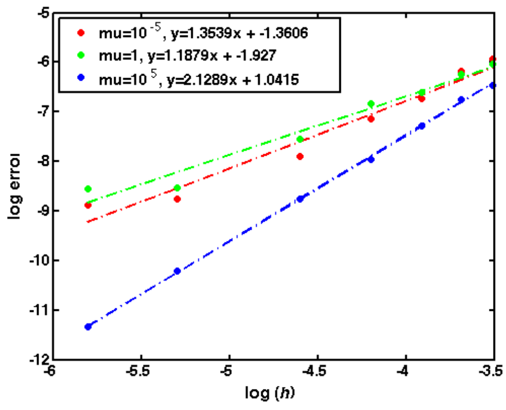
Figure 3.17: Plots of the numerical errors (external degrees of freedom).



(a) $e_{\mathbf{u},\,L^2(\Omega)}$, Using external degrees of freedom



(b) $e_{p,\,L^2(\Omega)}$, Using external degrees of freedom



(c) $e_{\overline{\mathbf{u}},\,L^2(\Gamma)}$, Using external degrees of freedom

Figure 3.18: Log-log convergence curves with linear fits (external degrees of freedom).

On Figures 3.17 and 3.18 can be seen the color plots with the absolute errors as a function of mesh size h and viscosity μ and the log-log convergence plots for some of those viscosities. We recall the reader that this method has no additional parameter.

The convergence orders obtained are satisfactory, even though slightly suboptimal in velocity. This was due to the fact that the cutting of the boundary on the background mesh for very fine meshes produced localized instabilities, that cannot be resolved with this method as explained above. This can be seen in both Figures 3.18a and 3.18c, where for the finer mesh the boundary error does not follow the trend of the aligned points and therefore provoques as well an increase on the full velocity error since the effect of these localized instabilities can span over quite a large layer of elements.

3.4 The flow past a circular cylinder

Once all the methods to approximate Dirichlet boundary conditions have been described and their numerical convergence with respect to an analytical solution assessed, we decided that a good exercise to see their performance in a more or less *comparative* way was to perform a well-known benchmark, the flow past a circular cylinder, which yields the well-studied Von Karman vortexes from a given Reynolds number. The geometry of the problem and its boundary conditions are shown on Figure 3.19, where the weak Dirichlet BC will be imposed in the inner cylinder. We first wanted to assess in a simple visual way if the methods were powerful enough to induce the typical behaviour of the flow for such geometric conditions and finally found out, as we will see, that it pointed out an important drawback of our formulation.

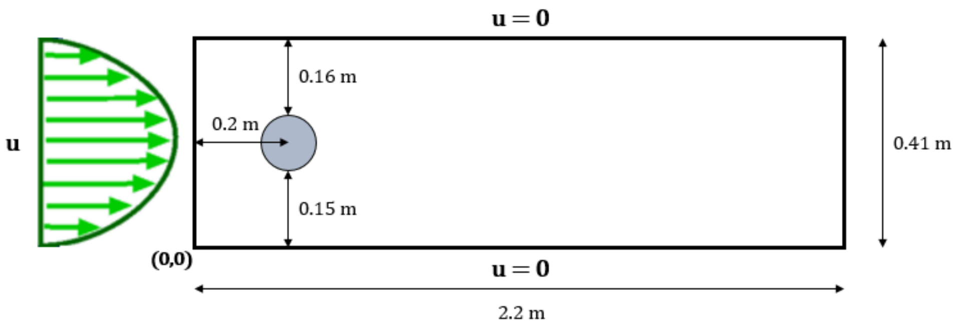


Figure 3.19: Geometry and boundary conditions for the flow past a cylinder benchmark problem

The parabolic inflow profile is defined as:

$$\mathbf{u}(\,(0,y)\,,\,t) = 4\frac{U_{max}y(0.41-y)}{0.41^2} \tag{3.28}$$

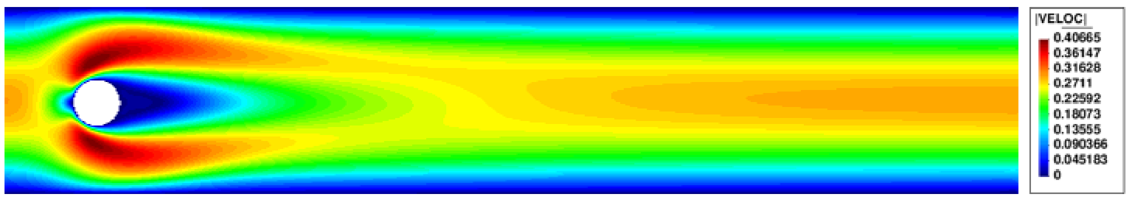
where by varying U_{max} the Reynolds number of the problem can be controlled. If the Reynolds number is computed with the mean velocity in the profile $\overline{U}$ we find te following values:

$$Re = \frac{\overline{U}D}{\nu} = 20 \quad \text{if } U_{max} = 0.3\,m/s \tag{3.29a}$$

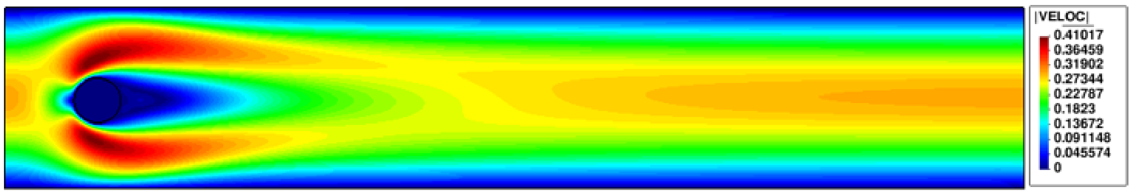
$$= 100 \text{ if } U_{max} = 1.5\,m/s \tag{3.29b}$$

where D is the diameter of the cylinder cross-section and ν has been chosen as 0.001 to compare all our results with the various existing results of the benchmark. The chosen Reynolds numbers are known to produce a steady flow for case Re=20 and an unsteady periodic flow for Re=100 with the characteristic Von Karman vortex shedding [37].

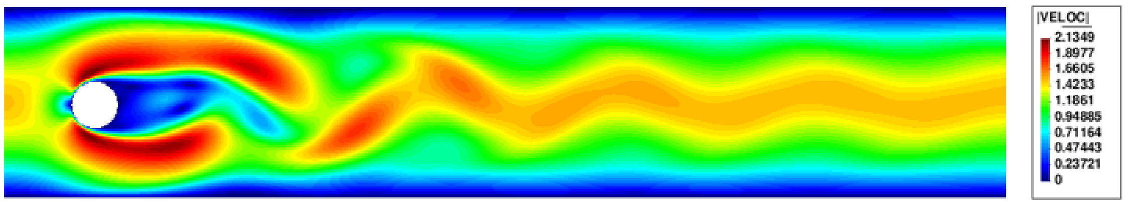
Let us first see on Figures 3.20 and 3.21 how the solution looks like when no approximate boundary conditions are used on the immersed fixed cylinder and when the *linked* Lagrange multiplier method is used.



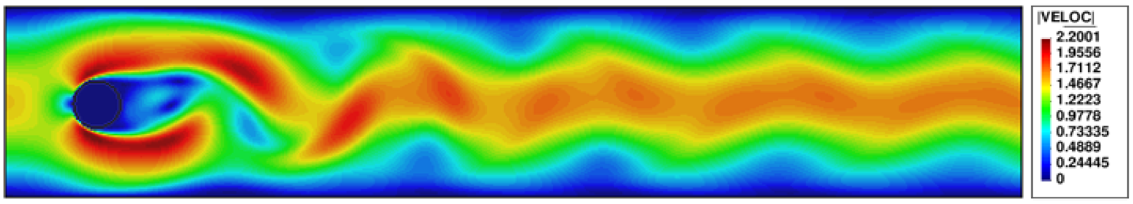
(a) Strong Dirichlet BC, Re=20, steady solution



(b) *Linked* Lagrange multiplier method, Re=20, steady solution

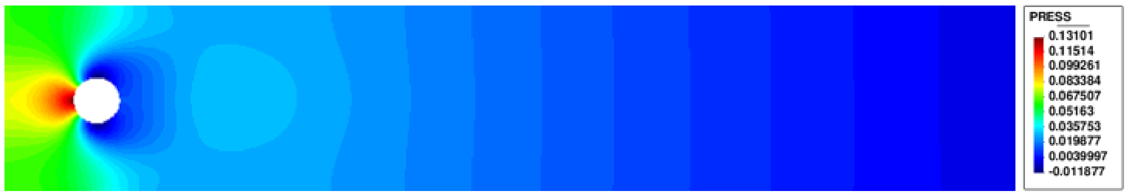


(c) Strong Dirichlet BC, Re=100, unsteady solution at t=5s

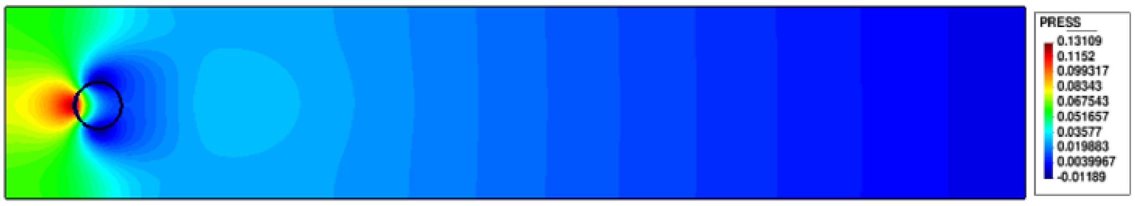


(d) *Linked* Lagrange multiplier method, $Re = 100$, unsteady solution at t=5s

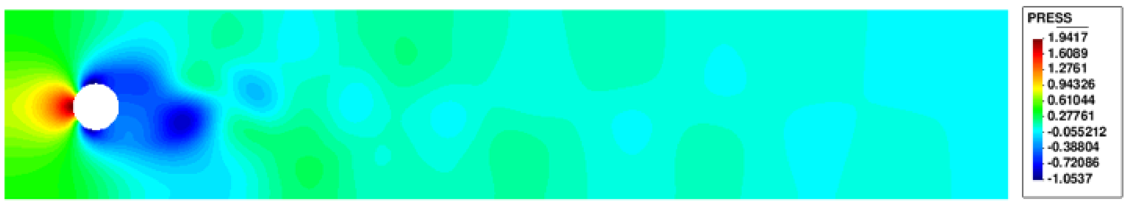
Figure 3.20: Solution $\|\mathbf{u}_h\|$ for the flow past a cylinder for Re=20 and Re=100



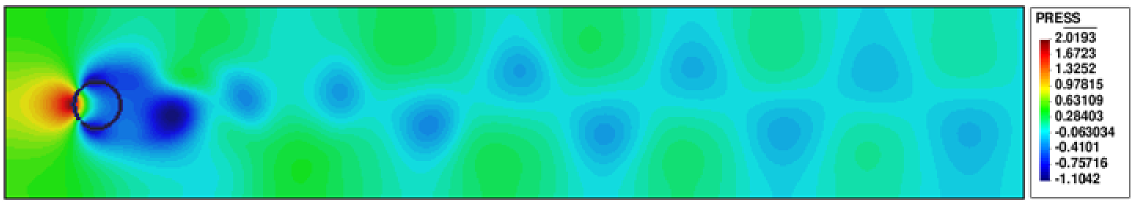
(a) Strong Dirichlet BC, Re=20, steady solution



(b) *Linked* Lagrange multiplier method, Re=20, steady solution



(c) Strong Dirichlet BC, Re=100, unsteady solution at t=5s



(d) *Linked* Lagrange multiplier method, Re=100, unsteady solution at t=5s

Figure 3.21: Solution p_h for the flow past a cylinder for Re=20 and Re=100

The solutions exhibit mostly the same result, the small differences may be due to the different approaches, but still we see that all the physics are captured, as for example the vortex street that appears for Re=100. The phase difference in the time-dependent solutions (and therefore the visible differences in the solutions themselves) should not be surprising since the time for which the vortexes start to detach depends on a large amount of numerical parameters. Other slight differences could be assigned to differences in the mesh, since for the *weak imposition* case we had to create a finer mesh around the projected level-set. For such vortexes to appear in both cases the mesh had to be fine enough and the time step smaller than $\Delta t = 1/100$ for a second order backward differences scheme (BDF2).

