The Fixed-Mesh ALE method applied to multiphysics problems using stabilized formulations

Joan Baiges

Advisor: **Ramon Codina**

Escola Tècnica Superior d'Enginyers de Camins, Canals i Ports de Barcelona

Universitat Politècnica de Catalunya

December 2010

ACTA DE QUALIFICACIÓ DE LA TESI DOCTORAL

Reunit el tribunal integrat pels sota signants per jutjar la tesi doctoral:

Títol de la tesi: The Fixed-Mesh ALE method applied to multiphysics problems using stabilized formulations

Autor de la tesi: Joan Baiges

Acorda atorgar la qualificació de:

- ☐ No apte
- ☐ Aprovat
- ☐ Notable
- ☐ Excel·lent
- ☐ Excel·lent Cum Laude

Barcelona, ... de/d’ . . . . de . . . .

El President	El Secretari
. . . . (nom i cognoms)	. . . . (nom i cognoms)

El vocal	El vocal	El vocal
. . . . (nom i cognoms)	. . . . (nom i cognoms)	. . . . (nom i cognoms)

Acknowledgements

I would like to thank my advisor, Ramon, for the opportunity of joining his group and developing my thesis with him. This has been a very enriching period of my life. I would like to particularly acknowledge his always enlightening suggestions and advice, his patience and all the things I have learnt from him.

I would like to thank my work mates Herbert Coppola-Owen, Christian Muñoz, Matías Ávila, Santi Badia, Ramon Planas, Vladimir Jazarevic, Shu-Ren Hysing, Pablo Mata and Fermín Otero for the everyday work in the C1-102 office. Special thanks go to Javier Príncipe for the long time spent together in the development of $FELAP$ and his good attitude towards discussing scientific matters and helping all of his work mates. I would also like to thank Ricardo Rossi and Pooyan Dadvan for introducing me to parallel computations, Carlos Labra for allowing me to take a look to his *spatial containers* library and sharing his deep Fortran knowledge and Daniel Pérez for the time spent together in the development of the seeing around telescopes projects.

Between the beginning of August and the end of October of 2010 Professor Wolfgang A. Wall received me for a stay in the *Lehrstuhl für Numerische Mechanik* in the *Technische Universität München* in Germany, during which Chapter 3 of this thesis was developed. I would like to thank him and his group for their warm reception, especially Florian Henke, Shadan Shahmiri and Johannes Neumayer.

J.M. Aristoff and T.T. Truscott very kindly provided me with the experimental data of their *Water entry of decelerating spheres* experiment. I would like to thank them for this.

I would also like to thank the *Universitat Politècnica de Catalunya*, the *Resistència de Materials i Estructures a l'Enginyeria* department and the *Centre Internacional de Mètodes Numèrics a l'Enginyeria* for the opportunity of working in such prestigious scientific institutions.

The financial support received from the *Agència de Gestió dAjuts Universitaris i de Recerca of the Generalitat de Catalunya* through an FI grant, the *Ministerio de Educación of the Spanish Government* through an FPU grant and the *Col·legi d'Enginyers de Camins de Barcelona* is acknowledged.

I would finally like to specially thank my parents, my sister and Maria. They were the ones that encouraged me to begin this thesis and the ones that supported me all over the process, they were always there when I needed them.

Abstract

The finite element method is a tool very often employed to deal with the numerical simulation of multiphysics problems. Many times each of these problems can be attached to a subdomain in space which evolves in time. Fixed grid methods appear in order to avoid the drawbacks of remeshing in ALE (Arbitrary Lagrangian-Eulerian) methods when the domain undergoes very large deformations. Instead of having one mesh attached to each of the subdomains, one has a single mesh which covers the whole computational domain. Equations arising from the finite element analysis are solved in an Eulerian manner in this background mesh. In this work we present our particular approach to fixed mesh methods, which we call FM-ALE (Fixed-Mesh ALE). Our main concern is to properly account for the advection of information as the domain boundary evolves. To achieve this, we use an arbitrary Lagrangian-Eulerian framework, the distinctive feature being that at each time step results are projected onto a fixed, background mesh, that is where the problem is actually solved. We analyze several possibilities to prescribe boundary conditions in the context of immersed boundary methods.

When dealing with certain physical problems, and depending on the finite element space used, the standard Galerkin finite element method fails and leads to unstable solutions. The variational multiscale method is often used to deal with this instability. We introduce a way to approximate the subgrid scales on the boundaries of the elements in a variational two-scale finite element approximation to flow problems. The key idea is that the subscales on the element boundaries must be such that the transmission conditions for the unknown, split as its finite element contribution and the subscale, hold. We then use the subscales on the element boundaries to improve transmition conditions between subdomains by introducing the subgrid scales between the interfaces in homogeneous domain interaction problems and at the interface between the fluid and the solid in fluid-structure interaction problems. The benefits in each case are respectively a stronger enforcement of the stress continuity in homogeneous domain decomposition problems and a considerable improvement of the behaviour of the iterative algorithm to couple the fluid and the solid in fluid-structure interaction problems.

We develop *FELAP*, a linear systems of equations solver package for problems arising from finite element analysis. The main features of the package are its capability to work with symmetric and unsymmetric systems of equations, direct and iterative solvers and various renumbering techniques. Performance is enhanced by considering the finite element mesh graph instead of the matrix graph, which allows to perform highly efficient block computations.

Contents

Chapter 1

Introduction

The finite element method is a tool very often employed to deal with the numerical simulation of multiphysics problems. Many times each of these problems can be attached to a subdomain in space which evolves in time. In these cases one usually relies in ALE (Arbitrary Lagrangian-Eulerian) formulations. The ALE method consists in giving the finite element mesh an arbitrary movement in such a way that the mesh continues to be a partition of the considered subdomain accounting for its movement in time, and at the same time the shape of the elements which conform the mesh remains as undistorted as possible. Obviously, this procedure introduces some modifications in the computation of the convective terms arising in the problem equations.

ALE methods work very well if the shape and size of each of the subdomains undergoes relatively small changes in time. However, when these changes are large, it is impossible to maintain mesh distortion at a reasonable level, which leads to an ill-conditioning of the systems of equations which arise from the finite element analysis or even to folded elements. In this case the deformed mesh is useless and remedies have to be devised. Classical ALE methods usually deal with this problem by computing a new undistorted mesh which fits with the deformed domain. However, this can be an expensive procedure, especially if it has to be repeated many times during the whole simulation procedure. Moreover, most finite element codes rely the construction of the finite element meshes on external programs, which would imply stopping the execution of the simulation many times, or coding a master program which connects both codes.

Fixed grid methods appear in order to avoid the drawbacks of remeshing in ALE methods. Instead of having one mesh attached to each of the subdomains, one has a single mesh which covers the whole computational domain. Equations arising from the finite element analysis are solved in an Eulerian manner in this background mesh. This obviously avoids remeshing, since the mesh remains undeformed during the whole simulation process, but some other issues appear. Let us consider a Fluid-Structure Interaction problem in which the flow problem is solved by means of a fixed grid method.

- At each time step we have to solve a flow problem with a mesh which does not fit the domain of the flow problem. If the boundary of the domain coincided with the edges or faces of some elements, it would be immediate to consider a submesh covering only the flow domain, but in general the domain boundary will *cut* the elements in an arbi-

trary way. Over which part of the region covered by the mesh do we solve the physical problem?

- Another important issue is the imposition of Dirichlet boundary conditions. In boundary fitting meshes, the imposition of Dirichlet boundary conditions is straightforward, since the boundary of the domain coincides with the edges or faces of the elements of the mesh. This allows to prescribe the values of the unknowns in the nodes of the mesh ubicated on the boundary of the domain. However, this is not possible in fixed grid methods (there are no element edges which define the boundary of the domain). This forces us to devise some alternative strategies to prescribe boundary conditions.

- If we are dealing with a time dependent problem, we face with the need of computing time derivatives. In the Eulerian finite element method used in fixed grid methods, material time derivatives are separated into their local and convective parts. The local time derivative is basically computed as the difference between the value of the unknown at a node in the current time step and the value of the unknown at the same node but in the previous time step. This leads us to the issue of *newly created nodes*: in evolving in time domains, there will be nodes of the fixed mesh which were out of the domain in the previous time step but inside the domain in the current one. How do we compute local time derivatives if we do not know the value of the unknown in the previous time step?

- Finally, it is possible to deal with different physics problems which can be decomposed into different subdomains with a single background mesh. However, in most cases the unknowns fields (and their gradients) will be discontinuous across the interface which separates the various subdomains. Finite element shape functions are in general continuous in the element interiors, so, how are we going to deal with the discontinuity of the unknowns in the boundary of the subdomains?

As we have seen there are four major issues with which a fixed mesh method has to deal with. Any fixed grid method can be classified depending on how it deals with each of these issues. In this work we present our particular approach to fixed mesh methods, which we call FM-ALE (Fixed-Mesh ALE).

In Chapter 2 we analyze several possibilities to strongly prescribe boundary conditions in the context of immersed boundary methods. As starting variational approach we consider Nitsche's method, and we then move to two options that yield non-symmetric problems but that turn out to be robust and efficient. The essential idea is to use the degrees of freedom of certain nodes of the finite element mesh to minimize the difference between the exact and the approximated boundary condition.

In Chapter 3 we propose a way to weakly prescribe Dirichlet boundary conditions in embedded grids. The key feature of the method is that no large penalty parameter is needed and that it is symmetric for symmetric problems. In the Poisson problem this is achieved by introducing an additional element-discontinuous stress variable. Additional terms are required in order to guarantee stability in the convection-diffusion equation and the Stokes problem. The proposed method is then easily extended to the transient Navier-Stokes equations.

In Chapter 4 we propose the FM-ALE method to approximate flow problems in moving domains using always a given grid for the spatial discretization. Our main concern is to properly account for the advection of information as the domain boundary evolves. To achieve this,

we use an arbitrary Lagrangian-Eulerian framework, the distinctive feature being that at each time step results are projected onto a fixed, background mesh, that is where the problem is actually solved.

In Chapter 5 we extend the FM-ALE method to the context of Solid Mechanics and Fluid-Structure Interaction problems. For solid mechanics problems subject to large strains the FM-ALE method avoids the element stretching found in fully Lagrangian approaches. For FSI problems FM-ALE allows for the use of a single background mesh to solve both the fluid and the structure. We also apply the FM-ALE method to the problem of floating solids, in which it is used together with the level set function method.

When dealing with certain physical problems, and depending on the finite element space used, the standard Galerkin finite element method fails and leads to unstable solutions. This is why a great effort has been put during the last decades to develop stabilized formulations which deal with the stability problems of the standard Galerkin method. One of these stabilizing techniques is the subgrid scale method, which is motivated by the decomposition of the continuous solution into a coarse component (finite element solution) and a fine (subgrid) component. In most cases these subscales are considered to vanish on the boundaries of the elements.

In Chapter 6 we introduce a way to approximate the subscales on the boundaries of the elements in a variational two-scale finite element approximation to flow problems. The key idea is that the subscales on the element boundaries must be such that the transmission conditions for the unknown, split as its finite element contribution and the subscale, hold. In particular, we consider the scalar convection-diffusion-reaction equation, the Stokes problem and Darcy's problem. For these problems the transmission conditions are the continuity of the unknown and its fluxes through element boundaries. The former is automatically achieved by introducing a single valued subscale on the boundaries (for the conforming approximations we consider), whereas the latter provides the effective condition for approximating these values. The final result is that the subscale on the interelement boundaries must be proportional to the jump of the flux of the finite element component and the average of the subscale calculated in the element interiors.

In Chapter 7 we use the subscales on the element boundaries to improve transmition conditions between subdomains by introducing the subgrid scales between the interfaces in homogeneous domain interaction problems and at the interface between the fluid and the solid in fluid-structure interaction problems. The benefits in each case are respectively a stronger enforcement of the stress continuity in homogeneous domain decomposition problems and a considerable improvement of the behaviour of the iterative algorithm to couple the fluid and the solid in fluid-structure interaction problems.

When performing numerical simulations with the finite element method, one invariably ends up with the need of solving a linear system of equations. Most finite element codes use linear system solvers developed by other groups and for other purposes. In most cases, this solvers are designed to cope with the most general kind of systems of equations, which means that they do not take advantage of the particularities of the systems of equations arising from the finite element analysis. This is why we aim to develop a solver package especially designed to solve finite element problems.

In Chapter 8 we present *FELAP*, a linear systems of equations solver package for problems arising from finite element analysis. The main features of the package are its capability

to work with symmetric and unsymmetric systems of equations, direct and iterative solvers and various renumbering techniques. Performance is enhanced by considering the finite element mesh graph instead of the matrix graph, which allows to perform highly efficient block computations.

In Chapter 9 we apply the FM-ALE method to solve fluid-structure interaction problems in 3D. We pay special attention to the algorithms needed to compute the mesh-mesh intersections and the subelement integration, which are a bit more complex when extended to 3D. The *FEAP* package for solving linear systems of equations is used. The behaviour of both algorithms is tested in two numerical experiments with satisfactory results.

We close the work with Chapter 10, where conclusions and further possible research lines are summarized. Chapters are quite self contained even if this implies the need of repeating some information. This is due to the fact that each chapter is based on the following publications:

- Chapter 2: ”Approximate imposition of boundary conditions in immersed boundary methods”, R. Codina and J. Baiges, *Int. J. Numer. Meth. Engng*, 80:1379-1405, 2009.
- Chapter 3: ”A symmetric parameter-free method for weakly imposing Dirichlet boundary conditions in embedded grids”, J. Baiges, R.Codina, F. Henke, S. Shahmiri and W.A. Wall, *In preparation*, 2010.
- Chapter 4: ”The Fixed-Mesh ALE method for the numerical approximation of flows in moving domains”, R. Codina, G. Houzeaux, H. Coppola-Owen and J. Baiges, *J. Comput. Phys.*, 228:1591-1611,2009.
- Chapter 5:
  - ”The Fixed-Mesh ALE approach applied to Solid Mechanics and Fluid - Structure Interaction problems”, J. Baiges and R. Codina, *Int. J. Numer. Meth. Engng*, 81:1529-1557, 2010.
  - ”The Fixed-Mesh ALE approach for the numerical simulation of floating solids”, J. Baiges and R. Codina, *Int. J. Numer. Meth. Fluids, Accepted*, 2010.
- Chapter 6: ”Subscales on the element boundaries in the variational two-scale finite element method”, R. Codina, J. Príncipe and J. Baiges, *Computer Methods in Applied Mechanics and Engineering*, 198:838-852, 2009.
- Chapter 7: ”Finite element approximation of transmission conditions in fluids and solids introducing boundary subgrid scales”, R. Codina and J. Baiges, *Submitted*, 2010.
- Chapter 8: ”FELAP Technical Reference Guide”, J. Baiges, J. Príncipe and R. Codina, 2009.

Chapter 2

A non-symmetric method for strongly imposing Dirichlet boundary conditions in embedded grids

In this chapter we analyze several possibilities to prescribe boundary conditions in the context of immersed boundary methods. As basic approximation technique we consider the finite element method with a mesh that does not match the boundary of the computational domain, and therefore Dirichlet boundary conditions need to be prescribed in an approximate way. As starting variational approach we consider Nitsche's method, and we then move to two options that yield non-symmetric problems but that turn out to be robust and efficient. The essential idea is to use the degrees of freedom of certain nodes of the finite element mesh to minimize the difference between the exact and the approximated boundary condition.

2.1 Introduction

The numerical approximation of boundary value problems on non-matching grids has the obvious advantage of the freedom to generate the grid. Only a grid *covering* the computational domain has to be created, leaving the imposition of boundary conditions to the numerical formulation being used. The physical boundary is contained in the domain actually discretized, which is the reason why these methods are called *immersed boundary methods* (IBM).

The price to be paid when using IBM is a lack of control on the grid close to the boundary, which may be very important in flow problems with boundary layers, for example. However, this difficulty may be dealt with using composite grids or Chimera type techniques as that proposed in [72]. Nevertheless, we will not touch this point here, nor the aspect that makes methods with non-matching grids really attractive and that has been our main motivation (see Chapter 4), which is the modelling of flows with moving boundaries keeping the grid fixed. In this case, not only the freedom to generate this grid is important, but also the fact that re-griding as the computational domain evolves may be avoided. This is probably the reason why the so called fixed grid methods have received and are currently receiving a great deal of attention in the numerical literature (see for example the reviews [127, 103, 100]). Since the fixed grid used is often Cartesian, these methods can be found under the keywords *Cartesian*

grid methods.