Let us now define some values that were computed to assess the validity of our methods other than visually. Several values are used in the framework of this benchmark but we want here to focus on the calculation of the drag and lift coefficients. For such a purpose, let us introduce the drag and lift force as the forces exerted by the flow on the solid along the positive x-axis and the positive y-axis and the corresponding coefficients:

$$F_D = \int_S (\mu \frac{\partial v_t}{\partial n} n_y - p n_x) dS \longrightarrow c_D = \frac{2F_D}{\rho \overline{U}^2 D} \tag{3.30a}$$

$$F_L = \int_S -(\mu \frac{\partial v_t}{\partial n} n_x + p n_y) dS \longrightarrow c_L = \frac{2F_L}{\rho \overline{U}^2 D} \tag{3.30b}$$

For the case with Re=20 the drag and lift coefficients c_D and c_L will be constant and their values will be given for the different methods, while their maximum values once the periodic flow is stable will be used as benchmarks for the unsteady flow case with Re=100. On Table 3.1 the results of the benchmark are given for the methods described previously in this chapter (except for the penalty method) and for the values described in this section, where the missing data was not computed due to the large computational times taken for the transient-periodic solution to develop and due to evidence from the low Reynolds case that the methods were already not performing as expected.

		Re=20		Re=100	
Method	Parameters	c_D	c_L	c_D^{max}	c_L^{max}
Strong imposition	-	5.5357	0.0109	3.2234	0.9572
Lagrange	$s_1 = 100$	4.9731	0.0094	3.0219	0.9172
	$s_1 = 1000$	5.1361	0.0127	-	-
	$s_1 = 10000$	3.7329	0.1098	-	-
Penalty with tractions	$s_1 = 100$	5.1732	0.0137	3.0935	0.8761
	$s_1 = 1000$	5.2342	0.0122	-	-
	$s_1 = 10000$	2.8131	0.1321	-	-
Nitsche	$s_1 = 100$	5.1561	0.0138	2.9875	0.8692
	$s_1 = 1000$	5.2271	0.0121	-	-
	$s_1 = 10000$	2.8291	0.1451	-	-
External degrees	-	3.128	-0.1251	6.1691	1.4592

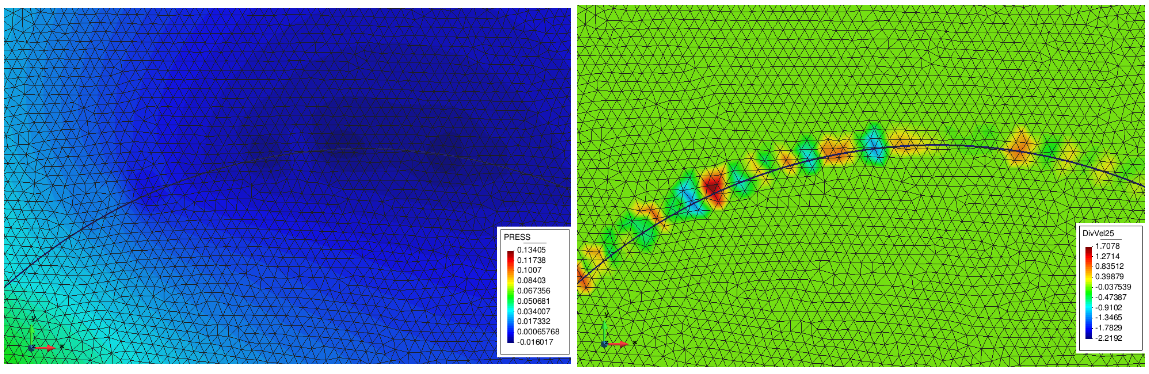
Table 3.1: Results of the benchmark of the 2D flow past a cylinder

We see that results with all of the methods are not accurate enough (the mesh used was very fine) and none of the possible calculated values fall into the expected ranges [37], with obviously the exception of the strong imposition. We were first extremely shocked by these results since we had assessed the convergence of the methods and the solutions produced were visually nearly equivalent to the solutions obtained with strong imposition for the Re=20 and Re=100 cases. Nevertheless, we realized that the *ghost stabilization* terms described at the beginning of the chapter were not necessary during the case used for the convergence analysis but they were here (it is precisely the fact of not having this stabilization that yields poor results for the method using external DOFs, which is not stabilized).

That made us analyze a bit further the nature of this stabilization but no clear conclusion has been achieved for the moment. It is important to note that the stabilization terms on velocity that the research community normally uses [38, 39] are the following:

$$S_{ghost}(\mathbf{v}_h; \mathbf{u}_h) = \sum_{f \in F_g} \alpha_f \langle [\![\partial_{\mathbf{n}} \mathbf{u}_h]\!], [\![\partial_{\mathbf{n}} \mathbf{v}_h]\!] \rangle_f \tag{3.31}$$

where F_g is the set of faces cut by the interface and the ones from the first fluid inner layer connected to those. These terms formally differ from the ones we used and presented. However one can proof that the velocity term in (3.1) is equivalent to the one in (3.31) if the orthogonal projection is also taken for the gradient of $\mathbf{v}_h$ in (3.1). We have used only $\nabla \mathbf{v}_h$ and not $\mathbf{P}_h^{\perp}(\nabla \mathbf{v}_h)$ due to orthogonality, however this property is only ensured in the whole physical domain Ω and not strictly in Ω_{cut}. This is one of the possible reasons why the stabilization terms could be giving raise to problems. Nevertheless, coding the orthogonal projection to the FE space of the gradient of the velocity test function implied an important change in the structure of the code and it has not yet been decided what will be done. A first easy alternative/try would be to compute the projections in the whole domain and take their restriction to the layers of interest inside Ω_{cut}. Another possibility would be that the addition of the non-consistent *ghost stabilization* terms could provoke a loss in the divergence-free property of the flow coming from the continuity equation. A sign of such purpose can be seen on Figure 3.22, where we see that the pressure, whose associated test function q_h is the multiplier for the continuity equation, is having an improper behaviour and the projection of the divergence of the velocity on the two affected layers is high.

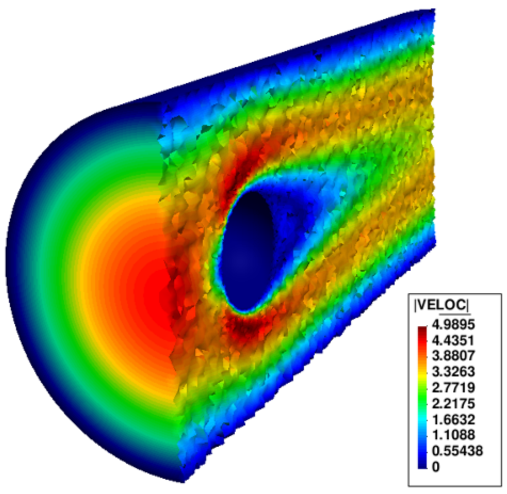


(a) p_h (b) $\mathbf{P}_h(\nabla \cdot \mathbf{u}_h)$

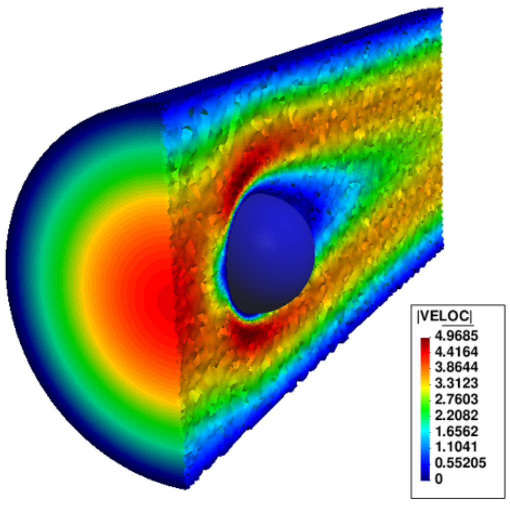
Figure 3.22: Problem in pressure field when using the *ghost stabilization* and possible explanation by the loss of the incompressibility condition.

Solving the issues related to these terms is of utter importance and we will carry on doing the tests that we have in mind. If those did not solve the issues, the terms in (3.31) will need to be explicitly implemented, since other research groups are applying them and obtaining satisfying results.

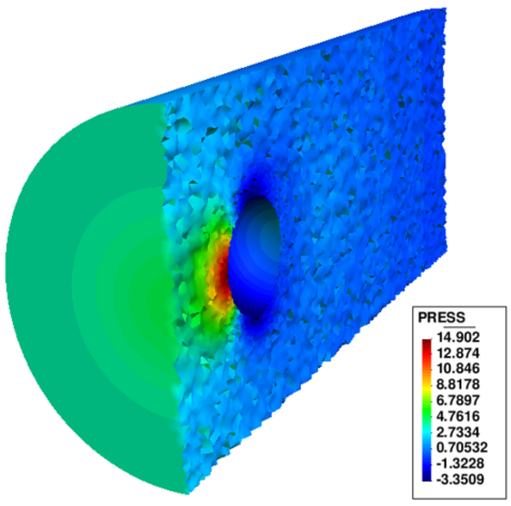
To end this chapter, where approximate BCs are used in fixed domains, we wanted to say that a similar problem was used in order to check the validity of our methods when simulating 3D problems. The approach followed until now has been quite progressive due to the harder nature of 3D problems, both in terms of physics and numerical issues. Indeed, the implementation of FEM routines for a 3D problem is always more subtle and we faced many unexpected problem when 3D cases were first analyzed. Very few 3D simulations were run, but we show in Figure 3.23 the results of a simulation of a sphere inside a cylinder with a parabolic inflow profile (U_{max}=4.5 m/s) using both strong Dirichlet BCs and the *Linked* Lagrange multiplier method. The Reynolds number was approximately $Re \simeq 30$ for which we find a stationary solution with some recirculation behind the sphere. The reader will note that the results using approximate Dirichet BCs are accurate if compared to a strong imposition. The blue sphere seen for the *Linked* Lagrange multiplier method corresponds to zero-valued iso-surface of the level-set projection into the background mesh.



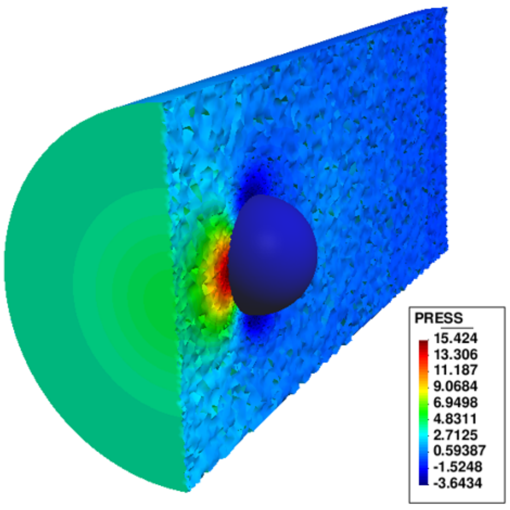
(a) $\|\mathbf{u}_h\|$, Strong Dirichlet BC



(b) $\|\mathbf{u}_h\|$, *Linked* Lagrange multiplier



(c) p_h, Strong Dirichlet BC



(d) p_h, *Linked* Lagrange multiplier

Figure 3.23: Solutions $\|\mathbf{u}_h\|$ and p_h of a 3D case of a flow at $Re \simeq 30$ in a cylinder with an embedded sphere using strong Dirichlet BCs and the *Linked* Lagrange multiplier method.

Chapter 4

Numerical treatment of moving domains

In the last chapter of this thesis we dealt with approximate imposition of Dirichlet boundary conditions (BCs) on non-matching grids using a stationary domain. Nevertheless, the reader will recall that the goal of this project is to use such techniques in problems where the physical domain Ω evolves with time. We already mentioned in the introduction of Chapter 3 that such movement requires the use of additional numerical techniques in order to gather information of the previous time step(s) to compute time derivatives. We refer the reader to Figure 3.2 to recall what is the problem when a pure fixed-mesh is used regarding time integration.

Several solutions exist to deal with moving domains when using fixed meshes. The most straightforward idea is to extend the solution to the nodes of the background mesh inside the solid domain in what is usually called a fictitious domain method [40]. In such framework, several possibilities concerning the extension exist; the problem being that none of them provides a physical solution since time derivatives near the boundary could be calculated using velocity values from the non-physical meaningless domain. We will therefore use a method currently used at CIMNE called the Fixed-Mesh ALE method (FM-ALE), which can be summarized as an Arbitrary Lagrangian Eulerian (ALE) method that projects back into the original fixed mesh to avoid large distortions. In the following sections we will introduce the FM-ALE method, which will require explaining what is an ALE method and rewriting the Navier-Stokes (N-S) equations in the mesh system of reference. We will then solve a 2D case of a rotating fan using all the notions introduced until now in this work.

4.1 The Fixed-Mesh Arbitrary Lagrangian-Eulerian method

In order to cope with the issue of time integration over fixed meshes, the Fixed-Mesh Arbitrary Lagrangian-Eulerian method (FM-ALE) [2] was developed with the objective of accurately computing the temporal derivative at nodes close to the boundary by using an ALE strategy followed by a projection step back to the original background mesh. In order to understand the algorithmic steps involved in such numerical approach, we need here to briefly introduce the reader to ALE formulations.

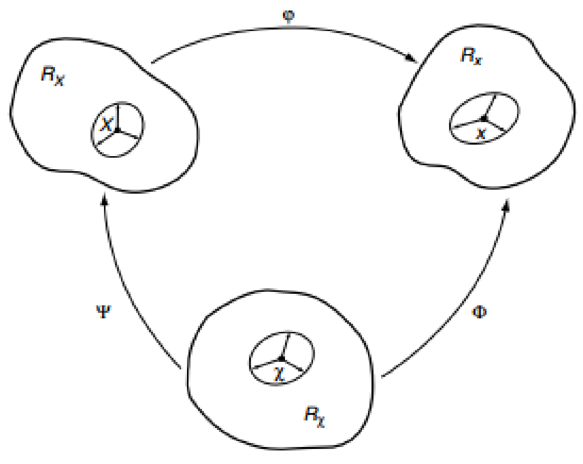
A fundamentally important consideration when developing a computer code for simulating mechanical problems is the choice of an appropriate *kinematical description* of the continuum we need to deal with. This is of utter importance since the choice determines the relationship between the deforming continuum and the finite grid or mesh and therefore conditions the ability of the numerical method to deal with large distortions and to provide an accurate resolution of material interfaces and mobile boundaries. During our academic program we have been introduced and led to use algorithms of continuum mechanics that make use of the two classical descriptions of motion: the Lagrangian description and the Eulerian description. The first description produces algorithms where the computational mesh follows the particles during motion, potentially provoques large distortions, while the second uses fixed computational meshes where boundaries are difficult to track in a direct way [1]. The Arbitrary LagrangianEulerian (ALE) description was developed in an attempt to combine the advantages and minimize the drawbacks of the above classical kinematical descriptions. The idea of the ALE description is that the nodes of the computational mesh may be moved in an arbitrary manner with the continuum to give a continuous rezoning. Figure 4.1b is illustrative of this definition. If such approach is to be used, we see that a technique to choose the arbitrary mesh velocity will need to be implemented.

The form of the different conservation-balance equations on an ALE kinematical description is obviously different from the form when written in purely Eulerian or Lagrangian description. The ALE description can in fact be understood as a generalization of the classical Lagrangian and Eulerian descriptions of motion if a so-called referential domain is introduced as well as the mapping between the referential domain and the classical, material, and spatial domains, depicted in Figure 4.1b. If the bijective mapping $\boldsymbol{\Psi}^{-\mathbf{1}} \equiv \boldsymbol{\chi}$ between the material and referential domain which for every point $\mathbf{X} \in \Omega(0)$ and time instant t returns a point $\mathbf{x} = \boldsymbol{\chi}(\mathbf{X}, t) \in \Omega(t)$ is introduced, we can define the following quantity:

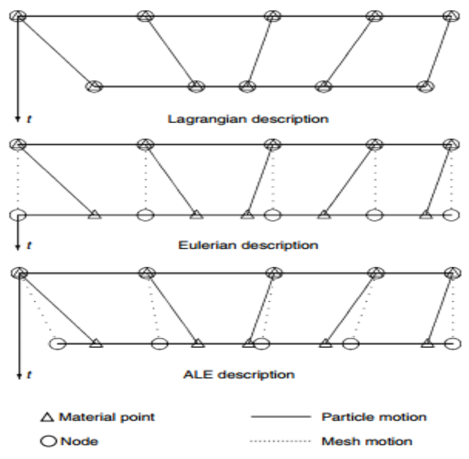
$$\mathbf{u}_{dom}(\mathbf{x}, t) = \frac{\partial \boldsymbol{\chi}(\mathbf{X}, t)}{\partial t} \tag{4.1}$$

where $\mathbf{u}_{dom}$ can be interpreted as the velocity of the mesh with respect to the material domain. With the introduction of this notion the ALE form of the two conservation

[1]What has been introduced since this point was obviously using an Eulerian description



(a)



(b)

Figure 4.1: (a) Mappings between referential, spatial and material domains, (b) One-dimensional example of Lagrangian, Eulerian and ALE mesh and particle motion. Both taken from [3].

equations we are dealing with (mass and momentum) can be written as:

$$\left.\frac{\partial \rho}{\partial t}\right|_{\boldsymbol{\chi}} + \mathbf{a}\cdot\nabla\rho = -\rho\boldsymbol{\nabla}\cdot\mathbf{u} \qquad \text{Conservation of mass} \tag{4.2a}$$

$$\rho\left.\frac{\partial \mathbf{u}}{\partial t}\right|_{\boldsymbol{\chi}} + \rho\,(\mathbf{a}\cdot\nabla)\,\mathbf{u} - \nabla\cdot\underline{\underline{\boldsymbol{\sigma}}} = \rho\,\mathbf{f} \qquad \text{Conversation of linear momentum} \tag{4.2b}$$

with $\mathbf{a} := \mathbf{u} - \mathbf{u}_{dom}$. It is pretty straightforward to see that the Lagrangian and Eulerian forms are contained in such formulation for $\mathbf{a} = \mathbf{0}$ and $\mathbf{a} = \mathbf{u}$ respectively.

Let us consider a region $\Omega_0 \subset \mathbb{R}^d$ (d = 2,3) where a flow will be taking place during the time interval [0;T]. We consider the case in which the fluid at time t occupies a subdomain $\Omega(t) \subset \Omega_0$ and that the boundary of $\Omega(t)$ is defined by part of $\partial\Omega_0$ and by a moving boundary that we call $\Gamma_{mov} = \partial\Omega(t)\setminus\partial\Omega_0$ [2]. The incompressible N-S equations in an ALE formulation read as:

$$\rho\left.\frac{\partial \mathbf{u}}{\partial t}\right|_{\boldsymbol{\chi}} + \rho\,\mathbf{a}\cdot\nabla\mathbf{u} - \mu\boldsymbol{\Delta}\mathbf{u} + \nabla p = \rho\,\mathbf{f} \qquad \text{in } \Omega(t), \quad t>0 \tag{4.3a}$$

$$\nabla\cdot\mathbf{u} = 0 \qquad \text{in } \Omega(t), \quad t>0 \tag{4.3b}$$

with the appropriate BCs [3]. We see that the only difference in practice with respect to the Eulerian version used until now is that the advective velocity **a** is now different from

[2]This moving boundary Γ_{mov} could represent an interface in various situations: a solid with an imposed movement, a solid in a fluid-structure interaction problem, a free surface, etc.

[3]On Γ_{mov} those will be Neumann conditions for a free surface and Dirichlet conditions for an immersed body

$\mathbf{u}$. When it comes to the variational or weak formulation of the problem the only difference will be as well the role of the advective velocity. Therefore everything that had been exposed in Chapters 2 and 3 holds, including the use of approximate BCs if non-matching grids are used and Dirichlet conditions have to be applied. The only added element is the choice of $\mathbf{u}_{dom}$ and that the mesh should be deformed after every time step.

For the Arbitrary Lagrangian-Eulerian mesh movement $\mathbf{u}_{dom}$ various possibilities for defining the mesh velocity exist. The essential requisite for the mesh movement is that the area of the mesh which covers Ω at time t_n is deformed in such a way that it covers Ω at t_{n+1}. This obviously translates into a boundary condition on the mesh velocity normal to the boundaries:

$$\mathbf{n} \cdot \mathbf{u}_{dom} = \mathbf{n} \cdot \mathbf{u} \qquad \text{in}\, \Gamma(t), \quad t>0 \tag{4.4}$$

and for the movement in the rest of the domain, a smooth extension of the mesh displacement in the boundaries needs to be obtained. Several methods exist for this which minimize the element distortion, such as solving elastic problems with elementally varying stiffness coefficients or using Laplace problems [41]. We opted for the last choice and therefore solved:

$$\boldsymbol{\Delta} \mathbf{u}_{dom} = 0, \qquad \text{in } \Omega_0 \tag{4.5a}$$

$$\mathbf{n} \cdot \mathbf{u}_{dom} = \mathbf{n} \cdot \mathbf{u} \qquad \text{in } \Gamma(t) \tag{4.5b}$$

where Ω_0 represents a fixed domain for which $\Omega(t)$ satisfies that $\Omega(t) \subset \Omega_0$, $\forall t \in [0; T]$. This problem is solved using a standard Galerkin finite element approximation, where the Dirichlet slip boundary condition has to be weakly enforced in the part of $\Gamma(t)$ which is immersed in Ω_0. We relied for this problem on a simple penalty method as the one exposed on Section 3.1.2 adapted to a Laplace problem. We used $\mathbf{u}$ computed from the deformation of the boundary Γ_{mov} to define $\mathbf{u}$ in the boundary.

The problem of a pure ALE approach in which we deform the mesh at every time step and solve the discretized version of system (4.3) is that, as explained in the introduction of Chapter 3, they are in practice unfit for use when we deal with simulations where the computational domain undergoes very large deformations since the necessary remeshing process to build a new mesh with good element geometrical properties requires to stop the parallel simulation in order to call a mesh generator. The main idea behind the FM-ALE method is that since the mesh does not match the boundary of the physical domain and $\Omega(t) \subset \Omega_0$, the original undistorted mesh covering domain Ω_0 can always be used as the new mesh when element distortion becomes too large and is in fact the only mesh in which the solution is computed if approximate BCs are used. The reader will be therefore able to understand why the method has been called *Fixed-mesh Arbitrary Lagrangian-Eulerian method*.

We are now ready to define the algorithmic steps of the FM-ALE method, which are the following:

- **Step 1: Boundary function update.** The moving boundary Γ_{mov} is updated in a way which is completely problem dependent. The updating may be due to the movement induced from the solid in a fluid-structure interaction problem or from the movement of a free surface using the velocity from the previous step. Therefore, the blue line in the top-right configuration of Figure 4.2 is updated to the position defined in the bottom-left configuration.

- **Step 2: Compute mesh velocity.** The update of the boundary function defines the deformation of the domain from $\Omega(t_n)$ to $\Omega(t_{n+1})$. As a consequence, the mesh M_n (depicted on the top-right configuration of Figure 4.2) needs to be deformed to adapt to the domain $\Omega(t_{n+1})$, task that is done by means of a mesh velocity. The mesh velocity on the boundary points can be computed from their position $\mathbf{x}_{n+1}$ with a simple first-order finite difference scheme, while it is extended using Equation (4.5a).

- **Step 3: Write down the ALE incompressible Navier-Stokes equations on the deformed mesh.** The previous two steps define the domain $\Omega(t_{n+1})$ and a mesh that will be here called $M_{n+1,virt}$ (bottom-left) that is obtained from deforming mesh M_n using a finite differences approach with $\mathbf{u}_{dom,n+1}$ obtained in Step 2. We now can write the incompressible N-S equation using the discretized stabilized version of the system in (4.3) in the deformed mesh. We will call our solution on mesh $M_{n+1,virt}$, $[\mathbf{u}^{virt}_{h,n+1}, p^{virt}_{h,n+1}]$. The velocities $\mathbf{u}_{h,n}$ from the previous step are known since the nodes in $M_{n+1,virt}$ are obtained from deforming mesh M_n. Therefore, a pure ALE formulation obviously resolves the issue of time integration that is at the core of this Chapter.

- At this point, two options -whose idea is mostly the same and whose difference should converge to zero- exist:

  - **Step 4: Solve the ALE incompressible Navier-Stokes equations on the deformed mesh and project the solution back to the background mesh.** Let P^{n+1} be the projection of finite element functions defined on $M_{n+1,virt}$ to M_{n+1}, that, when approximate BCs are used and therefore no new boundary nodes are created and the red lines in Figure 4.2 are only for integration purpses, corresponds with M_0 (top-left in Figure 4.2). To define such a projection, for each node on $M_{n+1} = M_0$ the element in $M_{n+1,virt}$ where it is placed has to be identified. The way the projection is constructed, or transfer information between finite element meshes, could be a Master Thesis by itself so we will not give details of the procedure here. We solve for $[\mathbf{u}^{virt}_{h,n+1}, p^{virt}_{h,n+1}]$ the equations obtained from Step 3 and then project the results from $M_{n+1,virt}$ into M_{n+1} using P^{n+1}.