Our attention will be focused to finite element methods for flow problems, but the ideas to be presented are extendable to other numerical formulations and other physical problems. However, some of the difficulties we shall mention are characteristic of flow problems. Likewise, we will consider general non-structured meshes, the application to Cartesian meshes being obvious. Furthermore, the exposition will be based on 2D linear triangular meshes, although, again, extensions to 3D and other finite element interpolations is straightforward.

Let us describe the problem to be solved. Consider the situation depicted in Fig. 2.1. A domain $\Omega \subset \mathbb{R}^d$, $d = 2, 3$, with boundary $\Gamma = \partial\Omega$ (red curve in Fig. 2.1), is covered by a mesh that occupies a domain $\Omega_h = \Omega_{\text{in}} \cup \Omega_\Gamma$, where $\Omega_{\text{in}} \subset \Omega$ is formed by the elements interior to Ω and Ω_Γ is formed by a set of elements cut by Γ. In turn, let us split $\Omega_\Gamma = \Omega_{\Gamma,\text{in}} \cup \Omega_{\Gamma,\text{out}}$, where $\Omega_{\Gamma,\text{in}} = \Omega \cap \Omega_\Gamma$ and $\Omega_{\Gamma,\text{out}}$ is the interior of $\Omega_\Gamma \setminus \Omega_{\Gamma,\text{in}}$. Note that $\Omega = \Omega_{\text{in}} \cup \Omega_{\Gamma,\text{in}}$. For simplicity, we will assume that the intersection of Γ with the element domains is a piecewise polynomial curve (in 2D) or surface (in 3D) of the same order as the finite element interpolation. This will be used in the proof of stability presented in Subsection 2.3.2, although in fact it is not necessary to apply the method.

Suppose we want to solve a boundary value problem for the unknown u in Ω with the mesh of Ω_h already created and boundary conditions $u = \bar{u}$ on Γ. The obvious choice would be:

- Obtain the nodes of Γ (circles in Fig. 2.1) from the intersection with the element edges.
- Split the elements of $\Omega_{\Gamma,\text{in}}$ so as to obtain a grid matching the boundary Γ.
- Prescribe the boundary condition $u_h = \bar{u}$ in the classical way, where u_h denotes the approximate solution.

This strategy leads to a local remeshing close to Γ that is involved from the computational point of view. Obviously, the implementation of the strategy described is very simple for unstructured simplicial meshes, but it is not so easy if one wants to use other element shapes and, definitely, prevents from using Cartesian meshes. Moreover, if the boundary Γ evolves in time (a situation not considered in the following) the number of degrees of freedom changes at each time instant, thus modifying the structure and sparsivity of the matrix of the final algebraic system. This is clearly an inconvenience even when using unstructured simplicial meshes.

Other possibilities can be found in the literature. One of them is the widely used *Immersed Boundary Method* in its original form [113], which consists in adding point-wise penalty forces in the domain boundary so that the boundary conditions are fulfilled. The method is first order accurate even if second order approximation schemes are used, although *formal second order accuracy* has been reported in [87]. The more recent *Immersed Interface Method* achieves higher order accuracy by avoiding the use of the Dirac delta distribution to define the forcing terms (see [91, 92, 137]).

Another approach is the use of Lagrange multipliers to enforce the boundary conditions. However, the finite-element subspaces for the bulk and Lagrange multiplier fields must satisfy the classical inf-sup condition proposed by Babuška [126], which usually leads to the need for stabilization (see [70, 14, 82]). Moreover, additional degrees of freedom must be added to the problem. The use of Lagrange multipliers is the basis of the *fictitious domain method* [62, 63] (see also Chapter VIII in [61]).

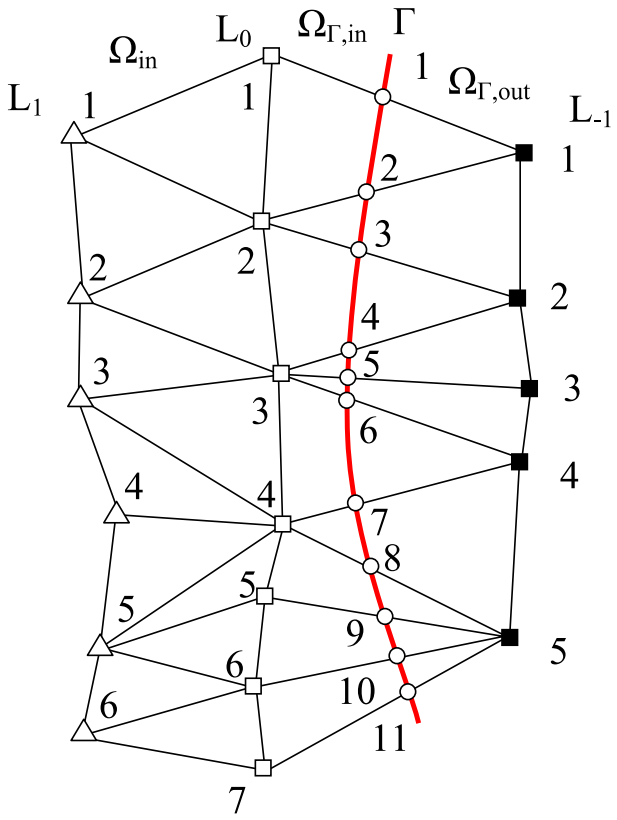


Figure 2.1: Setting

Recently, *hybrid Cartesian/immersed boundary methods* have been developed for Cartesian grids, which use the grid nodes closest to the boundary to enforce boundary conditions [60, 139, 104]. The method is second order accurate, but it does not guarantee that the distance between u_h and $\bar{u}$ in Γ is minimized.

A *discontinuous-Galerkin-based immersed boundary method* is proposed in [93], which consists in switching elements intersected by the boundary to a discontinuous-Galerkin approximation and impose the Dirichlet boundary conditions strongly. Although optimal-order accuracy is achieved, the method requires additional degrees of freedom.

The target we pose is therefore *to impose the Dirichlet boundary conditions (in an approximate way) without adding new degrees of freedom except from those of the original mesh in Ω_h, in such a way that the distance between u_h and $\bar{u}$ in Γ is minimized in a certain norm*. In the following Section we describe Nitsche's method as a first approach to achieve this.

In Section 2.3 we introduce a first modification of Nitsche's method, the main advantage being that there are no parameters to choose and there is no ill-conditioning of the final algebraic system due to large factors enforcing the boundary condition. This is crucial for general flow problems in which there is no rule to choose the parameter appearing in Nitsche's method. The essential idea is to use the degrees of freedom associated with $\Omega_{\Gamma,\mathrm{out}}$ to prescribe approximately the boundary conditions, while the discrete version of the differential operator is only imposed for nodes in Ω_{in}. The drawback is that the problem obtained is not symmetric even for symmetric problems, although the problems we are interested in are non-symmetric. In particular, we have applied the methods to be described to transient incompressible flow problems in moving domains in Chapter 4.

The formulation of Section 2.3 turns out to be accurate, but depending on the way the physical boundary Γ cuts the elements in Ω_h may lead to ill-conditioned matrices and difficulties in the convergence of iterative schemes for nonlinear problems. We present a modification in Section 2.4. In this case, the idea is to solve the problem only in the domain formed by the elements inside Ω, and prescribe the boundary conditions using the degrees of freedom associated to the first layer of nodes *inside* Ω, that is to say, on $\partial\Omega_{\mathrm{in}}$.

Numerical examples showing the performance of the different methods described are presented in Section 2.5, and some concluding remarks close the chapter in Section 2.6.

2.2 Nitsche's method revisited

Our intention is to consider flow problems and, in particular, the scalar convection-diffusion-reaction equation and the incompressible Navier-Stokes equations. However, for the exposition it is enough to consider the former, leaving the latter for the numerical examples.

Let us consider the problem

$$\mathcal{L}u := -k\Delta u + \boldsymbol{a}\cdot\nabla u + su = f \quad \text{in } \Omega, \tag{2.1}$$

$$u = \bar{u} \quad \text{on } \Gamma = \partial\Omega, \tag{2.2}$$

where $k > 0$, $\boldsymbol{a}$ is the advection velocity, $s \geq 0$, f is a given forcing function and $\bar{u}$ is the given Dirichlet boundary condition. We assume that the subdomain Ω is polyhedral, and covered by the domain Ω_h, as explained in Section 2.1.

Let $\mathcal{P}_h = \{K\}$ be a finite element partition of Ω_h from which we construct the finite element space $V_h \subset H^1(\Omega_h)$ (we will consider V_h made of continuous functions). On $V_h \times V_h$ we define the bilinear form

$$B(u_h, v_h) = k(\nabla u_h, \nabla v_h) + (\boldsymbol{a} \cdot \nabla u_h, v_h) + s(u_h, v_h), \tag{2.3}$$

where $(\cdot, \cdot)$ is the L^2-inner product *in* Ω, not in Ω_h. In general, the integral of two functions f_1 and f_2 in a region ω will be denoted by $\langle f_1, f_2 \rangle_\omega$. The norm in a space X will be indicated as $\|\cdot\|_X$, except when $X = L^2(\Omega)$, case in which the subscript will be omitted.

Nitsche's method applied to problem (2.1)-(2.2) reads: find $u_h \in V_h$ such that

$$\begin{aligned} B(u_h, v_h) - k\langle \partial_n u_h, v_h \rangle_\Gamma - k\langle u_h, \partial_n v_h \rangle_\Gamma + \frac{\alpha k^*}{h}\langle u_h, v_h \rangle_\Gamma & \\ = \langle f, v_h \rangle_\Omega - k\langle \bar{u}, \partial_n v_h \rangle_\Gamma + \frac{\alpha k^*}{h}\langle \bar{u}, v_h \rangle_\Gamma \qquad & \forall v_h \in V_h, \end{aligned} \tag{2.4}$$

where $\alpha > 0$ is a numerical parameter, k^* a parameter with the same dimensions as k (here introduced with the only purpose to make the equations dimensionally consistent) and h is the element size, that is to say, $h = \max_K h_K$, with $h_K = \mathrm{diam} K$, $K \in \mathcal{P}_h$. For simplicity, we will consider quasi-uniform partitions $\mathcal{P}_h$.

It is observed that, apart from the way to impose the boundary conditions, (2.4) is based on the standard Galerkin method to solve the convection-diffusion-reaction equation. This method is stable only for high values of the diffusion coefficient k. Even though in the examples we will consider convection dominated flows solved using a stabilized formulation, for the sake of conciseness the exposition will be developed in the diffusion dominated case. Likewise, we will consider $\boldsymbol{a}$ constant, for simplicity.

In the following we will try to "rederive" method (2.4). This will allow us to introduce the modification we propose. Let us consider the splitting $V_h = V_{h,0} \oplus V_{h,\Gamma}$, where $V_{h,0}$ is the subspace of V_h of functions vanishing at the nodes outside Ω_{in}, including its boundary, and $V_{h,\Gamma}$ the complement, that is, the subspace of functions that are zero at the nodes in the interior of Ω_{in}. According to this splitting, we may split the unknown as $u_h = u_{h,0} + u_{h,\Gamma}$ and the test functions as $v_h = v_{h,0} + v_{h,\Gamma}$.

Nitsche's method (2.4) can be obtained from the following set of equations

$$B(u_{h,0}, v_{h,0}) - k\langle \partial_n u_{h,0}, v_{h,0} \rangle_\Gamma + B(u_{h,\Gamma}, v_{h,0}) - k\langle \partial_n u_{h,\Gamma}, v_{h,0} \rangle_\Gamma = \langle f, v_{h,0} \rangle_\Omega, \tag{2.5}$$

$$B(u_{h,0}, v_{h,\Gamma}) - k\langle \partial_n u_{h,0}, v_{h,\Gamma} \rangle_\Gamma + B(u_{h,\Gamma}, v_{h,\Gamma}) - k\langle \partial_n u_{h,\Gamma}, v_{h,\Gamma} \rangle_\Gamma = \langle f, v_{h,\Gamma} \rangle_\Omega, \tag{2.6}$$

$$- k\langle \partial_n v_{h,0}, u_{h,0} \rangle_\Gamma - k\langle \partial_n v_{h,0}, u_{h,\Gamma} \rangle_\Gamma = -k\langle \partial_n v_{h,0}, \bar{u} \rangle_\Gamma, \tag{2.7}$$

$$- k\langle \partial_n v_{h,\Gamma}, u_{h,0} \rangle_\Gamma - k\langle \partial_n v_{h,\Gamma}, u_{h,\Gamma} \rangle_\Gamma = -k\langle \partial_n v_{h,\Gamma}, \bar{u} \rangle_\Gamma, \tag{2.8}$$

$$\frac{\alpha k^*}{h}\langle u_{h,0}, v_{h,0} \rangle_\Gamma + \frac{\alpha k^*}{h}\langle u_{h,\Gamma}, v_{h,0} \rangle_\Gamma = \frac{\alpha k^*}{h}\langle \bar{u}, v_{h,0} \rangle_\Gamma, \tag{2.9}$$

$$\frac{\alpha k^*}{h}\langle u_{h,0}, v_{h,\Gamma} \rangle_\Gamma + \frac{\alpha k^*}{h}\langle u_{h,\Gamma}, v_{h,\Gamma} \rangle_\Gamma = \frac{\alpha k^*}{h}\langle \bar{u}, v_{h,\Gamma} \rangle_\Gamma. \tag{2.10}$$

The first two equations (2.5)-(2.6) are obtained by multiplying the differential equation by $v_{h,0}$ and $v_{h,\Gamma}$ and integrating by parts. Note that no boundary conditions are imposed, and thus the solution of (2.5)-(2.6) is not unique. Equations (2.7)-(2.8) can be understood as a weak form

of the boundary condition $u_h = \bar{u}$, weighting this equation by $-k\partial_n v_{h,0}$ and $-k\partial_n v_{h,\Gamma}$. These two equations are needed to keep the symmetry of the problem when $B(u_h, v_h) = B(v_h, u_h)$, that is to say, when $\boldsymbol{a} = \mathbf{0}$ (see (2.3)). Finally, equations (2.9)-(2.10) are also obtained as a weak form of the boundary condition $u_h = \bar{u}$, weighting now this equation by $v_{h,0}$ and $v_{h,\Gamma}$.

Obviously, equations (2.5)-(2.10) are all consistent, in the sense that if u_h is replaced by the exact solution u of problem (2.1)-(2.2) they hold exactly, provided this solution is regular enough. However, system (2.5)-(2.10) as a whole *is overdetermined*, and there are several possibilities to extract a system of algebraic equations with a unique solution from it. In particular, Nitsche's method (2.4) is obtained by adding together all the equations. The method to be proposed in the following section can be understood as the method obtained *keeping only (2.5) and (2.10)*. In fact, for stability reasons described later it turns out to be convenient to *subtract* (2.7) from (2.5).

Before describing an alternative to Nitsche's method, let us comment on the role played by the factor $\frac{\alpha k^*}{h}$. Suppose that $\boldsymbol{a} = \mathbf{0}$, so that B is symmetric, and define the functionals $J_1(u_{h,0}, u_{h,\Gamma}) = \frac{1}{2}B(u_{h,0} + u_{h,\Gamma}, u_{h,0} + u_{h,\Gamma}) - k\langle\partial_n(u_{h,0} + u_{h,\Gamma}), (u_{h,0} + u_{h,\Gamma})\rangle_\Gamma - \langle f, u_{h,0} + u_{h,\Gamma}\rangle_\Omega - k\langle\bar{u}, \partial_n(u_{h,0} + u_{h,\Gamma})\rangle_\Gamma$ and $J_2(u_{h,0}, u_{h,\Gamma}) = \frac{\alpha k^*}{h}\|u_{h,0} + u_{h,\Gamma} - \bar{u}\|^2_{L^2(\Gamma)}$. If $\delta_{(v_{h,0}, v_{h,\Gamma})}$ denotes the weak (Gâteaux) derivative of a functional in the direction of $v_h = (v_{h,0}, v_{h,\Gamma})$ we may write problem (2.4) as

$$\delta_{(v_{h,0}, v_{h,\Gamma})}(J_1(u_{h,0}, u_{h,\Gamma}) + J_2(u_{h,0}, u_{h,\Gamma})) = 0. \tag{2.11}$$

From this expression it follows that satisfying the Dirichlet boundary condition must *compete* with satisfying the differential equation, $\frac{\alpha k^*}{h}$ being the weight of the former. Moreover, since the norm $h^{-1/2}\|\cdot\|_{L^2(\Gamma)}$ is equivalent to the norm of $\|\cdot\|_{H^{1/2}(\Gamma)}$ in V_h (see [22, 48]), the relevant weighting is in fact the parameter α. The higher the value of α, the better the approximation to the boundary condition at the expense of a poorer approximation to the differential equation. However, it is possible to show that the method is stable and optimally convergent for a suitable value of α (in fact, stability is even easier to show than for the method to be presented in the following section). See [83] for a proof, including more general boundary conditions than used here (although for Poisson's problem). The good performance of Nitsche's method has been exploited also in other contexts, such as the imposition of boundary conditions for discontinuous finite element approximations (see the original work in [5] and the extension in [65], for example), the imposition of transmission conditions in domain decomposition with non-matching grids (as in [16, 64], among many others) or also in some stabilized finite element methods for which this method fits nicely [25].