– **Step 4bis: Project the ALE incompressible Navier-Stokes equations written on the deformed mesh on the background mesh and solve them on the background mesh.** If we want to always solve in a fixed mesh this last step is necessary. The idea is to solve at time step $n+1$ the ALE incompressible N-S equations in the background mesh using the projection to gather the necessary information for the time integration. Let us define:

$$\mathbf{u}_{h,n+1} = P^{n+1}(\mathbf{u}^{virt}_{h,n+1}) \tag{4.6}$$

the problem becomes finding $\mathbf{u}_{h,n+1}$ and $p_{h,n+1}$ on mesh M_0 using as values of the velocity from the previous step n and as domain velocity the following:

$$\mathbf{u}_{h,n} = P^{n+1}(\mathbf{u}^{virt}_{h,n}) \tag{4.7a}$$

$$\mathbf{u}_{dom,h,n+1} = P^{n+1}(\mathbf{u}^{virt}_{dom,h,n+1}) \tag{4.7b}$$

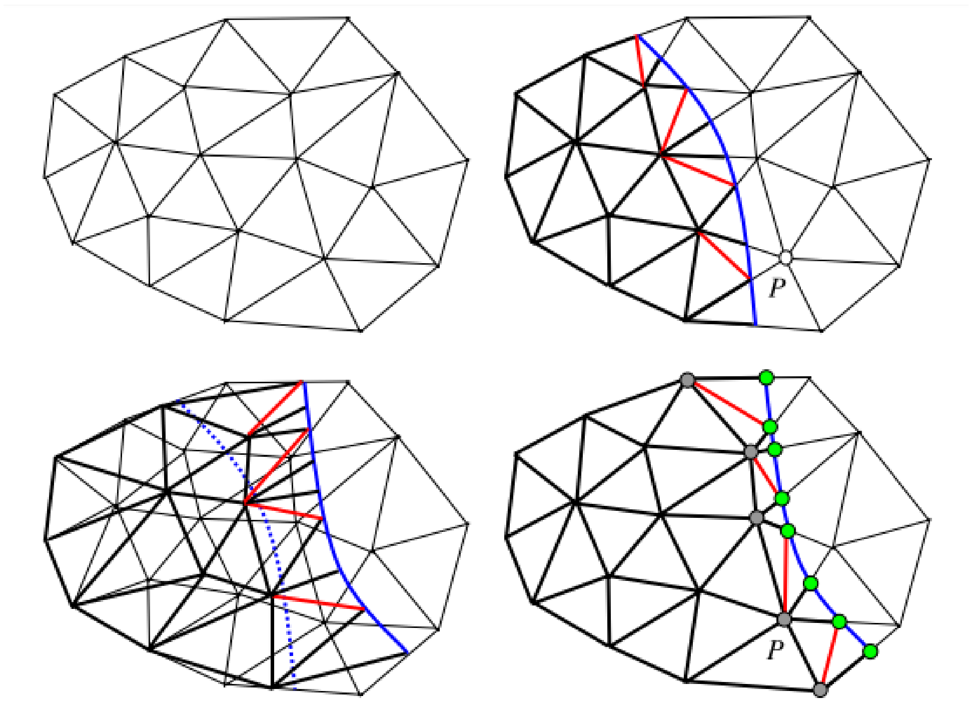


Figure 4.2: Two dimensional FM-ALE schematic. Top-left: background original fixed mesh M_0. Top-right: Position of Γ_{mov} at time t_n in blue and induced computational domain M_n, the red lines being the splitting of the cut elements for subintegration. Bottom-left: updating of M_n to $M_{n+1,virt}$ with old Γ_{mov} in dotted blue. Bottom-right: computational domain M_{n+1} with interesction points in green.

We therefore end up with a system to solve on the background mesh $M_{n+1} = M_0$ (bottom-right) that uses the ALE framework to obtain the velocities of newly created nodes, that are now, with no doubt, known (Step 4bis) or solve the equations directly on the deformed mesh, projecting them afterwards (Step 4). In any case, the underlying idea is the same, but, since we want to work with a fully fixed mesh we have chosen the second approach.

This brief description of the FM-ALE method completes our description of the numerical techniques required to solve evolving-domain incompressible flow problems. The only side ingredient that has not been described yet is how we deal with the tracking of Γ_{mov}. In general, we can assume that this part of the boundary of the flow domain is defined by what we call generically a boundary function, that may be defined analytically or by discrete means, for example through interpolation from some nodes that define the location of Γ_{mov}. In some applications, as the ones that will be developed herein, Γ_{mov} is represented with a level set function ψ, which is basically a function whose value is 0 on Γ_{mov} and different than 0 elsewhere. This function has to be advected at each time step to find the position of Γ_{mov}. For further detail on level set problems we ask the reader to refer to [42].

4.2 Numerical tests on a rotating fan

We are now ready to solve a specific problem using all the numerical techniques discussed until now. We will use a FM-ALE framework with adaptive refinement near Γ_{mov}. In order to use Dirichlet approximate BCs and not get into more complicated fluid-solid interaction problems we will deal with a solid with a prescribed motion, in our case a rotating fan. The position of the fan blades on the background mesh will be obtained by projecting a level-set function from a physical problem with only the rotating blades surrounded by a circle where a level-set problem is solved exactly (Figure 4.4a). The problem to solve is depicted in Figure 4.3 and in Figure 4.4 the reader can see the meshes used, including the mesh produced after some steps once some refinement has been produced around the zero level-set isosurface.

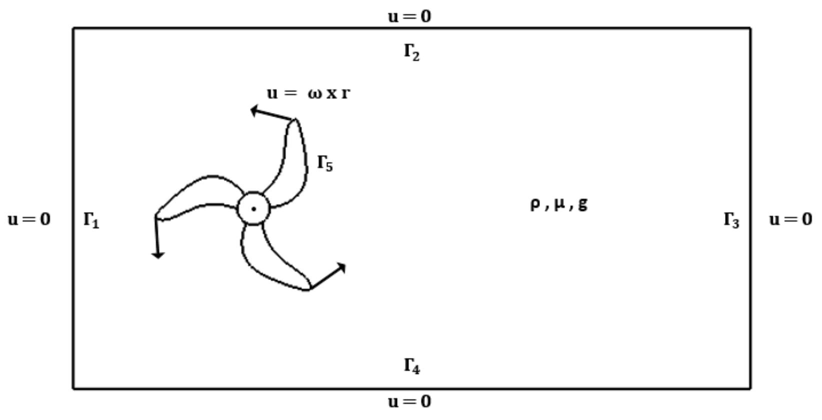


Figure 4.3: Geometry and boundary conditions of the rotating fan in a cavity problem.

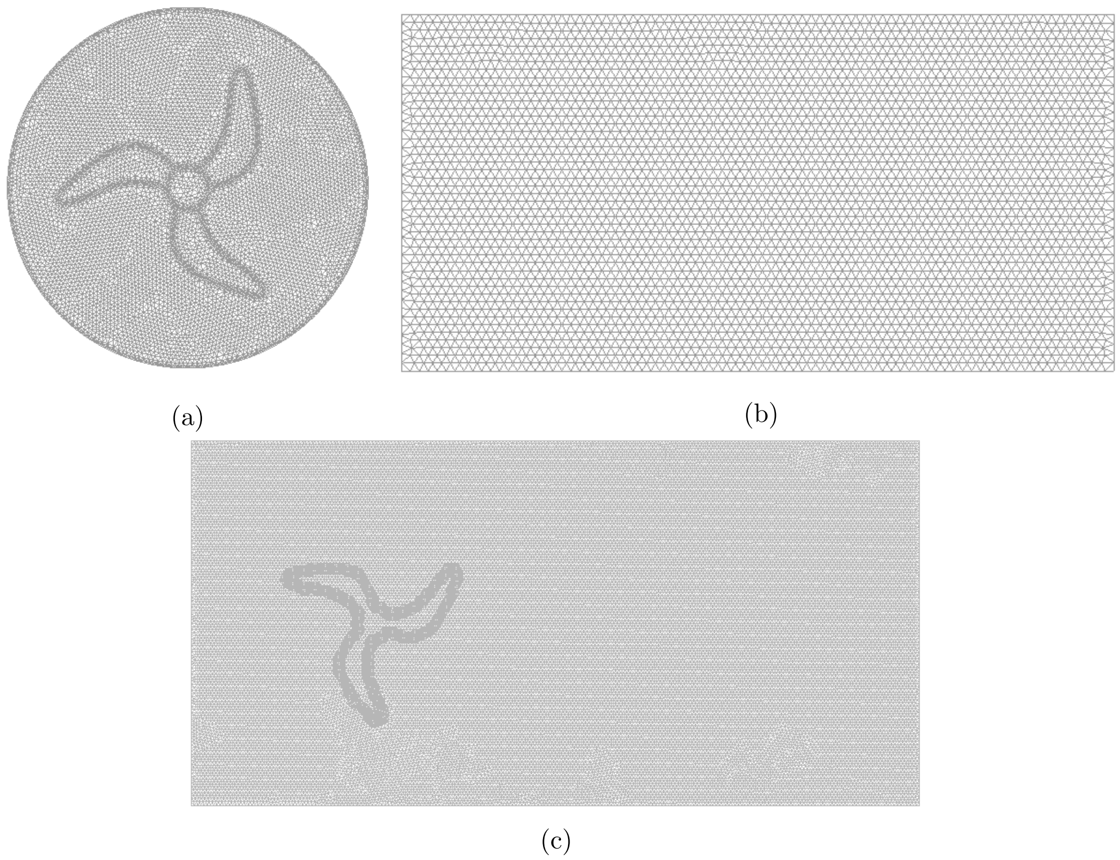
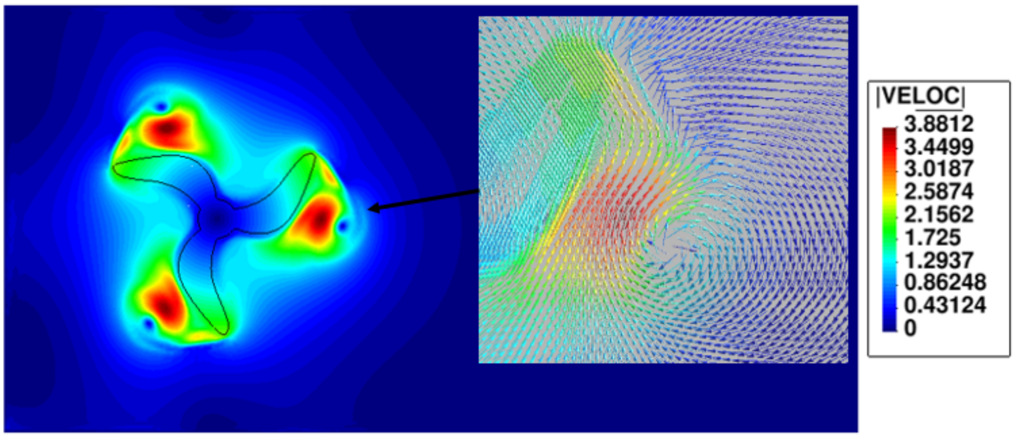


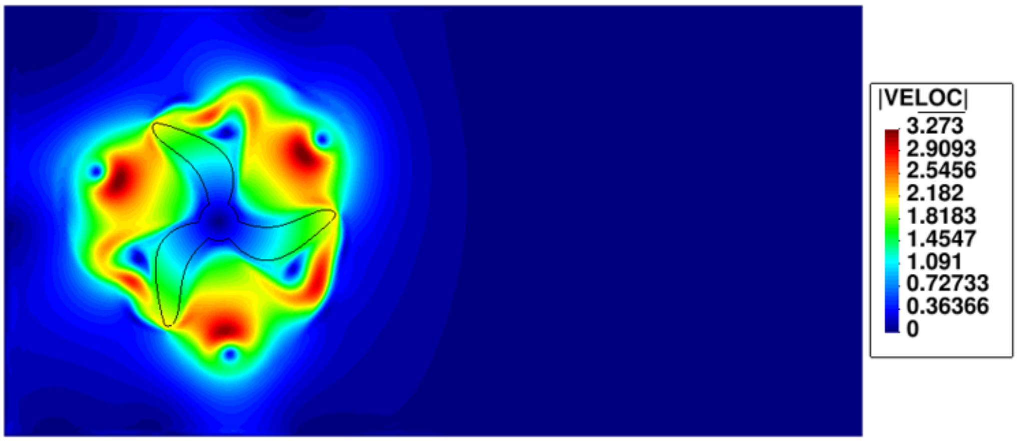
Figure 4.4: Unstructured meshes: (a) Meshing of the helix inside a circle to solve a level-set problem (b) Meshing of the background fixed mesh and (c) Computational mesh after some time steps with adaptive refinement around the projection of the level-set in the background.

The problem therefore consists of a rotating fan at angular speed ω inside a cavity. We can already see that the flow is created by the rotation of the fan, and, since the BC is imposed weakly, this gives the imposition of the BC a prominent role in the problem. After having proven that the different approximate BC methods performed well, we will here only use the *linked* Lagrange multiplier method to impose $\mathbf{u} = \boldsymbol{\omega}\times\mathbf{r}$ on the blades, while the condition on the walls of the cavity will be applied in a classical way. In what follows, if any reference to the mesh size h is done, it will be referring to the background mesh, since the helix mesh was kept fix. The reason why we used a fine mesh for the helix was only to capture the shape of the blades as accurately as possible, even though the projection of the level-set was done in such a way that the cuts were represented as straight-lines inside each element on the background mesh. Nevertheless this fine mesh on the helix allowed us to capture the boundary properly when the background mesh was sufficiently refined.

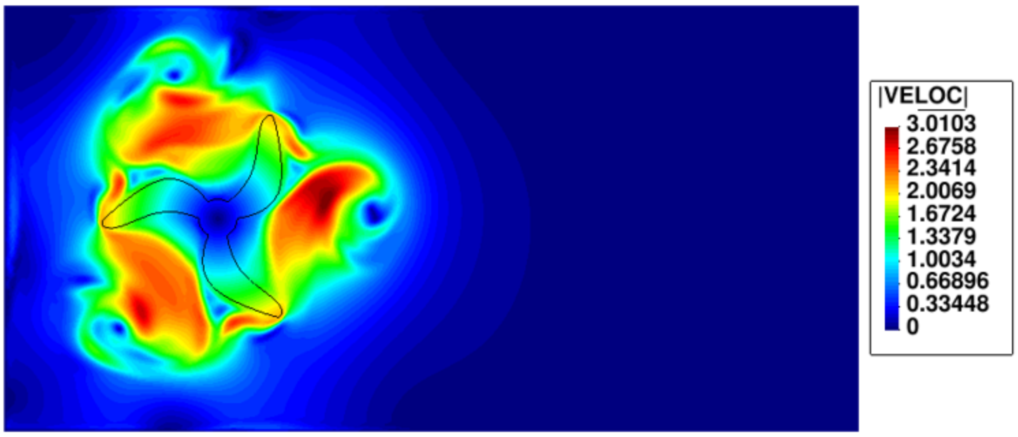
To check the proper functioning of the code and see the actual solution, we first set $\boldsymbol{\omega} = \mathbf{e}_z$ and used a very fine mesh. The physical parameters were set for the fluid to be air at 25°C, therefore $\rho = 1.09$ kg/m^3 and $\mu = 1.86 \cdot 10^{-5}$ Pa·s. In Figures 4.5 and 4.6 can be seen the solutions for the module of the velocity and the pressure at times approximately $t = i\frac{2\pi}{\|\boldsymbol{w}\|}, i = \frac{1}{4}, \frac{1}{2}, 1$.



(a) $t = \frac{\pi}{2}$

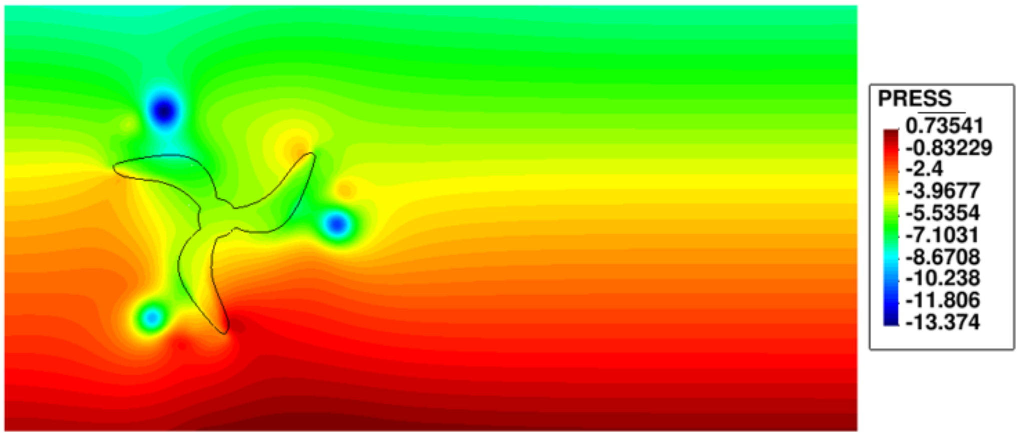


(b) $t = \pi$

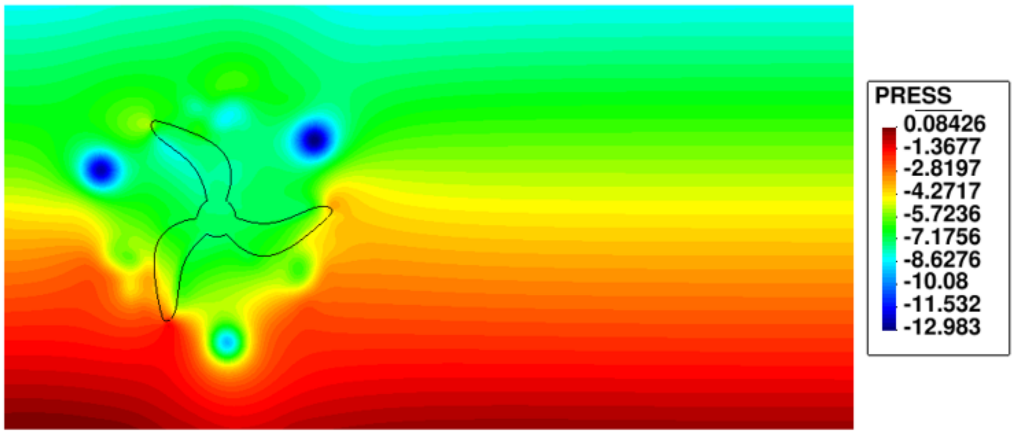


(c) $t = 2\pi$

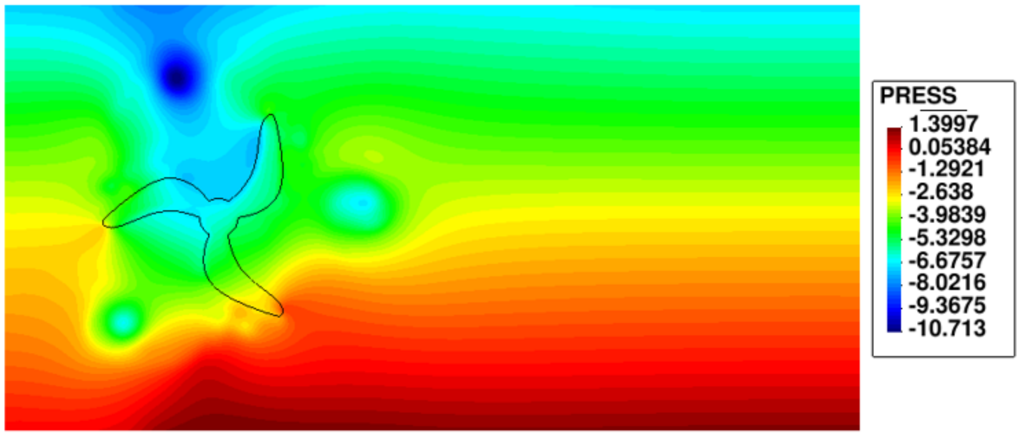
Figure 4.5: Solution $\|\mathbf{u}_h\|$ for $\mu = 1.86 \cdot 10^{-5}$ Pa·s, $\rho = 1.09$ kg/m^3



(a) $t = \frac{\pi}{2}$



(b) $t = \pi$



(c) $t = 2\pi$

Figure 4.6: Solution p_h for $\mu = 1.86 \cdot 10^{-5}$ Pa·s, $\rho = 1.09$ kg/m^3

As expected for a high Reynolds number ($Re \sim 10^6$ using the rotation speed imposed and the maximum radius) we see that vortexes are created at the edges of the blades of the fan[4]. Both the results in velocity and pressure are conforming to reality, with

[4]A caption of a region with the velocity vector field has been added on Figure 4.5a to show that the BC is well imposed in direction as well. The size of the arrows plotted being constant, the color gives the

overpressure related to the front side of the blades and viceversa, combined with the pressure distribution due to gravity and to localized depressions induced by vorticity. Nevertheless, since the evolution along the blade of the imposed boundary condition is hard to assess with these plots and even harder as soon as the vortexes are generated due to the colormap properties, we have plotted in Figure 4.7 the evolution with time of the error on the boundary condition using the same procedure as in Chapter 3. We profit here to show in Figure 4.7 such error for the different methods to weakly impose Dirichlet BCs presented previously, all of the simulations done in the same conditions, except from the setting of the penalty parameter s_2, which was chosen as varying parameter due to the convection-dominated nature of the flow. Even though we saw that parameters s were not extremely relevant in the last chapter, that was only for a given case where the solution happened to be very smooth and simple. For this case, the value of the parameter turned out to be of extreme importance for two reasons: the BC validity required high values of s_2 and the convergence of the non-linearity iterative algorithm using a Picard iterative method was harder to achieve for low values. As a consequence, values for s_2 higher than 100 were used for the further simulations.

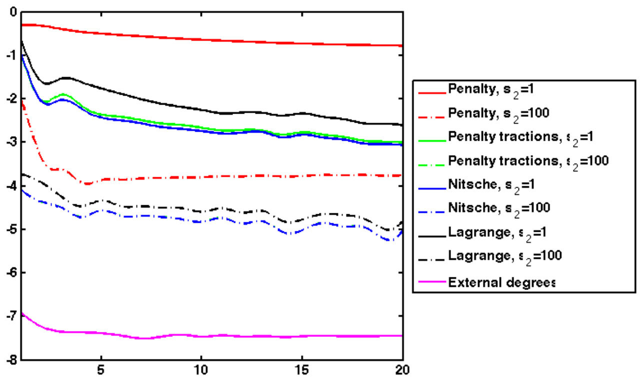


Figure 4.7: Time evolution of $e_{\overline{\mathbf{u}},\,L^2(\Gamma)}$ for the rotating fan problem.

We see that the error is reduced with increasing parameter s_2 for the methods that are defined through a parameter and that the method using external degrees of freedom performs very well in boundary condition imposition, since no local instabilities appeared during the 20 time steps of calculation.

In order to have as well an assessment of how the error in the boundary was distributed spatially, we plotted the modules of the velocities along with their radial distance to the center of rotation for different rotation speeds. The 20th time step was used since we have seen in Figure 4.7 that the error is stabilizing for those times, since previously the flow, that starts from a still fluid at $t = 0$, is being developed. We see that rotation speed is well imposed along all the blade independently of the rotation angular speed chosen

module of the velocity.

as a Dirichlet BC (Figure 4.8), except from the points at the edges of the blades, where the fit is less accurate. This may be due to the fact that vorticity is being generated at those points and we would maybe need a stronger imposition of the boundary condition. We actually performed the same fits by multiplying parameter a in the *Linked* Lagrange method by a constant and saw that the adequacy of the points at the edge of the blade to the imposed velocity could be improved.