Finally, let us remark that the volume integrals in (2.5)-(2.6) are performed over $\Omega = \Omega_{\text{in}} \cup \Omega_{\Gamma,\text{in}}$. Integrals over Ω_{in} are easily computed, but in order to compute integrals over $\Omega_{\Gamma,\text{in}}$ some care is needed. The simplest approach is to split the elements of $\Omega_{\Gamma,\text{in}}$ so as to obtain a grid matching the boundary Γ, and then proceed to compute the integrals over the resulting subelements (see [41]). Note that this splitting does not affect the degrees of freedom of the problem.

2.3 A first modification: using external degrees of freedom

2.3.1 The first method proposed

The essential idea of the method we propose first is to use only equations (2.5) and (2.10) above. As it has been mentioned in the previous section, it is also convenient, mainly for the stability analysis, to subtract (2.7) from (2.5). Thus, the problem to be solved is: find $u_{h,0} \in V_{h,0}$ and $u_{h,\Gamma} \in V_{h,\Gamma}$ such that

$$\begin{aligned} B(u_{h,0}, v_{h,0}) + B(u_{h,\Gamma}, v_{h,0}) - k\langle \partial_n u_{h,0}, v_{h,0}\rangle_\Gamma - k\langle \partial_n u_{h,\Gamma}, v_{h,0}\rangle_\Gamma \\ + k\langle \partial_n v_{h,0}, u_{h,0}\rangle_\Gamma + k\langle \partial_n v_{h,0}, u_{h,\Gamma}\rangle_\Gamma = \langle f, v_{h,0}\rangle_\Omega + k\langle \partial_n v_{h,0}, \bar{u}\rangle_\Gamma, \end{aligned} \tag{2.12}$$

$$\frac{\alpha k^*}{h}\langle u_{h,0}, v_{h,\Gamma}\rangle_\Gamma + \frac{\alpha k^*}{h}\langle u_{h,\Gamma}, v_{h,\Gamma}\rangle_\Gamma = \frac{\alpha k^*}{h}\langle \bar{u}, v_{h,\Gamma}\rangle_\Gamma, \tag{2.13}$$

for all $v_{h,0} \in V_{h,0}$ and $v_{h,\Gamma} \in V_{h,\Gamma}$.

Equation (2.13) can be equivalently written as

$$\delta_{(0,v_{h,\Gamma})} J_2(u_{h,0}, u_{h,\Gamma}) = 0. \tag{2.14}$$

From this equation it is clear that *the component $u_{h,\Gamma}$ of the unknown is determined from the condition that the distance between $u_{h,0} + u_{h,\Gamma}$ and $\bar{u}$ is minimized in the norm of $L^2(\Gamma)$*. Comparing this equation with (2.11), it is also seen that now this minimization does not compete with the satisfaction of the differential equation (in weak sense). Obviously, the parameter $\frac{\alpha k^*}{h}$ is here unnecessary, and it has been introduced only to compare the resulting method with (2.4).

Let us enumerate four major differences of (2.12)-(2.13) with respect to (2.4):

1. When Γ coincides with $\partial\Omega_h$, the boundary condition is imposed exactly (provided $\bar{u}$ is a finite element function).
2. There are no parameters to be tuned ($\frac{\alpha k^*}{h}$ can be canceled out in (2.13)).
3. The method is non-symmetric, even if B is symmetric.
4. The method is not well defined when Γ coincides with $\partial\Omega_{\text{in}}$.

The first two points are improvements with respect to Nitsche's method. In particular, they explain why the approximation of boundary conditions is in general better with our approach, as we have experimented from numerical tests. The third point is a drawback from the implementation point of view only for symmetric problems, and not for the flow problems we are interested in. The important issue is point 4. Clearly, when $\Gamma = \partial\Omega_{\text{in}}$ (2.13) yields $0 = 0$. In this case, elements outside Ω_{in} could be eliminated and the case reduced to the first one. However, this situation may be encountered if Ω_{in} is a domain moving in time inside Ω_h. In general, when Γ is close to $\partial\Omega_{\text{in}}$ we may expect instability problems. Small variations in $\bar{u}$ may yield large variations in $u_{h,\Gamma}$. This fact will be used to motivate the method proposed in Section 2.4. It is worth to mention that this type of instabilities are also encountered in other methods for which modifications are also required (see [139, 119]). We will come back to this point in Section 2.4.

Precise conditions under which the method is stable are discussed in the following subsection.

2.3.2 Stability

In this subsection we prove the following result: *if* Γ *is kept away from* $\partial\Omega_{\text{in}}$ *the formulation given by (2.12)-(2.13) is stable*. As a consequence, the discrete problem admits a unique solution.

Proving this fact requires some analytical technicalities that will make us depart from the line of formulating new methods rather than analyzing them. However, we believe this conclusion is important and deserves this parenthesis in the main syllabus of the chapter.

Preliminary result

We will make use of a general result applicable to coupled systems of variational equations of the form

$$a_{11}(u_1, v_1) + a_{12}(u_2, v_1) = l_1(v_1), \tag{2.15}$$
$$a_{21}(u_1, v_2) + a_{22}(u_2, v_2) = l_2(v_2), \tag{2.16}$$

where $u_1, v_1 \in V_1$, $u_2, v_2 \in V_2$, a_{ij} is a bilinear form defined on $V_j \times V_i$ and l_i a linear form on V_i, a Banach space with norm $\|\cdot\|_i$, $i, j = 1, 2$. We assume that all the forms a_{ij} are continuous and a_{ii} are coercive. Let C_{ij} be the constants defined by the inequalities

$$\begin{aligned} a_{11}(v_1, v_1) &\geq C_{11}\|v_1\|_1^2, & a_{22}(v_2, v_2) &\geq C_{22}\|v_2\|_2^2, \\ a_{12}(v_2, v_1) &\leq C_{12}\|v_1\|_1 \|v_2\|_2, & a_{21}(v_1, v_2) &\leq C_{21}\|v_1\|_1 \|v_2\|_2. \end{aligned}$$

We will now prove that if

$$C_{12}C_{21} < C_{11}C_{22}, \tag{2.17}$$

then there exists a constant $C > 0$ such that for all $(u_1, u_2) \in V_1 \times V_2$ there exists $(v_1, v_2) \in V_1 \times V_2$ such that

$$\begin{aligned} B((u_1, u_2), (v_1, v_2)) &:= a_{11}(u_1, v_1) + a_{12}(u_2, v_1) + a_{21}(u_1, v_2) + a_{22}(u_2, v_2) \\ &\geq C\left(\|u_1\|_1 + \|u_2\|_2\right)\left(\|v_1\|_1 + \|v_2\|_2\right), \end{aligned}$$

that is to say, *problem (2.15)-(2.16) is stable*.

In the following, C will denote a generic positive constant, not necessarily the same at different appearances. In the case in which (2.15)-(2.16) comes from a finite element approximation, the constant C will be independent of h and inequality (2.17) will be assumed to hold uniformly in h.

Let us start noting that using Young's inequality we have

$$B((u_1, u_2), (u_1, 0)) \geq C_{11}\|u_1\|_1^2 - C_{12}\left(\frac{\beta_1}{2}\|u_1\|_1^2 + \frac{1}{2\beta_1}\|u_2\|_2^2\right),$$
$$B((u_1, u_2), (0, u_2)) \geq C_{22}\|u_2\|_2^2 - C_{21}\left(\frac{\beta_2}{2}\|u_2\|_2^2 + \frac{1}{2\beta_2}\|u_1\|_1^2\right),$$

where β_1 and β_2 are positive constants to be determined. Thus, for any $\gamma > 0$ we have

$$B((u_1,u_2),(u_1,\gamma u_2)) \geq \left(C_{11} - C_{12}\frac{\beta_1}{2} - C_{21}\frac{1}{2\beta_2}\right)\|u_1\|_1^2 + \left(C_{22}\gamma - C_{21}\frac{\beta_2}{2}\gamma^2 - C_{12}\frac{1}{2\beta_1}\right)\|u_2\|_2^2.$$

The constants C_{12} and C_{21} must be such that there exists β_1, β_2 and γ for which

$$C_{11} - C_{12}\frac{\beta_1}{2} - C_{21}\frac{1}{2\beta_2} > 0, \tag{2.18}$$

$$C_{22}\gamma - C_{21}\frac{\beta_2}{2}\gamma^2 - C_{12}\frac{1}{2\beta_1} > 0. \tag{2.19}$$

Condition (2.18) holds if

$$\beta_1 < \frac{2C_{11}\alpha_1}{C_{12}}, \quad \frac{1}{\beta_2} < \frac{2C_{11}\alpha_2}{C_{21}}, \quad \alpha_1+\alpha_2 = 1.$$

Condition (2.19) requires then that

$$\gamma > A\gamma^2 + B, \quad A := \frac{C_{21}^2}{4C_{11}C_{22}\alpha_2}, \quad B := \frac{C_{12}^2}{4C_{11}C_{22}\alpha_1},$$

a condition that is possible to fulfill if

$$AB < \frac{1}{4} \iff C_{12}C_{21} < 2C_{11}C_{22}\sqrt{\alpha_1\alpha_2}.$$

Since $\alpha_1+\alpha_2 = 1$, the maximum of $\sqrt{\alpha_1\alpha_2}$ is $1/2$, from where the result follows.

Some useful relationships

The next step is to prove some inequalities that will be used later on. These inequalities make use of the inverse estimates (see [22, 48]):

$$\|v_h\|_{L^\infty(\omega)}^2 \leq \frac{C}{h^d}\|v_h\|_{L^2(\omega)}^2, \tag{2.20}$$

$$\|\nabla v_h\|_{L^2(\omega)}^2 \leq \frac{C}{h^2}\|v_h\|_{L^2(\omega)}^2, \tag{2.21}$$

where ω is any patch of elements of $\mathcal{P}_h$ (recall that this partition is assumed to be quasi-uniform) and v_h is a finite element function. Because of the assumption on the shape of Γ, ω can be also formed by subdomains of the form $K \cap \Omega_{\Gamma,\text{in}}$, $K \in \mathcal{P}_h$.

From these inequalities one can prove the following:

$$\|v_{h,\Gamma}\|^2 \leq C\delta_1 h\|v_{h,\Gamma}\|_{L^2(\Gamma)}^2, \tag{2.22}$$

$$\|\nabla v_{h,\Gamma}\|^2 \leq C\frac{1}{\delta_1 h}\|v_{h,\Gamma}\|_{L^2(\Gamma)}^2, \tag{2.23}$$

$$\frac{\delta_1}{\delta_2^2 h}\|v_{h,0}\|_{L^2(\Gamma)}^2 \leq C\|\nabla v_{h,0}\|_{L^2(\Omega_{\Gamma,\text{in}})}^2, \tag{2.24}$$

$$\|\partial_n v_h\|_{L^2(\Gamma)}^2 \leq \frac{C}{\delta_1 h}\|\nabla v_h\|_{L^2(\Omega_{\Gamma,\text{in}})}^2, \tag{2.25}$$

where (see Fig. 2.1)

$$\delta_1 = \frac{1}{h}\min_{\boldsymbol{x}\in L_0} \text{dist}(\boldsymbol{x},\Gamma), \quad \delta_2 = \frac{1}{h}\max_{\boldsymbol{x}\in L_{-1}} \text{dist}(\boldsymbol{x},\Gamma).$$

Let us start noting that (2.22) is a direct consequence of the shape of $\Omega_{\Gamma,\text{in}}$ and that $v_{h,\Gamma}$ vanishes at the nodes in the interior of this subdomain. The distance from Γ to the nodes of L_0 can be bounded by $C\delta_1 h$, where $1 \leq C$ is a constant which will be bounded as $h \to 0$ because of the quasi-uniformity of the partition.

The proof of (2.23) is as follows:

$$\begin{aligned}
\|\nabla v_{h,\Gamma}\|^2 &= \int_{\Omega_{\Gamma,\text{in}}} |\nabla v_{h,\Gamma}|^2 && (v_{h,\Gamma} \text{ is zero elsewhere})\\
&\leq \frac{C}{\delta_1^2 h^2}\int_{\Omega_{\Gamma,\text{in}}} |v_{h,\Gamma}|^2 && (\text{by (2.21)})\\
&\leq \frac{C}{\delta_1 h}\int_{\Gamma} |v_{h,\Gamma}|^2 && (\text{by (2.22)})\\
&= \frac{C}{\delta_1 h}\|v_{h,\Gamma}\|^2_{L^2(\Gamma)}.
\end{aligned}$$

For the proof of (2.24), let K be an element crossed by Γ and $E = K \cap \Gamma$. We have:

$$\begin{aligned}
\int_E v_{h,0}^2 &\leq \delta_2^2 h^2 \int_E \|\nabla v_{h,0}\|^2_{L^\infty(K)}\\
&\leq C\delta_2^2 h^2 h^{d-1}\|\nabla v_{h,0}\|^2_{L^\infty(K)}\\
&\leq C\frac{\delta_2^2}{\delta_1} h^{d+1} h^{-d}\|\nabla v_{h,0}\|^2_{L^2(K\cap\Omega_{\Gamma,\text{in}})}, && (\text{by (2.20)})
\end{aligned}$$

from where (2.24) is obtained from summation over all E that form Γ. Finally, (2.25) follows again from the shape of $\Omega_{\Gamma,\text{in}}$.

Application to the first method proposed

Finally, we will apply (2.22)-(2.25) to show that condition (2.17) holds, and thus the method given by (2.12)-(2.13) is stable. Let us define the bilinear forms

$$\begin{aligned}
a_{0,0}(u_{h,0}, v_{h,0}) &:= B(u_{h,0}, v_{h,0}) - k\langle\partial_n u_{h,0}, v_{h,0}\rangle_\Gamma + k\langle\partial_n v_{h,0}, u_{h,0}\rangle_\Gamma,\\
a_{0,\Gamma}(u_{h,\Gamma}, v_{h,0}) &:= B(u_{h,\Gamma}, v_{h,0}) - k\langle\partial_n u_{h,\Gamma}, v_{h,0}\rangle_\Gamma + k\langle\partial_n v_{h,0}, u_{h,\Gamma}\rangle_\Gamma,\\
a_{\Gamma,0}(u_{h,0}, v_{h,\Gamma}) &:= \langle u_{h,0}, u_{h,\Gamma}\rangle_\Gamma,\\
a_{\Gamma,\Gamma}(u_{h,\Gamma}, v_{h,\Gamma}) &:= \langle u_{h,\Gamma}, u_{h,\Gamma}\rangle_\Gamma,
\end{aligned}$$

and the norms

$$|\!|\!| v_h |\!|\!|_0^2 := k\|\nabla v_h\|^2 + s\|v_{h,0}\|^2, \quad |\!|\!| v_h |\!|\!|_\Gamma := \|v_h\|_{L^2(\Gamma)}.$$

As it has been mentioned in Section 2.2, we assume that the problem is diffusion dominated. More precisely, if $a = |\boldsymbol{a}|$, in what follows we assume that h is such that

$$k - C\frac{ah}{2}\frac{\delta_2^2}{\delta_1} \geq C_k k, \quad 0 < C_k < 1, \tag{2.26}$$

for a constant C introduced next.

We have to check (2.17), and therefore we need to estimate the coercivity constants of $a_{0,0}$ and $a_{\Gamma,\Gamma}$ and the continuity constants of $a_{0,\Gamma}$ and $a_{\Gamma,\Gamma}$. We have

$$\begin{aligned}
a_{0,0}(u_{h,0}, u_{h,0}) &= B(u_{h,0}, u_{h,0}) \\
&= k\|\nabla u_{h,0}\|^2 + s\|u_{h,0}\|^2 + (\boldsymbol{a}\cdot\nabla u_{h,0}, u_{h,0}) \\
&= k\|\nabla u_{h,0}\|^2 + s\|u_{h,0}\|^2 + \int_\Gamma \boldsymbol{n}\cdot\boldsymbol{a}\frac{1}{2}u_{h,0}^2 \\
&\geq k\|\nabla u_{h,0}\|^2 + s\|u_{h,0}\|^2 - \frac{a}{2}\|u_{h,0}\|^2_{L^2(\Gamma)} \\
&\geq k\|\nabla u_{h,0}\|^2 + s\|u_{h,0}\|^2 - C\frac{ah}{2}\frac{\delta_2^2}{\delta_1}\|\nabla u_{h,0}\|^2 \qquad \text{(by (2.24))} \\
&\geq C_k |\!|\!| u_{h,0} |\!|\!|_0^2,
\end{aligned}$$

and therefore the coercivity constant of $a_{0,0}$ may be taken as

$$C_{0,0} = C_k.$$

On the other hand, we have

$$a_{\Gamma,\Gamma}(u_{h,\Gamma}, u_{h,\Gamma}) = \|u_{h,\Gamma}\|^2_{L^2(\Gamma)} = |\!|\!| u_{h,\Gamma} |\!|\!|_\Gamma^2,$$

and hence

$$C_{\Gamma,\Gamma} = 1.$$

The continuity constant of $a_{0,\Gamma}$ is obtained from the following bounding process:

$$\begin{aligned}
a_{0,\Gamma}(u_{h,\Gamma}, v_{h,0}) &= k(\nabla u_{h,\Gamma}, \nabla v_{h,0}) + \langle \boldsymbol{a}\cdot\boldsymbol{n} u_{h,\Gamma}, v_{h,0}\rangle_\Gamma \\
&\quad - (u_{h,\Gamma}, \boldsymbol{a}\cdot\nabla v_{h,0}) + s(u_{h,\Gamma}, v_{h,0}) \\
&\quad - k\langle \partial_n u_{h,\Gamma}, v_{h,0}\rangle_\Gamma + k\langle \partial_n v_{h,0}, u_{h,\Gamma}\rangle_\Gamma \\
&\leq k\|\nabla u_{h,\Gamma}\|\,\|\nabla v_{h,0}\| + a\|u_{h,\Gamma}\|_{L^2(\Gamma)}\|v_{h,0}\|_{L^2(\Gamma)} \\
&\quad + a\|u_{h,\Gamma}\|\,\|\nabla v_{h,0}\| + s\|u_{h,\Gamma}\|\,\|v_{h,0}\| \\
&\quad + k\|\partial_n u_{h,\Gamma}\|_{L^2(\Gamma)}\|v_{h,0}\|_{L^2(\Gamma)} + k\|\partial_n v_{h,0}\|_{L^2(\Gamma)}\|u_{h,\Gamma}\|_{L^2(\Gamma)}
\end{aligned}$$

$$
\begin{aligned}
&\leq k\frac{C}{\delta_1^{1/2}h^{1/2}}\|u_{h,\Gamma}\|_{L^2(\Gamma)}\|\nabla v_{h,0}\| && \text{(by (2.23))}\\
&+ a\|u_{h,\Gamma}\|_{L^2(\Gamma)}\frac{Ch^{1/2}\delta_2}{\delta_1^{1/2}}\|\nabla v_{h,0}\| && \text{(by (2.24))}\\
&+ aC\delta_1^{1/2}h^{1/2}\|u_{h,\Gamma}\|_{L^2(\Gamma)}\|\nabla v_{h,0}\| && \text{(by (2.22))}\\
&+ sC\delta_1^{1/2}h^{1/2}\|u_{h,\Gamma}\|_{L^2(\Gamma)}\|v_{h,0}\| && \text{(by (2.22))}\\
&+ k\frac{C}{\delta_1 h}\|u_{h,\Gamma}\|_{L^2(\Gamma)}\frac{\delta_2 h^{1/2}}{\delta_1^{1/2}}\|\nabla v_{h,0}\| && \text{(by (2.23)-(2.25))}\\
&+ k\frac{C}{\delta_1^{1/2}h^{1/2}}\|\nabla v_{h,0}\|\,\|u_{h,\Gamma}\|_{L^2(\Gamma)} && \text{(by (2.25)).}
\end{aligned}
$$

This inequality can be written as

$$
a_{0,\Gamma}(u_{h,\Gamma},v_{h,0}) \leq CK\left(k\|\nabla v_h\|^2 + s\|v_{h,0}\|^2\right)^{1/2}\|u_h\|_{L^2(\Gamma)}, \tag{2.27}
$$

with

$$
K := \frac{k^{1/2}}{\delta_1^{1/2}h^{1/2}} + \frac{ah\delta_2}{k^{1/2}h^{1/2}\delta_1^{1/2}} + \frac{ah\delta_1^{1/2}}{k^{1/2}h^{1/2}} + \delta_1^{1/2}h^{1/2}s^{1/2} + \frac{k^{1/2}\delta_2}{\delta_1^{3/2}h^{1/2}} + \frac{k^{1/2}}{\delta_1^{1/2}h^{1/2}}.
$$

Using (2.26) and the fact that $0 < \delta_1, \delta_2 < 1$, from (2.27) we see that we may take the continuity constant of $a_{0,\Gamma}$ as

$$
C_{0,\Gamma} = C\frac{k^{1/2}}{h^{1/2}\delta_1^{3/2}}\left(1 + \frac{s^{1/2}h}{k^{1/2}}\right).
$$

The bound for $a_{\Gamma,0}$ is easily obtained using (2.24):

$$
\begin{aligned}
a_{\Gamma,0}(u_{h,0},v_{h,\Gamma}) &\leq \|u_{h,0}\|_{L^2(\Gamma)}\|v_{h,\Gamma}\|_{L^2(\Gamma)}\\
&\leq \frac{C\delta_2 h^{1/2}}{\delta_1^{1/2}}\|v_{h,\Gamma}\|_{L^2(\Gamma)}\|\nabla u_{h,0}\|,
\end{aligned}
$$

from where

$$
C_{\Gamma,0} = C\frac{\delta_2 h^{1/2}}{\delta_1^{1/2}k^{1/2}}.
$$

We are now in a position to check condition (2.17) in our case, which reads:

$$
C_{0,\Gamma}C_{\Gamma,0} = C\frac{\delta_2}{\delta_1^2}\left(1 + \frac{s^{1/2}h}{k^{1/2}}\right) < C_{0,0}C_{\Gamma,\Gamma} = C_k. \tag{2.28}
$$

This inequality is satisfied *provided δ_2 is small enough*, that is to say, Γ is sufficiently close to $\partial\Omega_h$. This is the result we wanted to prove, and which allows us to guarantee that problem (2.12)-(2.13) is well posed in this situation.

In passing, condition (2.28) allows us to observe how stability deteriorates in terms of δ_1, and also how the rate between reaction and diffusion effects, measured by sh^2/k, affects stability.

2.3.3 Implementation aspects

The purpose of this subsection is to express in matrix form problem (2.12)-(2.13) and to discuss some implementation aspects.

Suppose that the unknown u_h is interpolated as

$$\begin{aligned} u_h(\boldsymbol{x}) &= \sum_{a=1}^{n_{\text{in}}} I^a_{\text{in}}(\boldsymbol{x}) U^a_{\text{in}} + \sum_{b=1}^{n_{\text{out}}} I^b_{\text{out}}(\boldsymbol{x}) U^b_{\text{out}} \\ &= \boldsymbol{I}_{\text{in}}(\boldsymbol{x}) \boldsymbol{U}_{\text{in}} + \boldsymbol{I}_{\text{out}}(\boldsymbol{x}) \boldsymbol{U}_{\text{out}}, \end{aligned}$$

where $I^a_{\text{in}}(\boldsymbol{x})$ and $I^b_{\text{out}}(\boldsymbol{x})$ are the standard interpolation functions, n_{in} is the number of nodes in Ω_{in} (including layer L_0) and n_{out} the number of nodes in layer L_{-1} (see Fig. 2.1).

The objective is to compute $\boldsymbol{U}_{\text{out}}$. As it has been shown, (2.13) is equivalent to the minimization problem (2.14), that is to say, $\boldsymbol{U}_{\text{out}}$ can be computed by minimizing the functional

$$J_2(\boldsymbol{U}_{\text{in}}, \boldsymbol{U}_{\text{out}}) = \int_\Gamma \left(u_h(\boldsymbol{x}) - \bar{u}(\boldsymbol{x})\right)^2 = \int_\Gamma \left(\boldsymbol{I}_{\text{in}}(\boldsymbol{x}) \boldsymbol{U}_{\text{in}} + \boldsymbol{I}_{\text{out}}(\boldsymbol{x}) \boldsymbol{U}_{\text{out}} - \bar{u}(\boldsymbol{x})\right)^2$$

Obviously, other options would be possible. In the case we consider,

$$\frac{\partial J_2}{\partial \boldsymbol{U}_{\text{out}}} = 0 \;\Rightarrow\; \boldsymbol{M}_\Gamma \boldsymbol{U}_{\text{out}} = \boldsymbol{f}_\Gamma - \boldsymbol{N}_\Gamma \boldsymbol{U}_{\text{in}}, \tag{2.29}$$

where

$$\boldsymbol{M}_\Gamma = \int_\Gamma \boldsymbol{I}^{\text{t}}_{\text{out}}(\boldsymbol{x}) \boldsymbol{I}_{\text{out}}(\boldsymbol{x}), \quad \boldsymbol{f}_\Gamma = \int_\Gamma \boldsymbol{I}^{\text{t}}_{\text{out}}(\boldsymbol{x}) \bar{u}(\boldsymbol{x}), \quad \boldsymbol{N}_\Gamma = \int_\Gamma \boldsymbol{I}^{\text{t}}_{\text{out}}(\boldsymbol{x}) \boldsymbol{I}_{\text{in}}(\boldsymbol{x}).$$

Suppose the matrix form of (2.12) is

$$\boldsymbol{K}_{\text{in,in}} \boldsymbol{U}_{\text{in}} + \boldsymbol{K}_{\text{in,out}} \boldsymbol{U}_{\text{out}} = \boldsymbol{F}_{\text{in}}. \tag{2.30}$$

The domain integrals in matrices $\boldsymbol{K}_{\text{in,in}}$ and $\boldsymbol{K}_{\text{in,out}}$ extend only over Ω_{in}. The nodal values $\boldsymbol{U}_{\text{out}}$ are merely used as degrees of freedom to interpolate u_h in the subdomain Ω_{in}. Inserting (2.29) into (2.30) results in

$$\left(\boldsymbol{K}_{\text{in,in}} - \boldsymbol{K}_{\text{in,out}} \boldsymbol{M}_\Gamma^{-1} \boldsymbol{N}_\Gamma\right) \boldsymbol{U}_{\text{in}} = \boldsymbol{F}_{\text{in}} - \boldsymbol{K}_{\text{in,out}} \boldsymbol{M}_\Gamma^{-1} \boldsymbol{f}_\Gamma. \tag{2.31}$$

This would be the system to solve. However, since matrix $\boldsymbol{M}_\Gamma$ is not diagonal, this option is not feasible unless implemented in an iterative scheme, for example of the form

$$\boldsymbol{K}_{\text{in,in}} \boldsymbol{U}^k_{\text{in}} = \boldsymbol{F}_{\text{in}} - \boldsymbol{K}_{\text{in,out}} \boldsymbol{U}^{k-1}_{\text{out}}, \tag{2.32}$$

$$\boldsymbol{M}_\Gamma \boldsymbol{U}^k_{\text{out}} = \boldsymbol{f}_\Gamma - \boldsymbol{N}_\Gamma \boldsymbol{U}^k_{\text{in}}, \tag{2.33}$$

where k is the iteration counter.

The most natural option is to solve problem (2.12)-(2.13), whose matrix counterpart is (2.30)-(2.29), in a coupled way:

$$\begin{bmatrix} \boldsymbol{K}_{\text{in,in}} & \boldsymbol{K}_{\text{in,out}} \\ \boldsymbol{N}_\Gamma & \boldsymbol{M}_\Gamma \end{bmatrix} \begin{bmatrix} \boldsymbol{U}_{\text{in}} \\ \boldsymbol{U}_{\text{out}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{F}_{\text{in}} \\ \boldsymbol{f}_\Gamma \end{bmatrix}. \tag{2.34}$$

It is important to note that this implementation maintains the connectivity of the mesh of Ω_h, that is to say, the mesh of Ω_{in} extended with the nodes of Ω_{out} corresponding to elements cut by Γ.

Even though system (2.34) does not offer particular implementation problems, it could be interesting to consider the possibility to obtain an approximation of the form (2.29) for $\boldsymbol{U}_{\text{out}}$ but *replacing $\boldsymbol{M}_\Gamma$ by a diagonal matrix*. The practical reason for this need is clear. For example, in a fluid-structure interaction problem, in order not to duplicate degrees of freedom only nodal values *interior* to the fluid and the solid can be used when solving the corresponding problem.

Let $\boldsymbol{x}^b_{\text{out}}$ be a node on Ω_{out} corresponding to an element cut by Γ. Consider the edges emanating from $\boldsymbol{x}^b_{\text{out}}$ cut by Γ, and let Γ^b_{out} be the path (surface in 3D) formed by the intersection of these edges with Γ. These intersections are denoted by $\boldsymbol{x}_\Gamma$ with a superscript. In the case of Fig. 2.1, we would have that

$$
\begin{array}{ll}
\Gamma^1_{\text{out}} & \text{is the path formed by} \quad \boldsymbol{x}^1_\Gamma \text{ - } \boldsymbol{x}^2_\Gamma, \\
\Gamma^2_{\text{out}} & \text{is the path formed by} \quad \boldsymbol{x}^3_\Gamma \text{ - } \boldsymbol{x}^4_\Gamma, \\
\Gamma^3_{\text{out}} & \text{is just} \quad \boldsymbol{x}^5_\Gamma, \\
\Gamma^4_{\text{out}} & \text{is the path formed by} \quad \boldsymbol{x}^6_\Gamma \text{ - } \boldsymbol{x}^7_\Gamma, \\
\Gamma^5_{\text{out}} & \text{is the path formed by} \quad \boldsymbol{x}^8_\Gamma \text{ - } \boldsymbol{x}^9_\Gamma \text{ - } \boldsymbol{x}^{10}_\Gamma \text{ - } \boldsymbol{x}^{11}_\Gamma.
\end{array}
$$

When the path is just a point we can compute U^b_{out} by imposing the boundary condition at that point. In the rest of cases, on each path we have that

$$u_h(\boldsymbol{x})|_{\Gamma^b_{\text{out}}} = I^b_{\text{out}}(\boldsymbol{x}) U^b_{\text{out}} + \boldsymbol{I}_{\text{in}}(\boldsymbol{x}) \boldsymbol{U}_{\text{in}}.$$

The idea now is to impose that

$$\frac{\partial}{\partial U^b_{\text{out}}} \int_{\Gamma^b_{\text{out}}} (u_h(\boldsymbol{x}) - \bar{u}(\boldsymbol{x}))^2 = 0,$$

which yields the scalar equation

$$\left(\int_{\Gamma^b_{\text{out}}} I^b_{\text{out}}(\boldsymbol{x}) I^b_{\text{out}}(\boldsymbol{x}) \right) U^b_{\text{out}} = \int_{\Gamma^b_{\text{out}}} I^b_{\text{out}}(\boldsymbol{x}) \bar{u}(\boldsymbol{x}) - \int_{\Gamma^b_{\text{out}}} I^b_{\text{out}}(\boldsymbol{x}) \boldsymbol{I}_{\text{in}}(\boldsymbol{x}) \boldsymbol{U}_{\text{in}},$$

and we can proceed as above, now with a diagonal approximation to $\boldsymbol{M}_\Gamma$.