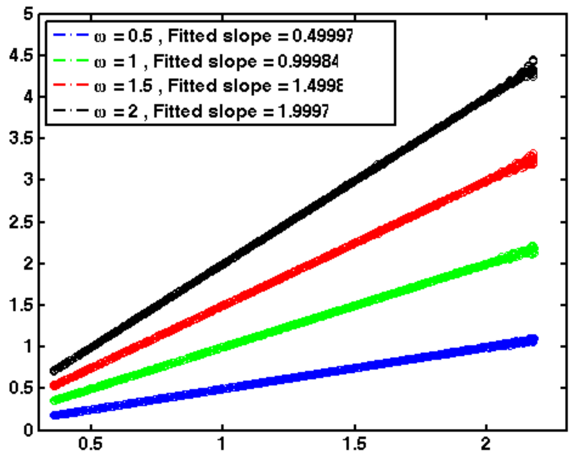
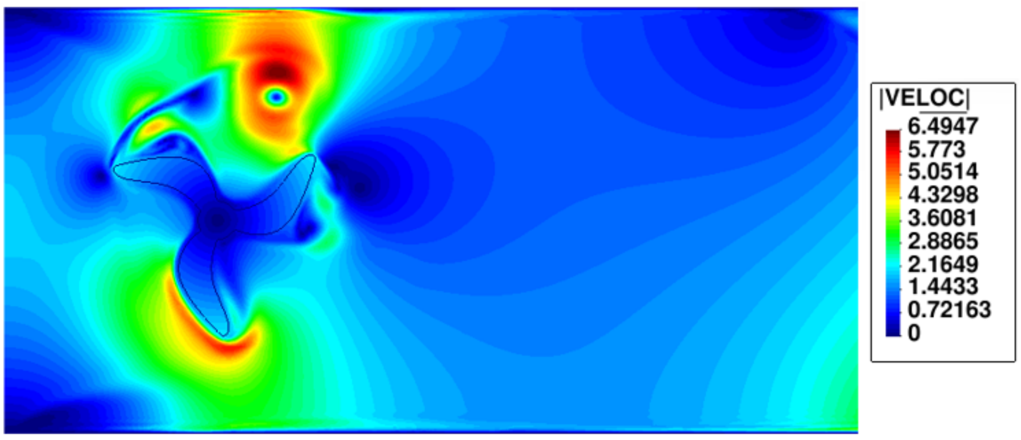
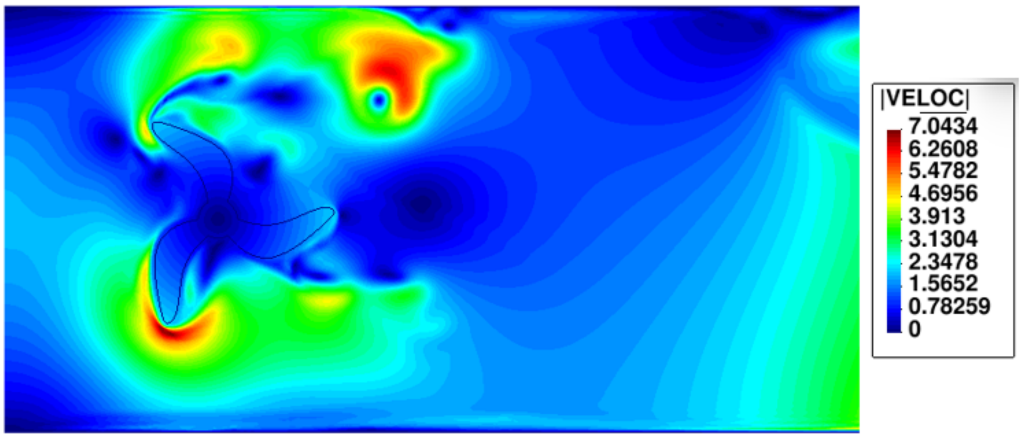


Figure 4.8: Module of the velocity $\|\mathbf{u}_h\|$ - distance r to the center of rotation for different angular velocities ω.

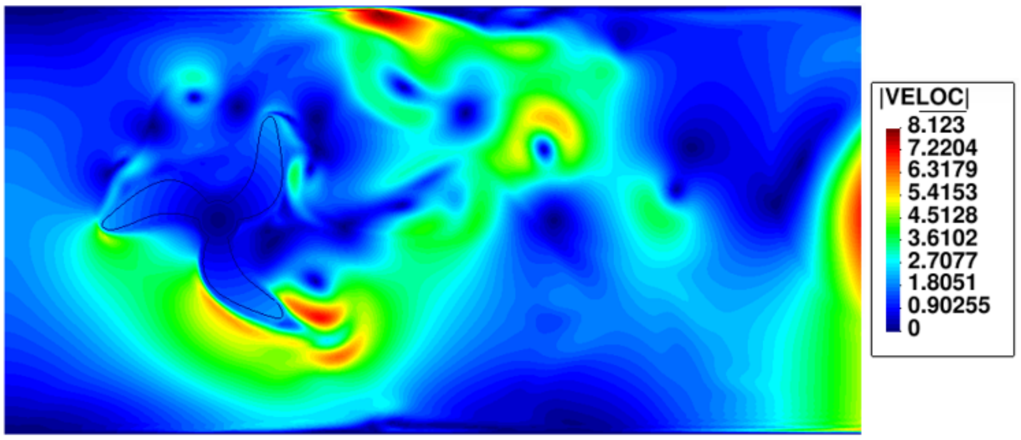
Now, since the case we wanted to end up studying was the flow around a wind turbine, we decided to assess how the approximate imposition of the Dirichlet BCs with non-zero inflow conditions performed on the case of the rotating fan. Even though here the rotation direction and the inflow velocity are in-plane, it will allow us to see the robustness of our methods to impose Dirichlt boundary conditions. The validity of the *linked* Lagrange method has been proved in a more or less systematic way since we did a series of simulations varying U_{max} in the inflow parabolic profile as well as the rotation velocity ω. The results obtained from doing the same fits as in Figure 4.8 can be seen on Table 4.1. We see that the methods are approximating the conditions with much precision for all the considered situations and the higher errors, which are still very small since the fitted slopes are very accurate, are found around the blades edges. We can therefore confirm that this particular method is suited for a broad simulation spectra. The reader can see in Figures 4.9 and 4.10 the solutions with the same conditions as previously except from an inflow condition in Γ_1 as the one in (3.28) using $U_{max} = 2$ m/s and a height of 8 m, as well as a free outflow at Γ_3. Maybe a finer mesh on the non-slip boundaries would have been necessary since gradients are extremely strong in those at some points.



(a) $t \simeq \frac{\pi}{2}$

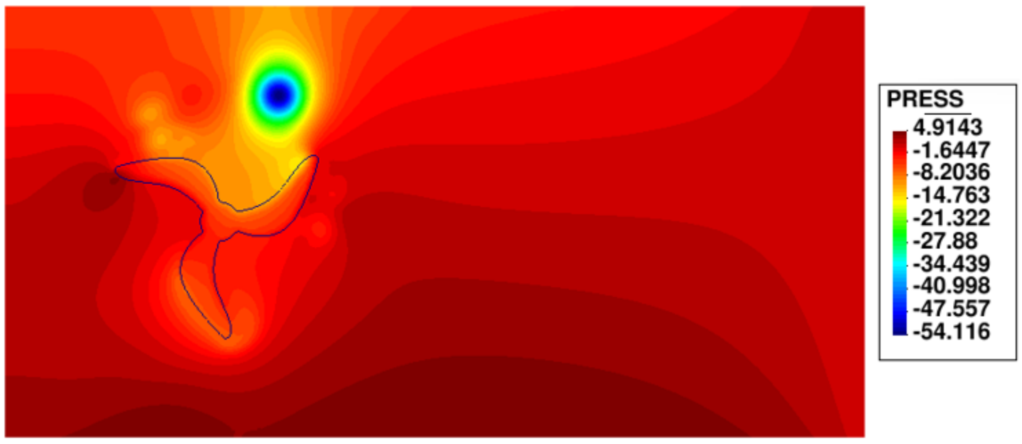


(b) $t \simeq \pi$

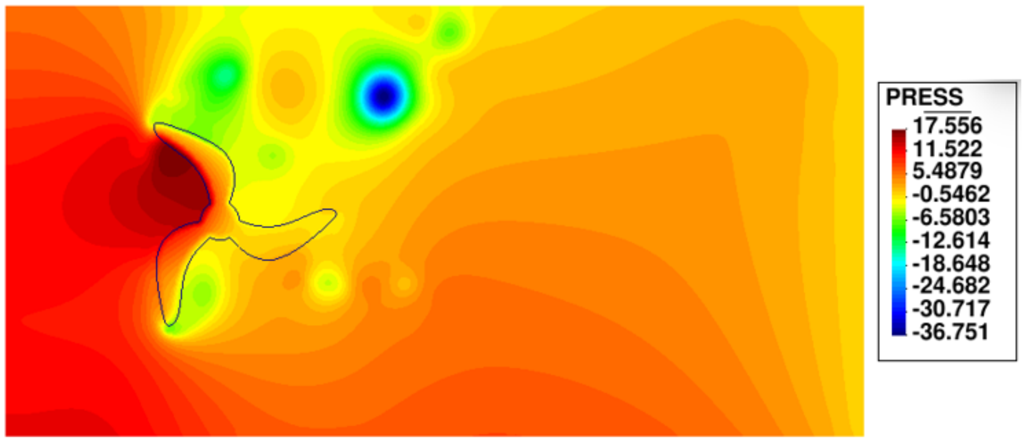


(c) $t \simeq 2\pi$

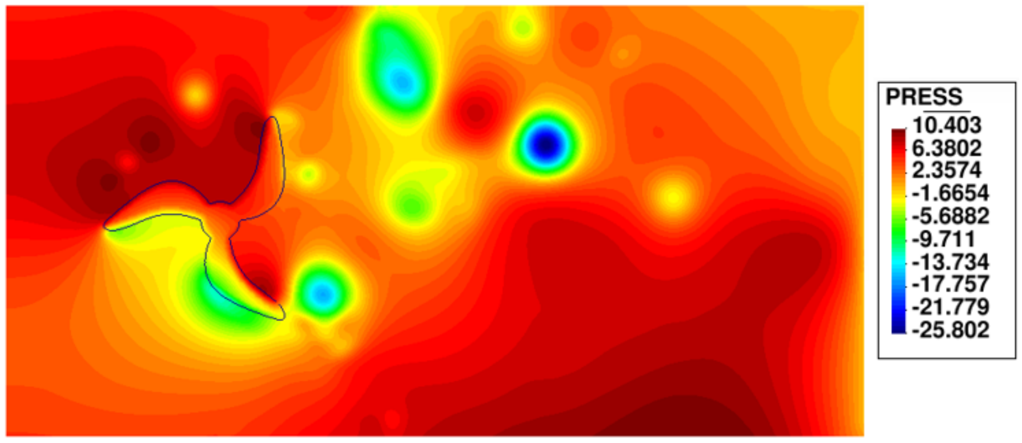
Figure 4.9: Solution $\|\mathbf{u}_h\|$ for $\mu = 1.86 \cdot 10^{-5}$ Pa·s, $\rho = 1.09$ kg/m^3



(a) $t = \frac{\pi}{2}$



(b) $t = \pi$



(c) $t = 2\pi$

Figure 4.10: Solution p_h for $\mu = 1.86 \cdot 10^{-5}$ Pa·s, $\rho = 1.09$ kg/m^3

			Angular velocity ω					
			0.5	1.0	1.5	2.0	2.5	3.0
Inflow velocity U_{max}	0	ω_{fit}	0.49997	0.99984	1.49981	1.99973	2.49923	2.99889
		R^2	0.99988	0.99991	0.99992	0.99993	0.99995	0.99995
		e_{max}	2.17251	2.17317	2.17310	2.17358	2.13553	2.14684
	0.5	ω_{fit}	0.49999	0.99987	1.49981	1.99970	2.49925	2.99880
		R^2	0.99976	0.99990	0.99992	0.99993	0.99995	0.99994
		e_{max}	2.16355	2.17317	2.17384	2.17258	2.13553	2.14684
	1.0	ω_{fit}	0.50002	0.99992	1.49982	1.99967	2.49925	2.99870
		R^2	0.99960	0.99989	0.99993	0.99993	0.99995	0.99994
		e_{max}	2.16355	2.17317	2.17384	2.17258	2.13553	2.14684
	1.5	ω_{fit}	0.50006	1.00001	1.49984	1.99965	2.49925	2.99860
		R^2	0.99941	0.99987	0.99993	0.99992	0.99995	0.99994
		e˙max	2.16355	2.17317	2.16757	2.17258	2.13553	2.14684
	2.0	ω_{fit}	0.50012	1.00013	1.49990	1.99966	2.49928	2.99851
		R^2	0.99917	0.99982	0.99992	0.99991	0.99995	0.99993
		e_{max}	2.16355	2.17317	2.16757	2.17258	2.13553	2.14684
	2.5	ω_{fit}	0.50012	1.00013	1.49990	1.99966	2.49928	2.99851
		R^2	0.99915	0.99982	0.99992	0.99991	0.99995	0.99993
		e_{max}	2.16355	2.17317	2.16757	2.17258	2.13553	2.14684

Table 4.1: Results of fitting the data points of the module of the velocity $\|\mathbf{u}_h\|$ against the distance to the rotation point r. ω_{fit} represents the fitted slope, R^2 the determination coefficient of the fit and e_{max} the radius r for which the actual velocity $\|\mathbf{u}_h\|$ is further from the fit.