Considering again the situation in Fig. 2.1, it can be seen that with the approximation described we could easily implement (2.31) *if the connectivities were not modified by the approximate imposition of boundary conditions*. In the case of paths of one or two nodes, that is the case, and (2.31) could be constructed by *trivial modifications of the element matrices*. However, the situation becomes more involved because of the path formed by $\boldsymbol{x}^8_\Gamma$ - $\boldsymbol{x}^9_\Gamma$ - $\boldsymbol{x}^{10}_\Gamma$ - $\boldsymbol{x}^{11}_\Gamma$. The minimization proposed would lead to the coupling of nodes $\boldsymbol{x}^4_{\text{in}}$, $\boldsymbol{x}^5_{\text{in}}$, $\boldsymbol{x}^6_{\text{in}}$ and $\boldsymbol{x}^7_{\text{in}}$ in layer L_0.

A possibility to avoid the complication described would be *to consider only elemental paths*. In the case of Fig. 2.1 that would mean to consider only paths of two nodes. Possible ways to choose this path are

- The longest among the two-node subpaths.
- The closest to the geometric center of the global path.

The first option has been used in a numerical example of Section 2.5.

2.4 Second approach: using internal degrees of freedom

The method described in the previous section works very well *if* Γ *is not too close to* $\partial\Omega_{\rm in}$. When this happens, the method becomes unstable and remedies have to be devised. Let us mention, however, that this instability is *not* particularly strong. In numerical experiments it has manifested as a difficulty for convergence in nonlinear problems (the Navier-Stokes equations in our case) and local spurious peaks close to boundaries for the values of $\boldsymbol{U}_{\rm out}$ with small influence on the values of $\boldsymbol{U}_{\rm in}$.

2.4.1 Description of the method

The idea of the method described in this section is *to impose the satisfaction of the differential equation in the nodes interior to* $\Omega_{\rm in}$*, and to use the nodes of* $\partial\Omega_{\rm in}$ *to prescribe the boundary conditions on* Γ. Let us elaborate this idea.

Let us consider again (2.5), which is the weak form of the differential equation to be solved tested with $v_{h,0}$. The space $V_{h,0}$ where this function belongs may be split as $V_{h,0} = V_{h,1} \oplus V_{h,00}$, where $V_{h,1}$ is the subspace of $V_{h,0}$ of functions vanishing on $\partial\Omega_{\rm in}$ (at nodes of layer L_0 in Fig. 2.1) and $V_{h,00}$ the complement, that is, the subspace of functions that are zero at the *interior* nodes of $\Omega_{\rm in}$. According to this splitting, we may split the unknown as $u_{h,0} = u_{h,1} + u_{h,00}$ and the test functions as $v_{h,0} = v_{h,1} + v_{h,00}$.

Equation (2.5) can be split as

$$B(u_{h,1}, v_{h,1}) + B(u_{h,00}, v_{h,1}) = \langle f, v_{h,1}\rangle_\Omega, \tag{2.35}$$

$$B(u_{h,1}, v_{h,00}) + B(u_{h,00}, v_{h,00}) - k\langle \partial_n u_{h,00}, v_{h,00}\rangle_\Gamma - k\langle \partial_n u_{h,\Gamma}, v_{h,00}\rangle_\Gamma = \langle f, v_{h,00}\rangle_\Omega. \tag{2.36}$$

Recall that integrals are performed over Ω, although the integrals in (2.35) are extended only over $\Omega_{\rm in}$ because this is the support of $v_{h,1}$. The idea now is to keep (2.35) and to replace (2.36) by an approximate prescription of the boundary conditions. In order to use only degrees of freedom of nodes in $\Omega_{\rm in}$, let E be the *extrapolation operator* of functions defined on the elements with an edge in 2D or face in 3D on $\partial\Omega_{\rm in}$ to $\Omega_{\Gamma,\rm in}$. The boundary conditions will be approximately imposed by minimizing the functional $J_2'(u_{h,1}, u_{h,00}) = \|Eu_{h,1} + Eu_{h,00} - \bar{u}\|^2_{L^2(\Gamma)}$, that is, by imposing that

$$\delta_{(0,v_{h,00})} J_2'(u_{h,1}, u_{h,00}) = 0. \tag{2.37}$$

Equations (2.35) and (2.37) form the system of equations of the method we propose, which reads: find $u_{h1} \in V_{h,1}$ and $u_{h,00} \in V_{h,00}$ such that

$$B(u_{h,1}, v_{h,1}) + B(u_{h,00}, v_{h,1}) = \langle f, v_{h,1}\rangle_\Omega, \tag{2.38}$$

$$\langle Eu_{h,1}, Ev_{h,00}\rangle_\Gamma + \langle Eu_{h,00}, Ev_{h,00}\rangle_\Gamma = \langle \bar{u}, Ev_{h,00}\rangle_\Gamma, \tag{2.39}$$

for all $v_{h,1} \in V_{h,1}$ and $v_{h,00} \in V_{h,00}$, where now functions in this last space are defined only on Ω_{in} and extrapolated to $\Omega_{\Gamma,\text{in}}$.

The description of the method is complete up to the definition of the extrapolation operator. In fact, the obvious choice is to extend the local polynomial expansion within the elements with an edge in 2D or face in 3D on $\partial\Omega_{\text{in}}$ to $\Omega_{\Gamma,\text{in}}$. Thus, what needs to be defined is only the domain of the extrapolation. The option we use is described in the following subsection.

A comparison between methods (2.12)-(2.13) and (2.38)-(2.39) in a one-dimensional case using linear elements is shown in Fig. 2.2. In this case it is possible to satisfy exactly the boundary condition $u_h = \bar{u}$.

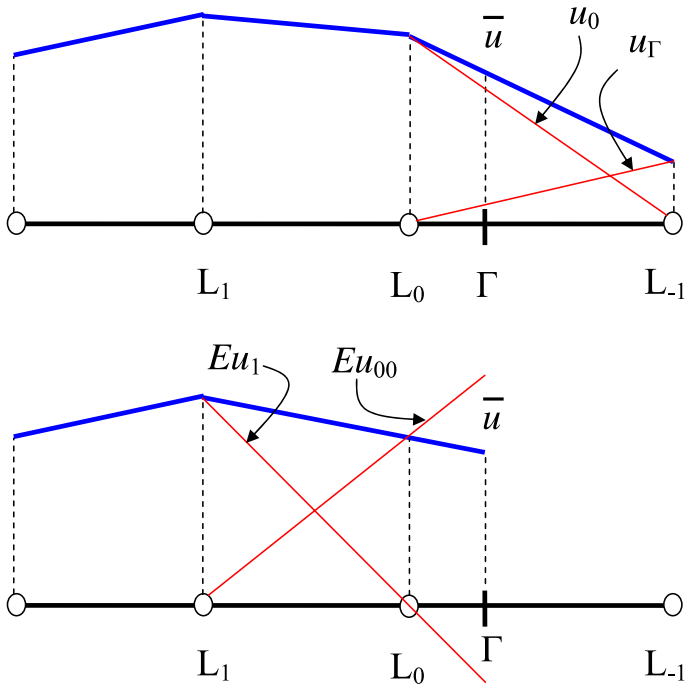


Figure 2.2: Comparison between methods (2.12)-(2.13) (top) and (2.38)-(2.39) (bottom) in a one-dimensional case. The blue line denotes the solution computed in both cases.

Comparing the method proposed in this section with (2.12)-(2.13), some remarks need to be made:

- No boundary integrals have to be computed in (2.38). This is a clear advantage over (2.12).
- The instability detected for the first method when Γ approaches $\partial\Omega_{\text{in}}$ does not appear in this second modification. In fact, the solution is exact when $\Gamma = \partial\Omega_{\text{in}}$ (if $\bar{u}$ is a finite element function).
- From the numerical experiments to be presented in Section 2.5 it is concluded that method (2.12)-(2.13) is more accurate than method (2.38)-(2.39). However, they have the same order of convergence (two when using linear elements).

2.4.2 Implementation aspects

The first point to consider is the extrapolation region of the operator E. There are several possibilities, but the one we have found most accurate is the following. Let K be an element with an edge (in 2D) or face (in 3D) F on $\partial\Omega_{\text{in}}$. Let K_Γ be the cylinder obtained from projecting F onto Γ in an orthogonal way. Then, E is defined as the extension from functions defined on K to functions defined on $K \cup K_\Gamma$. The extrapolation regions obtained this way in 2D using triangular elements are shown in Fig. 2.3.

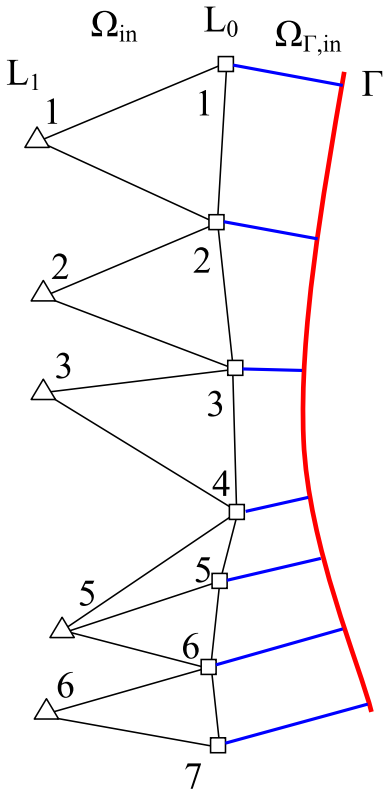


Figure 2.3: Domain of extrapolation in a 2D example.

Suppose now that in Ω_{in} the unknown u_h is interpolated as

$$\begin{aligned} u_h(\boldsymbol{x}) &= \sum_{a=1}^{n_1} I_1^a(\boldsymbol{x}) U_1^a + \sum_{b=1}^{n_{00}} I_{00}^b(\boldsymbol{x}) U_{00}^b \\ &= \boldsymbol{I}_1(\boldsymbol{x})\boldsymbol{U}_1 + \boldsymbol{I}_{00}(\boldsymbol{x})\boldsymbol{U}_{00}, \end{aligned}$$

where $I_1^a(\boldsymbol{x})$ and $I_{00}^b(\boldsymbol{x})$ are the standard interpolation functions, n_1 is the number of nodes interior to Ω_{in} (up to layer L_1) and n_{00} the number of nodes in layer L_0 (see Fig. 2.3).

The objective is to compute $\boldsymbol{U}_{00}$. Equation (2.39) is equivalent to the minimization problem (2.37), that is to say, $\boldsymbol{U}_{00}$ can be computed by minimizing the functional

$$J_2'(\boldsymbol{U}_1, \boldsymbol{U}_{00}) = \int_\Gamma (Eu_h(\boldsymbol{x}) - \bar{u}(\boldsymbol{x}))^2 = \int_\Gamma (E\boldsymbol{I}_1(\boldsymbol{x})\boldsymbol{U}_1 + E\boldsymbol{I}_{00}(\boldsymbol{x})\boldsymbol{U}_{00} - \bar{u}(\boldsymbol{x}))^2$$

which leads to

$$\frac{\partial J_2'}{\partial \boldsymbol{U}_{00}} = 0 \;\Rightarrow\; \boldsymbol{M}_{00}\boldsymbol{U}_{00} = \boldsymbol{f}_{00} - \boldsymbol{N}_{00}\boldsymbol{U}_1, \tag{2.40}$$

where

$$\boldsymbol{M}_{00} = \int_\Gamma E\boldsymbol{I}_{00}^{\mathrm{t}}(\boldsymbol{x})E\boldsymbol{I}_{00}(\boldsymbol{x}), \quad \boldsymbol{f}_{00} = \int_\Gamma E\boldsymbol{I}_{00}^{\mathrm{t}}(\boldsymbol{x})\bar{u}(\boldsymbol{x}), \quad \boldsymbol{N}_{00} = \int_\Gamma E\boldsymbol{I}_{00}^{\mathrm{t}}(\boldsymbol{x})E\boldsymbol{I}_1(\boldsymbol{x}).$$

Suppose the matrix form of (2.38) is

$$\boldsymbol{K}_{1,1}\boldsymbol{U}_1 + \boldsymbol{K}_{1,00}\boldsymbol{U}_{00} = \boldsymbol{F}_1.$$

Combining this with (2.40) it turns out that the final system to be solved is

$$\begin{bmatrix} \boldsymbol{K}_{1,1} & \boldsymbol{K}_{1,00} \\ \boldsymbol{N}_{00} & \boldsymbol{M}_{00} \end{bmatrix} \begin{bmatrix} \boldsymbol{U}_1 \\ \boldsymbol{U}_{00} \end{bmatrix} = \begin{bmatrix} \boldsymbol{F}_1 \\ \boldsymbol{f}_{00} \end{bmatrix}. \tag{2.41}$$

2.4.3 Blending

Let us write problem (2.34) of the previous section as

$$\begin{bmatrix} \boldsymbol{K}_{1,1} & \boldsymbol{K}_{1,00} & \boldsymbol{0} \\ \boldsymbol{K}_{00,1} & \boldsymbol{K}_{00,00} & \boldsymbol{K}_{00,\mathrm{out}} \\ \boldsymbol{0} & \boldsymbol{N}_{\Gamma,00} & \boldsymbol{M}_\Gamma \end{bmatrix} \begin{bmatrix} \boldsymbol{U}_1 \\ \boldsymbol{U}_{00} \\ \boldsymbol{U}_{\mathrm{out}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{F}_1 \\ \boldsymbol{F}_{00} \\ \boldsymbol{f}_\Gamma \end{bmatrix}, \tag{2.42}$$

where the splitting of the matrices corresponds to the splitting of $\boldsymbol{U}_{\mathrm{in}}$ into $\boldsymbol{U}_1$ and $\boldsymbol{U}_{00}$.

Problem (2.41) is obtained by considering the degrees of freedom of *all* nodes in layer L_0 as parameters to prescribe the boundary conditions, but of course the last equation in (2.42) can be kept, in which case the system to be solved is

$$\begin{bmatrix} \boldsymbol{K}_{1,1} & \boldsymbol{K}_{1,00} & \boldsymbol{0} \\ \boldsymbol{N}_{00} & \boldsymbol{M}_{00} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{N}_{\Gamma,00} & \boldsymbol{M}_\Gamma \end{bmatrix} \begin{bmatrix} \boldsymbol{U}_1 \\ \boldsymbol{U}_{00} \\ \boldsymbol{U}_{\mathrm{out}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{F}_1 \\ \boldsymbol{f}_{00} \\ \boldsymbol{f}_\Gamma \end{bmatrix}. \tag{2.43}$$

Clearly, $\boldsymbol{U}_{\mathrm{out}}$ depends on $\boldsymbol{U}_{00}$, but not the other way around. If Γ is very close to $\partial\Omega_{\mathrm{in}}$, the coefficients in $\boldsymbol{M}_\Gamma$ can be very small, but this does not affect the unknowns in the interior of the computational domain and, in fact, $\boldsymbol{M}_\Gamma$ can be replaced by any matrix without altering $\boldsymbol{U}_1$ and $\boldsymbol{U}_{00}$.

As it has been mentioned and as it will be shown in Section 2.5, method (2.42) is more accurate than method (2.43). In order to use (2.42) in all situations except when instability problems may appear, we have implemented a blending of methods (2.42) and (2.43). The idea is simple. When a node in layer L_0 is detected to be very close to Γ, its degree of freedom is used to prescribe the boundary conditions, that is to say, the row in the equation for $\boldsymbol{U}_{00}$ in (2.42) is replaced by the corresponding row in (2.43). This strategy has proved robust and effective. Since usually only a few equations need to be changed (in our case those for which the distance of a node in L_0 to Γ is less than $0.1h$), the overall accuracy obtained is very close to that of method (2.42).

2.5 Numerical examples

In this section we present the numerical results obtained with the two approximations of the Dirichlet boundary conditions proposed. As it has been mentioned, we are interested in flow problems, and in particular in situations in which the Galerkin formulation used heretofore may be unstable. This is why we start this section presenting the stabilized formulation used in the numerical examples.

2.5.1 Stabilized convection-diffusion-reaction and incompressible Navier-Stokes equations

It well known that when the diffusion coefficient k in (2.1) is small the Galerkin method fails and stabilized finite element methods need to be used. It is not our purpose here to explain the roots of the particular method we use (see for example [29]), but only to state it. The bottomline is to replace the bilinear form $B(u_h, v_h)$ and the linear form $\langle f, v_h\rangle_\Omega$ in (2.4) by $B_{\rm stab}(u_h, v_h)$ and $\langle f, v_h\rangle_{\rm stab}$, respectively, given by

$$\begin{aligned} B_{\rm stab}(u_h, v_h) &= B(u_h, v_h) + \sum_K \tau_K \langle -\mathcal{L}^* v_h, \mathcal{L} u_h\rangle_K \\ &= k(\nabla u_h, \nabla v_h) + (\boldsymbol{a}\cdot\nabla u_h, v_h) + s(u_h, v_h) \\ &\quad + \sum_K \tau_K \langle k\Delta v_h + \boldsymbol{a}\cdot\nabla v_h - s v_h, -k\Delta u_h + \boldsymbol{a}\cdot\nabla u_h + s u_h\rangle_K, \end{aligned}$$

and

$$\begin{aligned} \langle f, v_h\rangle_{\rm stab} &= \langle f, v_h\rangle_\Omega + \sum_K \tau_K \langle -\mathcal{L}^* v_h, f\rangle_K \\ &= \langle f, v_h\rangle_\Omega + \sum_K \tau_K \langle k\Delta v_h + \boldsymbol{a}\cdot\nabla v_h - s v_h, f\rangle_K, \end{aligned}$$

where the so called stabilization parameter τ_K is given by

$$\tau_K = \left(c_1 \frac{k}{h^2} + c_2 \frac{a}{h} + s \right)^{-1}.$$

In the numerical experiments presented below we have taken $c_1 = 4$, $c_2 = 2$. The relationship between τ_K and the stabilization parameter of other formulations can be found in [38].

The other problem for which a numerical example is presented below is the incompressible Navier-Stokes equations, which consist in finding a velocity field $\boldsymbol{u}$ and a pressure p such that

$$\begin{aligned} \partial_t \boldsymbol{u} + \boldsymbol{u}\cdot\nabla\boldsymbol{u} - \nu\Delta\boldsymbol{u} + \nabla p &= \boldsymbol{f}, \\ \nabla\cdot\boldsymbol{u} &= 0, \end{aligned}$$

in Ω and for $t > 0$, where $\boldsymbol{f}$ is the vector of body forces and ν the kinematic viscosity. Appropriate initial and boundary conditions have to be appended to this problem. They are described for the particular example of the flow over a cylinder shown later.

Except for the treatment of the Dirichlet boundary conditions for the velocity, which is similar to the one described in detail for the scalar convection-diffusion-reaction equation, the space-discrete problem we solve is

$$
\begin{aligned}
&(\partial_t \boldsymbol{u}_h, \boldsymbol{v}_h) + \langle \boldsymbol{u}_h \cdot \nabla \boldsymbol{u}_h, \boldsymbol{v}_h \rangle_\Omega + \nu(\nabla \boldsymbol{u}_h, \nabla \boldsymbol{v}_h) - (p_h, \nabla \cdot \boldsymbol{u}_h) + (q_h, \nabla \cdot \boldsymbol{v}_h) \\
&\quad + \sum_K \tau_K \langle \nu \Delta \boldsymbol{v}_h + \boldsymbol{u}_h \cdot \nabla \boldsymbol{v}_h + \nabla q_h, \partial_t \boldsymbol{u}_h - \nu \Delta \boldsymbol{u}_h + \boldsymbol{u}_h \cdot \nabla \boldsymbol{u}_h + \nabla p_h \rangle_K \\
&\quad - \langle \boldsymbol{f}, \boldsymbol{v}_h \rangle_\Omega - \sum_K \tau_K \langle \nu \Delta \boldsymbol{v}_h + \boldsymbol{u}_h \cdot \nabla \boldsymbol{v}_h + \nabla q_h, \boldsymbol{f} \rangle_K = 0,
\end{aligned}
$$

where $\boldsymbol{v}_h$ is the velocity test function, q_h the pressure test function and now the stabilization parameter is computed as

$$
\tau_K = \left(c_1 \frac{\nu}{h^2} + c_2 \frac{|\boldsymbol{u}_h|_K}{h} \right)^{-1},
$$

where $|\boldsymbol{u}_h|_K$ is the mean velocity modulus in element K. Any finite difference scheme can be used to approximate the time derivative $\partial_t \boldsymbol{u}_h$. In particular, the second order Crank-Nicolson scheme has been used in the example of Subsection 2.5.3.

Details for the motivation of the formulation described and stability and convergence properties can be found in [31]. The most salient property of the formulation is that *equal* velocity-pressure interpolations can be used. In particular, linear velocities and linear pressures have been used in the numerical example of Subsection 2.5.3 Note however that the pressure interpolation does not affect the approximate imposition of Dirichlet boundary conditions, since these affect only the velocity. Likewise, instabilities of the Galerkin method arising in convection-dominated flows are prevented using the stabilized formulation presented.

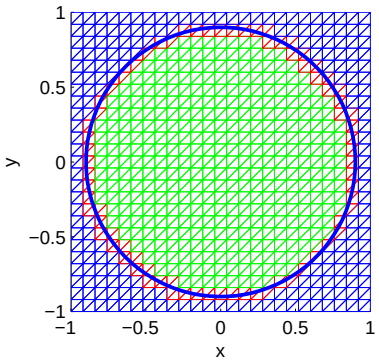


Figure 2.4: Structured mesh and domains Ω_{in} (green) and Ω_Γ (red)

2.5.2 Results for the scalar convection-diffusion-reaction equation

In this subsection we illustrate the behavior of the proposed methods for the scalar convection-diffusion-reaction equation. The Poisson, diffusion-reaction and convection-diffusion equations are solved in a domain Ω enclosed in a circle of radius $R < 1$. We choose the *hold-all*

domain $B = (-1, 1) \times (-1, 1)$, where a system of Cartesian coordinates (x, y) with its origin at the center of the circle has been adopted. A structured mesh of right-angled linear triangular elements is constructed in *B*, *h* being the length of the edges corresponding to the cathetus (see Fig. 2.4).

The Poisson equation

Let us start solving the Poisson equation with $k = 1$, $\boldsymbol{a} = \mathbf{0}$, $s = 0$, $f = 1$ to check the performance and convergence of the proposed methods. Results are shown in Fig. 2.5 (top and bottom left). No significative difference between the fields u_h obtained with the different methods can be appreciated, even for the coarsest meshes.

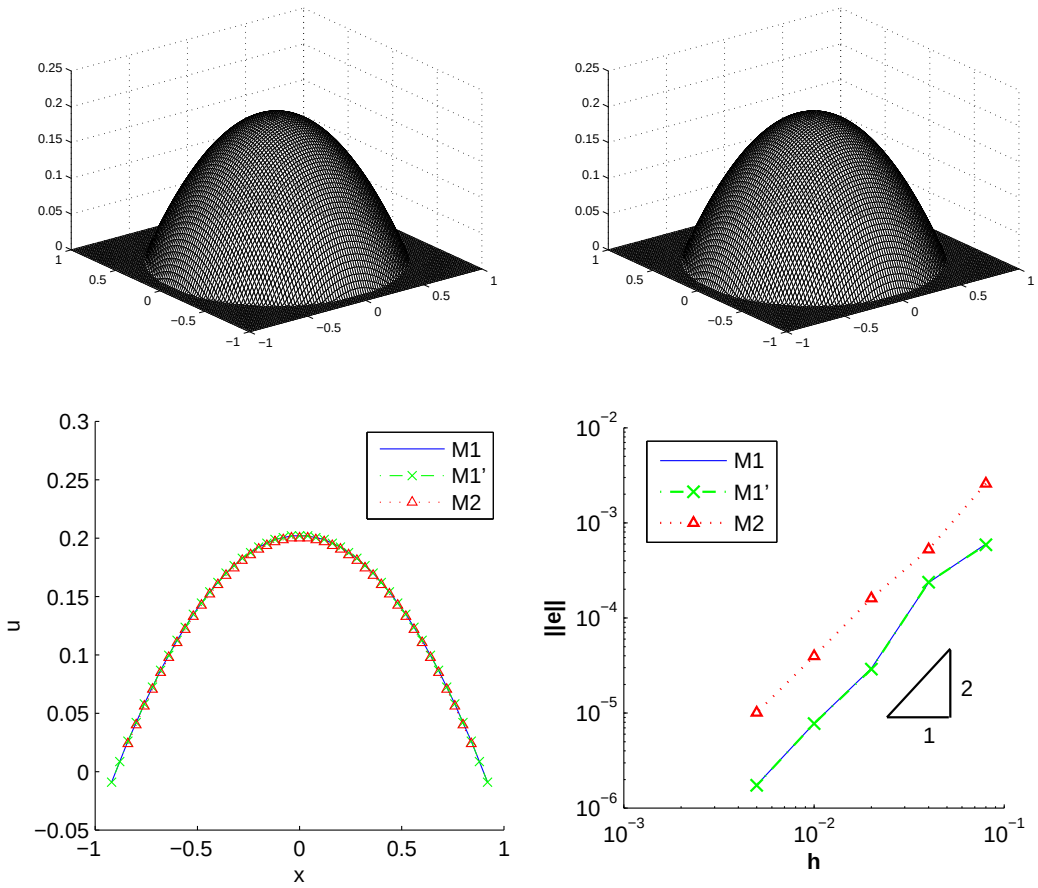


Figure 2.5: Comparison between the proposed methods. Top left: elevation u_h for the Poisson equation for M1. Top right: same for M2. Bottom left: cut along $y = 0$ for the coarsest mesh used, with $h = \frac{2}{25}$. Bottom right: convergence plot in $L^2(\Omega)$ for methods M1, M1' and M2

The analytical solution for this case is known to be

$$u(x, y) = \frac{1}{4}(R^2 - x^2 - y^2).$$

Fig. 2.5 (bottom right) shows the errors $\|u - u_h\|_{L^2(\Omega)}$ versus the element size h. As it can be seen, both the first method described in Section 2.3 (labeled M1 in the following), and the

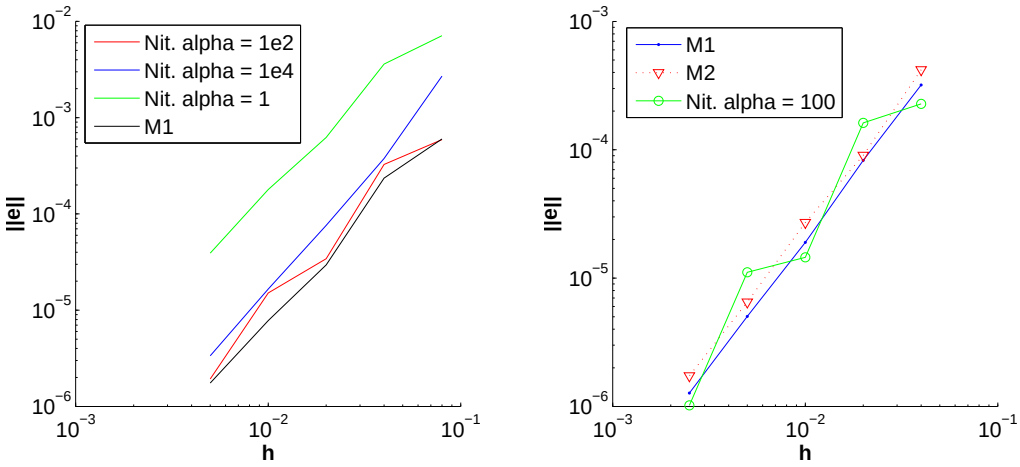


Figure 2.6: Comparison between M1, M2 and Nitsche's method. Left: convergence plot in $L^2(\Omega)$ for method M1 and Nitsche's method with different values of the parameter α. Right: convergence plot in $L^2(\partial\Omega)$ for methods M1, M2 and Nitsche's method with $\alpha = 100$.

second one described in Section 2.4 (labeled M2) show quadratic convergence, although the error turns out to be smaller for the former. The modified version of M1 (referred to as M1′), which uses a diagonal approximation of matrix $\boldsymbol{M}_\Gamma$ computed by considering only the longest elemental paths (see Subsection 2.3.3), shows no significative error increment with respect to M1.

In order to compare the performance of the methods proposed with Nitsche's method, in Fig. 2.6 (left) we have also plotted the convergence obtained using this method with three different choices of the parameter α in (2.4) (taking $k^* = k$), namely, $\alpha = 100$, which is approximately the optimal value found from numerical experiments, $\alpha = 1$ and $\alpha = 10000$. It can be observed that the performance of method M1 is superior to Nitsche's method, even for its optimal case, and that this method is sensitive to the choice of the parameter α. This is aggravated in problems with convection and/or reaction, for which k^* (or, alternatively, α) must be chosen in terms of the advection velocity and the reaction coefficient. In Fig. 2.6 (right) we have plotted convergence in $L^2(\partial\Omega)$, and therefore the error is due only to the imposition of the boundary conditions. Nitsche's method displays a non-monotone behavior due to the way the elements cut the boundary of the domain for different meshes. Again, method M1 and M2 show a similar behavior when only the errors on the boundary are taken into account.

Reaction-diffusion

When the reactive term s dominates over the diffusive one it is well known that oscillations in the finite element approximated solution u_h appear near the boundary layer. It is thus convenient to check how do the proposed methods behave in the presence of this *Gibb's phenomenon*. Fig. 2.7 shows u_h for M1, M2 and the local remeshing strategy described in Section 2.1, labeled CD in Fig. 2.7. These results correspond to the reaction dominated case, where $k = 10^{-5}$, $\boldsymbol{a} = \boldsymbol{0}$, $s = 1$, $f = 1$.

Although oscillations remain bounded close to the exact solution u both for M1 and M2,

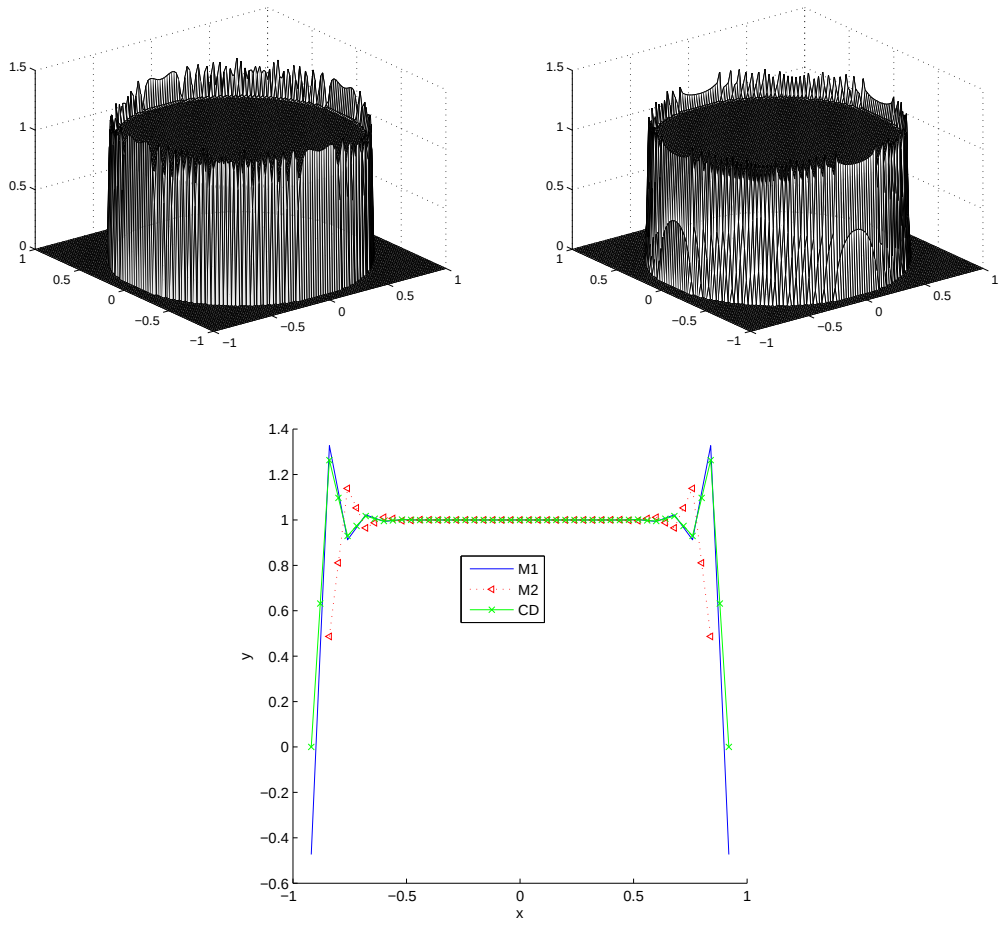


Figure 2.7: Reaction-dominated case, exact integration. Elevation u_h for M1 (top left) and M2 (top right). Cut along $y = 0$ for the coarsest used mesh $(h = \frac{2}{25})$ (bottom).

they happen to be greater in the former than in the latter. Nevertheless, when compared to results obtained with CD, oscillations in M1 are practically of the same magnitude as those obtained for the classical method, while the solution for M2 clearly shows a reduction in the amount of oscillation.