Chapter 5

Conclusions and future work

The study carried and summarized within this pages was set out to explore the concept of approximate Dirichlet boundary conditions in grid-embedded boundaries and has presented some methods that fall under its umbrella. It has sought to gently introduce the notions involved and the formulation of the methods, assessing their validity by carrying a convergence analysis. The additional ingredients required to deal with time-evolving domains have been introduced and the Fixed-Mesh ALE framework presented as a solution to the issues raised by this movement. Application of the methods has been also explored to the 2D problem of a rotating fan inside a cavity and to very simple 3D problems. Despite not having achieved the goal of carrying out large numerical simulations of the flow around a turbine, several conclusions can be drawn from all the numerical ingredients that we have worked with and that we have implemented in FEMUSS.

Some of the aforementioned conclusions are the following:

- The methods to weakly impose Dirichlet BCs presented in Chapter 3 are all intrinsically appropriate and exhibit convergence orders optimal or close to optimal and very similar to the ones exhibited by the same problem with strong imposition of Dirichlet BCs. Those are therefore a powerful numerical tool that leads to simpler numerical formulations and implementations due to the possibility of having a fixed mesh. Moreover, the final resulting methods are easy to implement, because they only require some additional boundary integrals to be added to the original variational form. Upon those methods, the simple penalty method exhibits a lower performance and the method using external degrees of freedom completely fails when the arbitrary cutting of the moving interface with the background fixed mesh gives raise to instabilities that, for this method, cannot be solved using the *ghost stabilization* technique.

- Concerning precisely this *ghost stabilization* technique, the necessary stabilization when the cutting is inappropriate allows to produce an overall adapted solution far from the boundary where weak Dirichlet BCs are imposed but induces a considerable error in the approximated pressure field p_h that make the methods less robust for accurate measurements along the boundary (i.e. tractions).

- If the stabilization terms of the cuts are not active and not required, physically meaningful solutions can be obtained for the case of a 2D rotating fan inside a cavity. Vortex shedding appearing with decreasing viscosity μ, an adequate imposition of the boundary condition, a plausible pressure distribution and properly converged non-linear iterative algorithms and solvers were signs of proper functioning of the methods.

- The validity of the methods for 3D simulations was checked, leading to trustful solutions when compared to the solutions obtained when dealing with strong Dirichlet BC imposition.

In order to go further in the subject and to find a solution or deeper understanding to some of the issues raised in these pages, still a lot can be done. The main limitation of this work if conceived in the mindset we had half a year ago -being able to simulate the influence of a wind turbine rotating in the fluid motion- is the lack of results produced in terms of 3D simulations. Several unexpected problems were faced when implementing the methods described, leading to a shortcut in the objectives of the thesis. Another limitation, but already made as an assumption at the beginning of the thesis, is that we only considered prescribed rigid solids, which leaves us far from the real simulation of flow-induced rotations, where energy transfer between the fluid and the solid is precisely the main mechanism and usually the objective of the whole system. Therefore, fluid-structure interaction or at least fluid-rigid solid interaction should be considered and the required changes in the approximate BCs subroutines introduced. Apart from these general remarks concerning what should be done to go further on the issue and on the application of such methods to practical cases, some other more specific are the following:

- In order to find a solution to the problem that appears when using the stabilization terms and to assess it in more depth, we propose to try to create a mesh for such cases that allows us to decouple the imposition of the BC and the stabilization. What we mean by this is, for example, to create a stationary case using quadrilateral elements all cut by the projected level-set or by a given analytical function in its middle line. That would allow us to deactivate the cut stabilization terms and actually only see the effect of the BC approximate imposition. Another approach would be to slightly change the shape of the interface in case the cut is close enough to the fluid node to give raise to instabilities by taking the node itself as intersection point.

- Working with analytical functions to define the interface could be an interesting solution if the shape of the interface is *a priori* known. We did not follow this approach since we wanted to set the problem for the most general case. The projection of a level-set function is more versatile to deal with, for example, fluid-solid interaction.

- Regarding the *ghost stabilization* terms, the implementation of the commonly used terms (3.31) seems unavoidable in order to be able to benchmark our results. Even though the stabilization we implemented should be equivalent, its numerical implementation uses assumptions such as a lumped mass on the calculation of the projections that can make the gap between both methods wider than it seems *a priori*.

- One of the unmentioned things in the formulation presented resulting from applying approximate Dirichlet BCs is the role of the subscales $\mathbf{u}^{'}$ on the immersed boundary. The assumption we did in Section 2.3 was that $\mathbf{u}^{'} = \mathbf{0}$ in ∂K, the edges of the elements, but therefore $\mathbf{u}^{'} \neq \mathbf{0}$ in Γ for non-matched grids. This fact has been neglected and has in fact not been much investigated yet. It could therefore be an interesting field for further work.

Those are only some ideas that we had in mind during these last months but we are aware that a broad field of research can make use of the methods exposed here and therefore some other diverse applications could be considered. Nevertheless, we believe that the work developed lays the foundations of some of the methods that could be used for large-scale numerical simulations in numerous disciplines.

Appendix A

Matricial form VMS incompressible Navier-Stokes equations

The goal of this appendix is to present the exact definition of the matrices involved in solving the linear system of equations issued from the applications of the Variational Multiscale Method (VMS) to the incompressible Navier-Stokes equations, as explained in Section 2.3.

Using a simple Picard iterative method for non-linearity and a first order backward Euler finite time differences approach and with a problem with full Dirichlet boundary conditions, in the exact same conditions as the ones explained in detail in Chapter 2, the linear system of equations within the VMS framework writes:

$$\begin{pmatrix} \tilde{\mathbf{K}}_{\mathbf{UV}} & \tilde{\mathbf{K}}_{\mathbf{UQ}} \\ \tilde{\mathbf{K}}_{\mathbf{PV}} & \tilde{\mathbf{K}}_{\mathbf{PQ}} \end{pmatrix} \cdot \begin{pmatrix} \mathbf{U}_{n+1}^{k+1} \\ \mathbf{P}_{n+1}^{k+1} \end{pmatrix} = \begin{pmatrix} \tilde{\mathbf{F}}_{\mathbf{v}\,n+1}^{\;k+1} \\ \tilde{\mathbf{F}}_{\mathbf{q}\,n+1}^{\;k+1} \end{pmatrix} \tag{A.1}$$

If the following ordering of the degrees of freedom involved in such system is chosen:

$$\begin{pmatrix} \tilde{\mathbf{K}}_{u^xv^x} & \tilde{\mathbf{K}}_{u^yv^x} & \tilde{\mathbf{K}}_{pv^x} \\ \tilde{\mathbf{K}}_{u^xv^y} & \tilde{\mathbf{K}}_{u^yv^y} & \tilde{\mathbf{K}}_{pv^y} \\ \tilde{\mathbf{K}}_{u^xq} & \tilde{\mathbf{K}}_{u^yq} & \tilde{\mathbf{K}}_{pq} \end{pmatrix} \cdot \begin{pmatrix} \mathbf{U}_{x,n+1}^{k+1} \\ \mathbf{U}_{y,n+1}^{k+1} \\ \mathbf{P}_{n+1}^{k+1} \end{pmatrix} = \begin{pmatrix} \tilde{\mathbf{F}}_{v^x,n+1} \\ \tilde{\mathbf{F}}_{v^y,n+1} \\ \tilde{\mathbf{F}}_{q,n+1} \end{pmatrix} \tag{A.2}$$

then the matrices involved can be defined as in Table A.1, where matrices $\mathbf{K}_{u^xv^x}$, $\mathbf{K}_{u^yv^y}$, $\mathbf{K}_{u^yv^x}$, $\mathbf{K}_{u^xv^y}$, $\mathbf{K}_{pv^x}$, $\mathbf{K}_{pv^y}$, $\mathbf{K}_{u^xq}$, $\mathbf{K}_{u^yq}$, $\hat{\mathbf{F}}_{v^x,n+1}$ and $\hat{\mathbf{F}}_{v^y,n+1}$ where already defined in the Section 2.2 and the terms added come from the stabilization performed in the VMS framework. $\mathbf{u}_{k,n+1}$ represents the finite element solution from the previous iteration, $\mathbf{u}_n$ the solution from the previous time step, Δt the time step, ρ the density, μ the dynamic viscosity and τ_K the stabilization coefficient defined in (2.34).

Block UV

$$[\tilde{\mathbf{K}}_{u^x v^x}]_{ij} = [\mathbf{K}_{u^x v^x}]_{ij} - \sum_K \tau_K(\frac{\rho^2}{\Delta t}\int_K u^x_{k,n+1}\phi_j \frac{\partial\phi_i}{\partial x} + \rho\mu\int_K \phi_j \boldsymbol{\Delta}\phi_i + \rho^2\int_K (u^x_{k,n+1})^2\,\nabla\phi_j\cdot\nabla\phi_i$$

$$+ \rho\mu\int_K u^x_{k,n+1}\frac{\partial\phi_j}{\partial x}\boldsymbol{\Delta}\phi_i - \rho\mu\int_K u^x_{k,n+1}\frac{\partial\phi_i}{\partial x}\boldsymbol{\Delta}\phi_j - \mu^2\int_K \boldsymbol{\Delta}\phi_i\boldsymbol{\Delta}\phi_j)$$

$$[\tilde{\mathbf{K}}_{u^y v^y}]_{ij} = [\mathbf{K}_{u^y v^y}]_{ij} - \sum_K \tau_K(\frac{\rho^2}{\Delta t}\int_K u^y_{k,n+1}\phi_j \frac{\partial\phi_i}{\partial y} + \rho\mu\int_K \phi_j \boldsymbol{\Delta}\phi_i + \rho^2\int_K (u^y_{k,n+1})^2\,\nabla\phi_j\cdot\nabla\phi_i$$

$$+ \rho\mu\int_K u^y_{k,n+1}\frac{\partial\phi_j}{\partial y}\boldsymbol{\Delta}\phi_i - \rho\mu\int_K u^y_{k,n+1}\frac{\partial\phi_i}{\partial y}\boldsymbol{\Delta}\phi_j - \mu^2\int_K \boldsymbol{\Delta}\phi_i\boldsymbol{\Delta}\phi_j)$$

$$[\tilde{\mathbf{K}}_{u^y v^x}]_{ij} = [\mathbf{K}_{u^y v^x}]_{ij} - \sum_K \tau_K(\frac{\rho^2}{\Delta t}\int_K u^y_{k,n+1}\phi_j \frac{\partial\phi_i}{\partial x} + \rho^2\int_K u^x_{k,n+1}u^y_{k,n+1}\,\nabla\phi_j\cdot\nabla\phi_i$$

$$+ \rho\mu\int_K u^y_{k,n+1}\frac{\partial\phi_j}{\partial x}\boldsymbol{\Delta}\phi_i - \rho\mu\int_K u^x_{k,n+1}\frac{\partial\phi_i}{\partial y}\boldsymbol{\Delta}\phi_j)$$

$$[\tilde{\mathbf{K}}_{u^x v^y}]_{ij} = [\mathbf{K}_{u^x v^y}]_{ij} - \sum_K \tau_K(\frac{\rho^2}{\Delta t}\int_K u^x_{k,n+1}\phi_j \frac{\partial\phi_i}{\partial y} + \rho^2\int_K u^y_{k,n+1}u^x_{k,n+1}\,\nabla\phi_j\cdot\nabla\phi_i$$

$$+ \rho\mu\int_K u^x_{k,n+1}\frac{\partial\phi_j}{\partial y}\boldsymbol{\Delta}\phi_i - \rho\mu\int_K u^y_{k,n+1}\frac{\partial\phi_i}{\partial x}\boldsymbol{\Delta}\phi_j)$$

Block PV

$$[\tilde{\mathbf{K}}_{pv^x}]_{ij} = [\mathbf{K}_{pv^x}]_{ij} - \sum_K \tau_K(\rho\int_K u^x_{k,n+1}\nabla\phi_j\cdot\nabla\phi_j + \mu\int_K \boldsymbol{\Delta}\phi_i\frac{\partial\phi_j}{\partial x})$$

$$[\tilde{\mathbf{K}}_{pv^y}]_{ij} = [\mathbf{K}_{pv^y}]_{ij} - \sum_K \tau_K(\rho\int_K u^y_{k,n+1}\nabla\phi_j\cdot\nabla\phi_j + \mu\int_K \boldsymbol{\Delta}\phi_i\frac{\partial\phi_j}{\partial y})$$

Block UQ

$$[\tilde{\mathbf{K}}_{u^x q}]_{ij} = [\mathbf{K}_{u^x q}]_{ij} - \sum_K \tau_K(\frac{\rho}{\Delta t}\int_K \phi_j\frac{\partial\phi_i}{\partial x} + \rho\int_K u^x_{k,n+1}\nabla\phi_j\cdot\nabla\phi_j - \mu\int_K \boldsymbol{\Delta}\phi_j\frac{\partial\phi_i}{\partial x})$$

$$[\tilde{\mathbf{K}}_{u^y q}]_{ij} = [\mathbf{K}_{u^y q}]_{ij} - \sum_K \tau_K(\frac{\rho}{\Delta t}\int_K \phi_j\frac{\partial\phi_i}{\partial y} + \rho\int_K u^y_{k,n+1}\nabla\phi_j\cdot\nabla\phi_j - \mu\int_K \boldsymbol{\Delta}\phi_j\frac{\partial\phi_i}{\partial y})$$