If nodal integration is used to compute the contribution of the reactive term to the resulting system of equations, oscillations can be avoided, since the resulting matrix is of *non-negative type*, and thus the *discrete minimum principle* is satisfied, that is to say, for $f \geq 0$ the minimum of the solution is attained at boundary nodes (this principle holds if and only if the discrete maximum principle does, see e.g. [28]). In this case none of the two methods shows any oscillation (see Fig. 2.8), and the only difference between them is due to the fact that M2 uses only the degrees of freedom corresponding to the nodes in the Ω_{in} domain, while M1 incorporates also the nodes corresponding to the Ω_{out} domain (this is also the reason why M1 leads to a better approximation to u_h than M2).

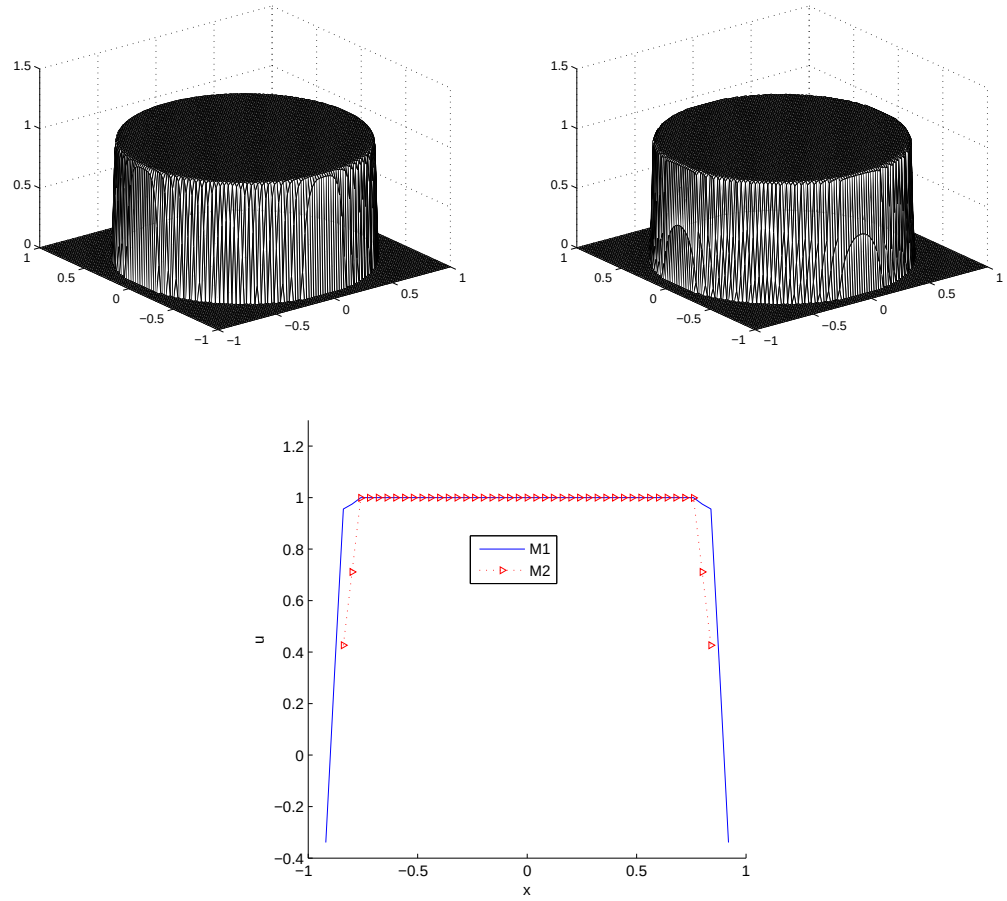


Figure 2.8: Reaction-dominated case, nodal integration. Elevation u_h for M1 (top left) and M2 (top right). Cut along $y = 0$ for the coarsest used mesh $(h = \frac{2}{25})$ (bottom).

Convection-diffusion

Fig. 2.9 shows the behavior of methods M1 and M2 in the convection-dominated case, where $k = 10^{-6}$, $\boldsymbol{a} = (1, 0)$, $s = 0$, $f = 1$. The stabilized formulation described in Subsection 2.5.1 has been used. Both methods M1 and M2 perform well, although again oscillations are greater for M1. This time, however, oscillations for M1 are substantially greater than those which appear when applying Dirichlet conditions in boundary fitting meshes CD, with the local remeshing strategy described in Section 2.1. Again also, M2 shows less oscillations than CD.

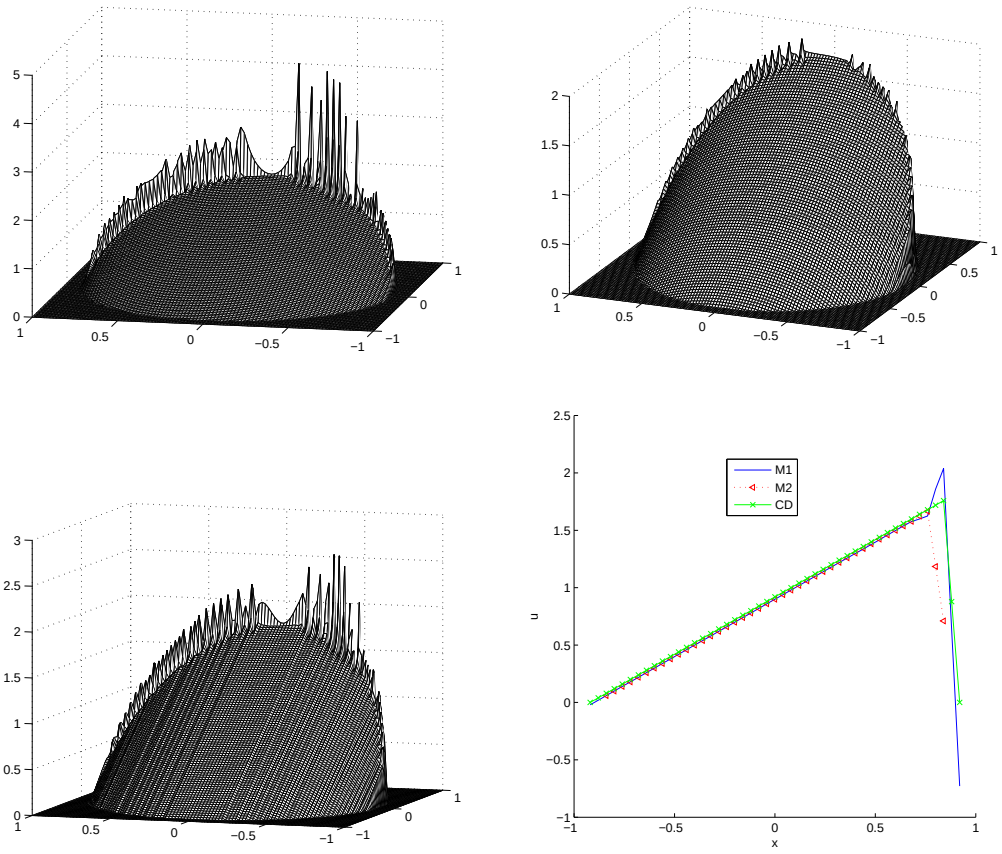


Figure 2.9: Convection-dominated case. Elevation u_h for M1 (top left), M2 (top right) and CD (bottom left). Cut along $y = 0$ for the coarsest used mesh $(h = \frac{2}{25})$ (bottom right).

Despite the different behavior that both methods show in the boundary layer, the difference between the two methods in the Ω_{in} domain is practically negligible.

The local oscillations appearing in M1, altogether with the fact that the splitting of elements in the Ω_{in} domain can lead to an ill-conditioning of the resulting system of equations when Γ is too close to $\partial\Omega_{\text{in}}$, can prevent convergence in nonlinear problems. This is what motivates the *blending strategy* proposed in Subsection 2.4.3.

2.5.3 Results for the incompressible Navier-Stokes equations

In this subsection we analyze a numerical example involving the flow past a cylinder. Again the formulation described in Subsection 2.5.1 has been used.

The *hold-all domain* is the rectangle $B = [0, 16] \times [0, 8]$, from which a cylinder of diameter $D = 1$ and centered at (4,4) is extracted. The velocity at $x = 0$ is prescribed to (10,0), whereas at $y = 0$ and $y = 8$ the $y-$velocity component is prescribed to 0 and the x-component is left free. The outflow (where both the x- and y-components are free) is $x = 16$. The Reynolds number is 100, based on the cylinder diameter and the prescribed inflow velocity. The finite element mesh employed consists of 10000 linear triangles. The Crank-Nicolson scheme has been used for the time integration, with a time step size $\delta t = 1$.

Velocity contours and pressure contours at $t = 200$ obtained using methods M1 and M2 are shown in Fig. 2.10. The important issue is to observe that boundary conditions are well approximated both using M1 and M2. The evolution of the y-velocity component at point $(10, 4)$ is shown in Fig. 2.11. It can be observed that both methods yield a similar amplitude, the frequency obtained with method M2 being slightly smaller. The dimensionless period of the oscillations is found to be $T = 6.11$ for method M1 and $T = 6.5$ with method M2. Consistently with the results for the convection-diffusion-reaction equation, method M2 seems to behave always as more dissipative than method M1.

2.6 Conclusions

In this chapter we have proposed a way to prescribe approximately Dirichlet boundary conditions for immersed boundary methods. The main idea is to use as degrees of freedom for this imposition those associated to the nodes adjacent to the boundary of the computational domain. In a first approach, these nodes are taken in the exterior of the domain, but this may yield instabilities (mild and unusual) that can be overcome by using interior nodes and extrapolation. In any case, the degrees of freedom are computed by minimizing the distance of the unknown to the boundary datum in the L^2 norm of the boundary.

The method proposed turns out to be accurate (second order for linear elements) and robust. We have checked its numerical performance in a variety of situations in flow problems, paying particular attention to problems that require stabilization.

From the implementation point of view, the method satisfies the main design condition of using only the degrees of freedom of the mesh of Ω_h. This is particularly important in the case of domains with moving boundaries in which a single fixed mesh is used during the whole calculation, which in fact is the motivation that led us to formulate the method proposed.

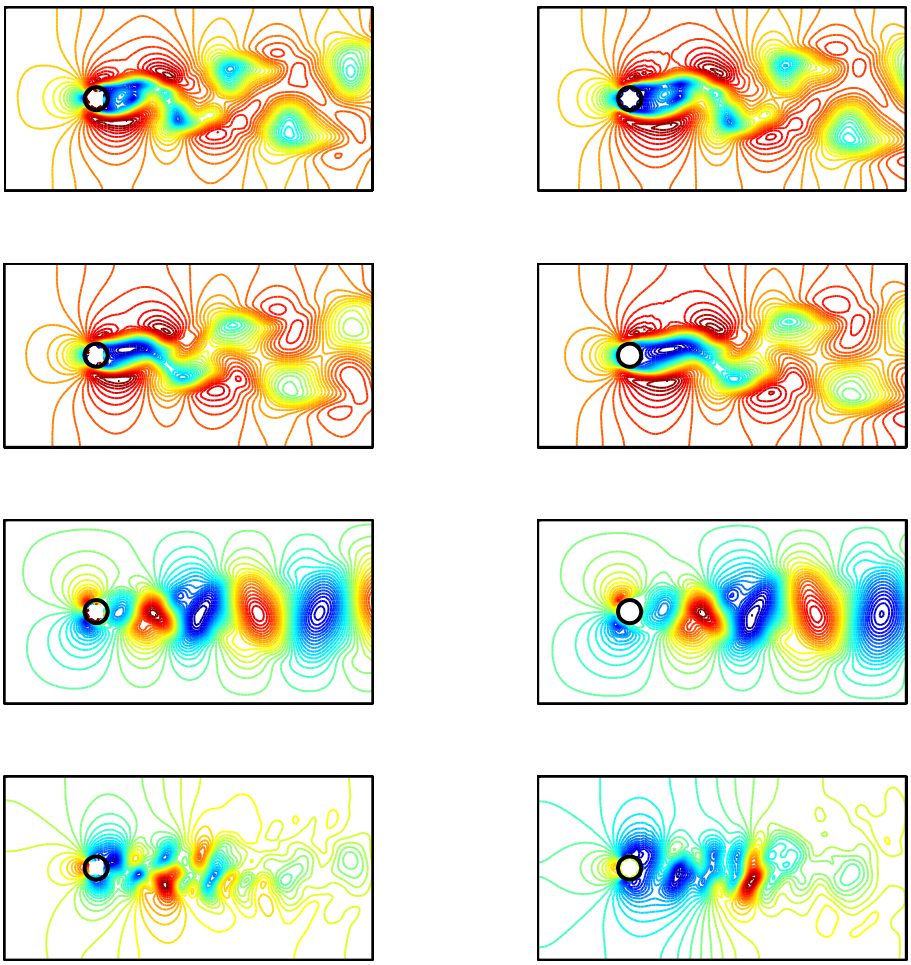


Figure 2.10: Incompressible Navier-Stokes equations. Solution at $t = 200$. Left: method M1, Right: method M2. From the top to the bottom: velocity module, contours of velocity x-component, contours of velocity y-component, pressure contours.

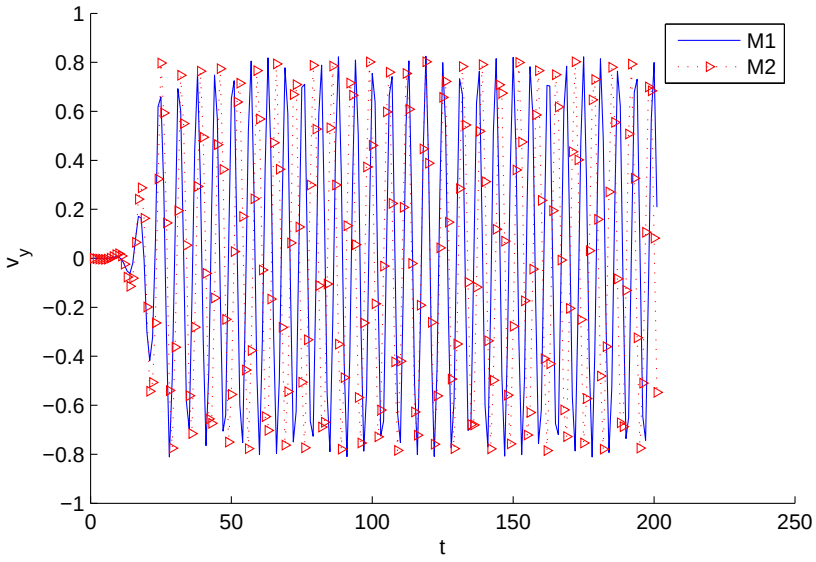


Figure 2.11: Incompressible Navier-Stokes equations. Evolution of the y-velocity component at point $(10, 4)$ for methods M1 and M2.

Chapter 3

A symmetric method for weakly imposing Dirichlet boundary conditions in embedded grids

In this chapter we propose a way to weakly prescribe Dirichlet boundary conditions in embedded grids. The key feature of the method is that no large penalty parameter is needed in order to ensure stability and that it is symmetric for symmetric problems. In the Poisson problem this is achieved by introducing an additional element-discontinuous stress variable. Additional terms are required in order to guarantee stability in the convection-diffusion equation and the Stokes problem. The proposed method is then easily extended to the transient Navier-Stokes equations.

3.1 Introduction

In this chapter we propose a new method for weakly imposing Dirichlet boundary conditions in embedded grids. In contrast to the method presented in the Chapter 2, the equations imposing boundary conditions need to *compete* with the ones enforcing the variational equation.

Several variations of Nitsche's method for weakly imposing boundary conditions can be found in the literature [64, 50, 108]. These methods are symmetric for symmetric problems, and do not need additional degrees of freedom to impose boundary conditions. However, a user defined stabilization parameter is required. Choosing 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 can be addressed by using the inverse estimates in order to define the minimum value for the stabilization parameter (see [44] in which the stabilization parameter for the heat transfer problem is studied). However there are still some non-dimensional constants to be defined in the inverse estimates, and it remains to be seen how to apply the method to non-symmetric problems such as the convection-diffusion equation.