Block PQ

$$[\tilde{\mathbf{K}}_{pq}]_{ij} = -\sum_K \tau_K\int_K \nabla\phi_j\cdot\nabla\phi_j$$

RHS $\mathbf{F_V}$

$$[\tilde{\mathbf{F}}_{v^x,n+1}]_i = [\hat{\mathbf{F}}_{v^x,n+1}]_i - \sum_K \tau_K(\rho^2\int_K((\mathbf{f} + \frac{\mathbf{u}_n}{\Delta t})\cdot\nabla\phi_i)u^x_{k,n+1} + \rho\mu\int_K(f^x + \frac{u^x_n}{\Delta t})\boldsymbol{\Delta}\phi_i)$$

$$+ \sum_{m=N_{free}}^{N_{nodes}}\sum_K \tau_K(\frac{\rho^2}{\Delta t}\int_K(\mathbf{u}_{k,n+1}\cdot\mathbf{u}_d(\mathbf{p}_m))\phi_m\frac{\partial\phi_i}{\partial x} + \rho\mu\int_K u^x_d(\mathbf{p}_m)\phi_m\boldsymbol{\Delta}\phi_i$$

$$+ \rho^2\int_K(u^x_{k,n+1})^2\,u^x_d(\mathbf{p}_m)\nabla\phi_m\cdot\nabla\phi_i + \rho\mu\int_K(\mathbf{u}_{k,n+1}\cdot\mathbf{u}_d(\mathbf{p}_m))\frac{\partial\phi_m}{\partial x}\boldsymbol{\Delta}\phi_i$$

$$- \mu^2\int_K u^x_d(\mathbf{p}_m)\boldsymbol{\Delta}\phi_i\boldsymbol{\Delta}\phi_m + \rho^2\int_K u^x_{k,n+1}u^y_{k,n+1}\,u^y_d(\mathbf{p}_m)\nabla\phi_m\cdot\nabla\phi_i$$

$$- \rho\mu\int_K u^x_{k,n+1}(\nabla\phi_i\cdot\mathbf{u}_d(\mathbf{p}_m))\boldsymbol{\Delta}\phi_m)$$

$$[\tilde{\mathbf{F}}_{v^y,n+1}]_i = [\hat{\mathbf{F}}_{v^y,n+1}]_i - \sum_K \tau_K(\rho^2\int_K((\mathbf{f} + \frac{\mathbf{u}_n}{\Delta t})\cdot\nabla\phi_i)u^y_{k,n+1} + \rho\mu\int_K(f^y + \frac{u^y_n}{\Delta t})\boldsymbol{\Delta}\phi_i)$$

$$+ \sum_{m=N_{free}}^{N_{nodes}}\sum_K \tau_K(\frac{\rho^2}{\Delta t}\int_K(\mathbf{u}_{k,n+1}\cdot\mathbf{u}_d(\mathbf{p}_m))\phi_m\frac{\partial\phi_i}{\partial y} + \rho\mu\int_K u^y_d(\mathbf{p}_m)\phi_m\boldsymbol{\Delta}\phi_i$$

$$+ \rho^2\int_K(u^y_{k,n+1})^2\,u^y_d(\mathbf{p}_m)\nabla\phi_m\cdot\nabla\phi_i + \rho\mu\int_K(\mathbf{u}_{k,n+1}\cdot\mathbf{u}_d(\mathbf{p}_m))\frac{\partial\phi_m}{\partial y}\boldsymbol{\Delta}\phi_i$$

$$- \rho\mu\int_K u^y_{k,n+1}(\nabla\phi_i\cdot\mathbf{u}_d(\mathbf{p}_m))\boldsymbol{\Delta}\phi_m - \mu^2\int_K u^y_d(\mathbf{p}_m)\boldsymbol{\Delta}\phi_i\boldsymbol{\Delta}\phi_m$$

$+\, \rho^2 \int_K u^y_{k,n+1} u^x_{k,n+1}\, u^x_d(\mathbf{p}_m) \nabla\phi_m \cdot \nabla\phi_i$
RHS $\mathbf{F_Q}$
$[\tilde{\mathbf{F}}_{q,n+1}]_i = -\sum_K \tau_K \rho \int_K (\frac{\mathbf{u}_n}{\Delta t} + \mathbf{f}) \cdot \nabla\phi_i$

Table A.1: Definition of the matrices and vectors required to construct the linear system of equations to be solved at time step t_{n+1} in iteration $k+1$ in the VMS framework.

Bibliography

[1] R. Codina and J. Baiges. Fixed mesh methods in computational mechanics.

[2] R. Codina, G. Houzeaux, H. Coppola-Owen, and J. Baiges. The fixed-mesh ALE approach for the numerical approximation of flows in moving domains. *Journal of Computational Physics*, 228(5):1591–1611, 2009.

[3] E. Stein, R. de Borst, and T. JR. Hughes, editors. *Encyclopedia of computational mechanics*. John Wiley, Chichester, West Sussex, 2004.

[4] Clay mathematics institute. `http://www.claymath.org/millennium-problems/navier%E2%80%93stokes-equation`, 2016. [Online; accessed 1-January-2016].

[5] J. Benk, M. Ulbrich, and M. Mehl. The Nitsche method of the Navier-Stokes equations for immersed and moving boundaries. In *Proceedings of the Seventh International Conference on Computational Fluid Dynamics, ICCFD7. International Conference on Computational Fluid Dynamics*, 2012.

[6] J.M. Urquiza, A. Garon, and M-I. Farinas. Weak imposition of the slip boundary condition on curved boundaries for Stokes flow. *Journal of Computational Physics*, 256:748–767, January 2014.

[7] W. Layton. Weak imposition of no-slip conditions in finite element methods. *Computers & Mathematics with Applications*, 38(56):129–142, September 1999.

[8] J. Serrin. Mathematical Principles of Classical Fluid Mechanics. Fluid Dynamics i. Number 3 in Encyclopedia of Physics, pages 125–263. Springer Berlin Heidelberg, 1959.

[9] L. Mu and X. Ye. A finite volume method for solving Navier Stokes problems. *Nonlinear Analysis: Theory, Methods & Applications*, 74(17):6686–6695, December 2011.

[10] P. Le Tallec. *Modelisation et calcul des milieux continus.* Ecole Polytechnique, Palaiseau, September 2009.

[11] G. Allaire. *Analyse numérique et optimisation.* Ecole Polytechnique, Palaiseau, September 2006.

[12] A. Ern and J-L. Guermond. *Theory and Practice of Finite Elements*. Springer Science & Business Media, March 2013.

[13] D.N. Arnold, F. Brezzi, and M. Fortin. A stable finite element for the Stokes equations. *CALCOLO*, 21(4):337–344, December 1984.

[14] M. Gander and G. Wanner. From Euler, Ritz, and Galerkin to Modern Computing. *SIAM Review*, 54(4):627–666, January 2012.

[15] W. Layton and W. Lenferink. Two-level Picard and modified Picard methods for the Navier-Stokes equations. *Applied Mathematics and Computation*, 69(23):263–274, May 1995.

[16] F-J. Sayas. *A gentle introduction to the Finite Element Method.* March 2008.

[17] M.A. Celia and W.G. Gray. An improved isoparametric transformation for finite element analysis. *International Journal for Numerical Methods in Engineering*, 20(8):1443–1459, August 1984.

[18] L.P. Franca and S.L. Frey. Stabilized finite element methods: II. The incompressible Navier-Stokes equations. *Computer Methods in Applied Mechanics and Engineering*, 99(2):209–233, September 1992.

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

[20] R. Codina. Stabilized finite element approximation of transient incompressible flows using orthogonal subscales. *Computer Methods in Applied Mechanics and Engineering*, 191(3940):4295–4321, August 2002.

[21] A. Masud and R. Calderer. A variational multiscale method for incompressible turbulent flows: Bubble functions and fine scale fields. *Computer Methods in Applied Mechanics and Engineering*, 200(33), 2011.

[22] J. Principe and R. Codina. On the stabilization parameter in the subgrid scale approximation of scalar convectiondiffusionreaction equations on distorted meshes. *Computer Methods in Applied Mechanics and Engineering*, 199(2122):1386–1402, April 2010.

[23] V. P. Nguyen, T. Rabczuk, S. Bordas, and M. Duflot. Meshless methods: A review and computer implementation aspects. *Mathematics and Computers in Simulation*, 79(3):763–813, December 2008.

[24] T. Belytschko, W.K. Liu, and B. Moran. *Nonlinear Finite Elements for Continua and Structures.* Wiley, Chichester, 1 edition, September 2000.

[25] S. Turek and J. Hron. Proposal for Numerical Benchmarking of Fluid-Structure Interaction between an Elastic Object and Laminar Incompressible Flow. In *Fluid-Structure Interaction*, number 53 in Lecture Notes in Computational Science and Engineering, pages 371–385. Springer Berlin Heidelberg, 2006.

[26] H. Luo. Immersed Boundary Method. In *Encyclopedia of Microfluidics and Nanofluidics*, pages 805–808. Springer US, 2008.

[27] C. Peskin. Flow patterns around heart valves: A numerical method. *Journal of Computational Physics*, 10(2):252–271, October 1972.

[28] R. Glowinski, T.-W. Pan, and J. Periaux. A Lagrange multiplier/fictitious domain method for the Dirichlet problem Generalization to some flow problems. *Japan Journal of Industrial and Applied Mathematics*, 12(1):87–108, February 1995.

[29] W. Ritz. Über eine neue Methode zur Lsung gewisser Variationsprobleme der mathematischen Physik. *Journal fr die reine und angewandte Mathematik*, 135:1–61, 1909.

[30] E. Burman. Ghost penalty. *Comptes Rendus Mathématique*, 348(2122):1217–1220, November 2010.

[31] S. Marella, S. Krishnan, H. Liu, and H.S. Udaykumar. Sharp interface Cartesian grid method I: An easily implemented technique for 3d moving boundary computations. *Journal of Computational Physics*, 210(1):1–31, November 2005.

[32] J.W. Barrett and C.M. Elliott. Finite element approximation of the Dirichlet problem using the boundary penalty method. *Numerische Mathematik*, 49(4):343–366, July 1986.

[33] R. Courant. Variational methods for the solution of problems of equilibrium and vibrations. *Bulletin of the American Mathematical Society*, 49(1):1–23, January 1943.

[34] J. Baiges, R. Codina, F. Henke, S. Shahmiri, and W.A. Wall. A symmetric method for weakly imposing Dirichlet boundary conditions in embedded finite element meshes. *International Journal for Numerical Methods in Engineering*, 90(5):636–658, May 2012.

[35] M. Juntunen and R. Stenberg. Nitsche's method for general boundary conditions. *Math. Comput.*, 78(267):1353–1374, 2009.

[36] R. Codina and J. Baiges. Approximate imposition of boundary conditions in immersed boundary methods. *International Journal for Numerical Methods in Engineering*, 80(11):1379 – 1405, 2009.

[37] M.S. Engelman and M-A. Jamnia. Transient flow past a circular cylinder: A benchmark solution. *International Journal for Numerical Methods in Fluids*, 11(7):985–1000, November 1990.

[38] A. Massing, M.G. Larson, A. Logg, and M.E. Rognes. A Stabilized Nitsche Fictitious Domain Method for the Stokes Problem. *Journal of Scientific Computing*, 61(3):604–628, March 2014.

[39] B. Schott, U. Rasthofer, V. Gravemeier, and W.A. Wall. A face-oriented stabilized Nitsche-type extended variational multiscale method for incompressible two-phase flow. *International Journal for Numerical Methods in Engineering*, 104(7):721–748, November 2015.

[40] R. Glowinski, T-W. Pan, and J. Periaux. A fictitious domain method for Dirichlet problem and applications. *Computer Methods in Applied Mechanics and Engineering*, 111(3):283–303, January 1994.

[41] M. Nazem, J.P. Carter, and D.W. Airey. Arbitrary Lagrangian Eulerian method for dynamic analysis of geotechnical problems. *Computers and Geotechnics*, 36(4):549–557, May 2009.

[42] S. Osher and J.A Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations. *Journal of Computational Physics*, 79(1):12–49, November 1988.