A list of desired properties for our strategy for imposing Dirichlet boundary conditions in non-matching grids can be extracted from the previously described methods:

- Optimal convergence order should be obtained.

- No additional degrees of freedom should be needed in order to enforce boundary conditions.

- The method should be free of user-defined penalty or stabilization parameters which might ill-condition the resulting system of equations.

- The resulting variational form should be symmetric for symmetric problems, but also capable of dealing with flow problems such as the convection-diffusion or the Navier-Stokes equations.

The starting point of the strategy we propose is the method presented in [58]. This method imposes Dirichlet boundary conditions weakly but does not require of any user defined stabilization or penalty parameter. In order to do so, a hybrid-formulation which introduces an additional element-wise discontinuous stress field is used. However, this additional stress field is only required in the elements which are cut by the immersed boundary, and since it is discontinuous across inter-element boundaries, it can be condensed prior to solving the resulting system of equations. The method shows optimal order of convergence and satisfies the design condition of not needing additional degrees of freedom in order to impose boundary conditions. However, it is non-symmetric even for symmetric problems.

In Section 3.2 a symmetric version of the method proposed in [58] for Poisson's problem is presented. The main idea is again to use a hybrid formulation with an additional element-wise discontinuous stress field. However, some additional terms are added so that the method is symmetric. A stability analysis is performed in order to ensure that the method is stable without the need of user defined penalty parameters. In Section 3.3 we extend the method to the convection-diffusion equation. Additional terms are required to further enforce boundary conditions in order to guarantee the stability of the method in the case of convection dominated flows. The stability analysis shows that boundary conditions can be given a different treatment in the inflow and the outflow boundary, which justifies the chosen weighting term for the boundary conditions enforcement. In Section 3.4 we deal with the treatment of boundary conditions in the case of the Stokes problem, and the stability of the proposed method for this particular problem is shown. Additional terms which enforce the velocity in the direction normal to the immersed boundary are required to keep the symmetry of the problem. Finally, in Section 3.5 we put together the terms which define our method for the convection-diffusion equation and the Stokes problem and we describe the strategy to impose boundary conditions in the transient incompressible Navier-Stokes equations. Numerical examples illustrate the behavior of the proposed method in a number of situations in Section 3.6 and some conclusions close the chapter in Section 3.7.

3.2 A symmetric method for Poisson's problem

In this section a symmetric method for imposing boundary conditions for Poisson's problem is presented. In the following sections the method will be extended to other symmetric and non-symmetric problems.

3.2.1 Problem statement

Let us consider the problem

$$-k\Delta u = f \quad \text{in } \Omega, \tag{3.1}$$

$$u = \bar{u} \quad \text{on } \Gamma = \partial\Omega, \tag{3.2}$$

where $k > 0$, f is a given forcing function and $\bar{u}$ is the given Dirichlet boundary condition. We assume that the subdomain Ω is polyhedral, and covered by the domain Ω_h, as explained in Section 3.1.

We can now consider a hybrid two-field formulation in which we introduce an additional flux unknown $\boldsymbol{\sigma}$ to the previous problem. The problem can now be written as:

$$-k\Delta u = f \quad \text{in } \Omega, \tag{3.3}$$

$$\frac{1}{k}\boldsymbol{\sigma} = \nabla u \quad \text{in } \Omega, \tag{3.4}$$

$$u = \bar{u} \quad \text{on } \Gamma = \partial\Omega, \tag{3.5}$$

3.2.2 Weak form

Let $\mathcal{P}_h = \{K\}$ be a finite element partition of Ω_h from which we construct the finite element space $V_h \subset H^1(\Omega_h)$ (we will consider V_h made of continuous functions) and $S_h \subset L^2(\Omega_h)^d$ (we will consider S_h made of element-wise discontinuous functions). Our symmetric variational form of the problem consists of finding $u_h \in V_h$ and $\boldsymbol{\sigma}_h \in S_h$ such that:

$$k(\nabla u_h, \nabla v_h) - \langle \boldsymbol{\sigma}_h \cdot \boldsymbol{n}, v_h\rangle_\Gamma + \frac{1}{n}(\nabla v_h, \boldsymbol{\sigma}_h) - \frac{1}{n}k(\nabla v_h, \nabla u_h) = \langle f, v_h\rangle_\Omega, \quad \forall v_h \in V_h, \tag{3.6}$$

$$-\frac{1}{nk}(\boldsymbol{\tau}_h, \boldsymbol{\sigma}_h) + \frac{1}{n}(\boldsymbol{\tau}_h, \nabla u_h) - \langle \boldsymbol{\tau}_h \cdot \boldsymbol{n}, u_h\rangle_\Gamma = -\langle \boldsymbol{\tau}_h \cdot \boldsymbol{n}, \bar{u}\rangle_\Gamma, \quad \forall \boldsymbol{\tau}_h \in S_h \tag{3.7}$$

where n is a free parameter for which we will propose an expression in the following sections. Note that in equation (3.6) we have used the $\boldsymbol{\sigma}$ field only in the terms corresponding to the fluxes. Here and below, $(\cdot,\cdot)$ denotes the L^2 product in Ω. In general, the integral of two function g_1 and g_2 over a domain ω will be denoted by $\langle g_1, g_2\rangle_\omega$, the $L^2(\omega)$ inner product by $(\cdot,\cdot)_\omega$ and the norm in a function space X by $\|\cdot\|_X$, with the simplifications $\|\cdot\|_{L^2(\Omega)} \equiv \|\cdot\|$ and $(\cdot,\cdot)_\Omega \equiv (\cdot,\cdot)$.

Note that there are four overlapping, non-independent equations in the previous variational form which are added together, with n playing the role of the weight assigned to each of the equations:

$$k(\nabla u_h, \nabla v_h) - \langle \boldsymbol{\sigma}_h \cdot \boldsymbol{n}, v_h\rangle_\Gamma = \langle f, v_h\rangle_\Omega, \tag{3.8}$$

$$\frac{1}{n}(\nabla v_h, \boldsymbol{\sigma}_h) - \frac{1}{n}k(\nabla v_h, \nabla u_h) = 0, \tag{3.9}$$

$$-\frac{1}{nk}(\boldsymbol{\tau}_h, \boldsymbol{\sigma}_h) + \frac{1}{n}(\boldsymbol{\tau}_h, \nabla u_h) = 0, \tag{3.10}$$

$$-\langle \boldsymbol{\tau}_h \cdot \boldsymbol{n}, u_h\rangle_\Gamma = -\langle \boldsymbol{\tau}_h \cdot \boldsymbol{n}, \bar{u}\rangle_\Gamma, \tag{3.11}$$

(3.8) is weakly enforcing (3.3) tested against v_h. (3.9) and (3.10) are weakly enforcing (3.4) tested against $\frac{1}{n}\nabla v_h$ and $-\frac{1}{n}\boldsymbol{\tau}_h$ respectively, which corresponds to the least squares minimization of the functional:

$$J_1(u_h, \boldsymbol{\sigma}_h) = \frac{1}{2n}\|\boldsymbol{\sigma}_h - k\nabla u_h\|^2. \tag{3.12}$$

Finally, (3.11) is weakly enforcing (3.5) tested against $-\boldsymbol{\tau}_h \cdot \boldsymbol{n}$. Note that the main difference between the presented method and the method described in [58] when applied to Poisson's problem is (3.10), which does not appear in [58], and is the term which makes the presented method symmetric.

Let us remark that the volume integrals in (3.6)-(3.7) are performed over $\Omega = \Omega_{\text{in}} \cup \Omega_{\Gamma,\text{in}}$ as explained in Chapter 2.

3.2.3 Stability

In this subsection we prove that the formulation given by (3.6)-(3.7) is stable, and as a consequence has a unique solution. We define the norm:

$$|\!|\!|(u, \boldsymbol{\sigma})|\!|\!|^2 = k\|\nabla u\|^2 + \frac{k}{h}\|u\|^2_{L^2(\Gamma)} + \frac{1}{k}\|\boldsymbol{\sigma}\|^2, \tag{3.13}$$

where h is the element size. For simplicity we will assume that $\mathcal{P}_h$ is a uniform finite element partition. We define the bilinear form on $[V_h, S_h] \times [V_h, S_h]$:

$$\begin{aligned} B([u_h, \boldsymbol{\sigma}_h], [v_h, \boldsymbol{\tau}_h]) =& k(\nabla u_h, \nabla v_h) - \langle \boldsymbol{\sigma}_h \cdot \boldsymbol{n}, v_h\rangle_\Gamma + \frac{1}{n}(\nabla v_h, \boldsymbol{\sigma}_h) - \frac{1}{n}k(\nabla v_h, \nabla u_h) \\ &- \frac{1}{nk}(\boldsymbol{\tau}_h, \boldsymbol{\sigma}_h) + \frac{1}{n}(\boldsymbol{\tau}_h, \nabla u_h) - \langle \boldsymbol{\tau}_h \cdot \boldsymbol{n}, u_h\rangle_\Gamma. \end{aligned} \tag{3.14}$$

We now suppose that V_h and S_h are such that the following conditions hold for all the elements cut by the boundary Γ:

$$\forall v_h \in V_h \quad \exists \boldsymbol{\tau}_h \in S_h | \quad \|v_h\|^2_{L^2(\Gamma)} \lesssim \langle \boldsymbol{\tau}_h \cdot \boldsymbol{n}, v_h\rangle_\Gamma + \delta_0 h\|\nabla v_h\|^2, \tag{3.15}$$

$$\|\boldsymbol{\tau}_h\|_{L^2(K)} = \|v_h\|_{L^2(K)}, \tag{3.16}$$

where δ_0 is a non-dimensional constant, and we have used the notation $\|A\|_X \lesssim \|B\|_Y$ if there exists a constant C such that $\|A\|_X \leq C\|B\|_Y$. In the case of a equal interpolation for V_h and S_h and straight intersection of Γ with the elements, $\boldsymbol{\tau}_h$ can be defined as:

$$\boldsymbol{\tau}_h = \boldsymbol{n} v_h, \tag{3.17}$$

where $\boldsymbol{n}$ is supposed to be constant in the part of the boundary corresponding to each element. In the case of a linear interpolation for V_h and piecewise constant interpolation for S_h, which is the situation of the numerical examples, we can define $\boldsymbol{\tau}_h$ in each element cut by the domain boundary as:

$$\boldsymbol{\tau}_h = \boldsymbol{n} v_{\text{qm}} \text{sgn}(v_{\text{lm}}), \tag{3.18}$$

where $v_{\rm lm}$ is the mean value of v_h in each element and $v_{\rm qm}$ is the square root of the mean value of v_h^2 in each element:

$$v_{\rm lm} = \frac{\int_K v_h}{\int_K 1}, \quad v_{\rm qm} = \sqrt{\frac{\int_K v_h^2}{\int_K 1}}.$$

(3.16) holds from the definition of $\boldsymbol{\tau}_h$. (3.15) holds for $\boldsymbol{\tau}_h^* = \boldsymbol{n} v_{\rm lm}$ since for linear elements:

$$\langle \boldsymbol{n} v_{\rm lm} \cdot \boldsymbol{n}, v_h \rangle_\Gamma = \langle v_{\rm lm}, v_{\rm lm} \rangle_\Gamma. \tag{3.19}$$

On the other hand Schwarz inequality states that:

$$\int_K fg \leq \left(\int_K f^2 \right)^{\frac{1}{2}} \left(\int_K g^2 \right)^{\frac{1}{2}},$$

which, if we take $f = v_h$ and $g = 1$ allows us to state:

$$v_{\rm qm} \geq v_{\rm lm}.$$

If we take $f = -v_h$ and $g = 1$ we can state:

$$v_{\rm qm} \geq -v_{\rm lm},$$

which implies $v_{\rm qm} \geq |v_{\rm lm}|$. Taking into account that $v_{\rm qm} \geq 0$ we can see that:

$$\begin{aligned} \langle \boldsymbol{n} v_{\rm qm} \mathrm{sgn}(v_{\rm lm}) \cdot \boldsymbol{n} \rangle_\Gamma &= v_{\rm qm} \mathrm{sgn}(v_{\rm lm}) \int_K v_h \\ &= v_{\rm qm} |v_{\rm lm}| \int_K 1 \geq |v_{\rm lm}|^2 \int_K 1 = \langle v_{\rm lm}, v_{\rm lm} \rangle_\Gamma, \end{aligned} \tag{3.20}$$

which demonstrates that (3.15) holds for the definition of $\boldsymbol{\tau}_h$ in (3.18) in the case of a linear interpolation space for V_h and elementwise constant stresses for S_h.

We take $[v_h, \boldsymbol{\tau}_h] = [u_h, -\boldsymbol{\sigma}_h - \frac{\beta}{h} k \tilde{\boldsymbol{\tau}}_h]$, where $\tilde{\boldsymbol{\tau}}_h$ is the counterpart of u_h in (3.15)-(3.16), h is the element size and β is a dimensionless constant to be defined. The following relation holds for u_h:

$$\|u_h\|^2_{L^2(K)} \lesssim h \|u_h\|^2_{L^2(\Gamma \cap K)} + h^2 \|\nabla u_h\|^2_{L^2(K)}. \tag{3.21}$$

We have:

$$
\begin{aligned}
&B([u_h,\boldsymbol{\sigma}_h],[u_h,-\boldsymbol{\sigma}_h-\frac{\beta}{h}k\tilde{\boldsymbol{\tau}}_h])\\
=&k(\nabla u_h,\nabla u_h)-\langle\boldsymbol{\sigma}_h\cdot\boldsymbol{n},u_h\rangle_\Gamma+\frac{1}{n}(\nabla u_h,\boldsymbol{\sigma}_h)-\frac{1}{n}k(\nabla u_h,\nabla u_h)\\
&+\frac{1}{nk}(\boldsymbol{\sigma}_h,\boldsymbol{\sigma}_h)-\frac{1}{n}(\boldsymbol{\sigma}_h,\nabla u_h)+\langle\boldsymbol{\sigma}_h\cdot\boldsymbol{n},u_h\rangle_\Gamma\\
&+\frac{\beta}{nh}(\tilde{\boldsymbol{\tau}}_h,\boldsymbol{\sigma}_h)-\frac{\beta k}{nh}(\tilde{\boldsymbol{\tau}}_h,\nabla u_h)+\frac{\beta k}{h}\langle\tilde{\boldsymbol{\tau}}_h\cdot\boldsymbol{n},u_h\rangle_\Gamma\\
\gtrsim&(1-\frac{1}{n})k\|\nabla u_h\|^2+\frac{1}{nk}\|\boldsymbol{\sigma}_h\|^2+\frac{\beta k}{h}\|u_h\|^2_{L^2(\Gamma)}\\
&-\frac{\beta}{nh}\|u_h\|\|\boldsymbol{\sigma}_h\|-\frac{\beta k}{nh}\|u_h\|\|\nabla u_h\|-\beta\delta_0 k\|\nabla u_h\|^2\\
\geq&(1-\frac{1}{n})k\|\nabla u_h\|^2+\frac{1}{nk}\|\boldsymbol{\sigma}_h\|^2+\frac{\beta k}{h}\|u_h\|^2_{L^2(\Gamma)}\\
&-\frac{\beta k}{2nh^2}\|u_h\|^2-\frac{\beta}{2kn}\|\boldsymbol{\sigma}_h\|^2-\frac{\beta k}{2nh^2}\|u_h\|^2-\frac{\beta k}{2n}\|\nabla u_h\|^2-\beta\delta_0 k\|\nabla u_h\|^2\\
\geq&\left(1-\frac{1}{n}-\beta(\frac{1}{2n}+\delta_0)\right)k\|\nabla u_h\|^2+\left(\frac{1}{kn}-\frac{\beta}{2kn}\right)\|\boldsymbol{\sigma}_h\|^2+\frac{\beta k}{h}\|u_h\|^2_{L^2(\Gamma)}-\frac{\beta k}{h^2 n}\|u_h\|^2\\
\gtrsim&\left(1-\frac{1}{n}-\beta(\frac{3}{2n}+\delta_0)\right)k\|\nabla u_h\|^2+\left(\frac{1}{kn}-\frac{\beta}{2kn}\right)\|\boldsymbol{\sigma}_h\|^2+(1-\frac{1}{n})\frac{\beta k}{h}\|u_h\|^2_{L^2(\Gamma)}.
\end{aligned}
\tag{3.22}
$$

Imposing:

$$
n>1,\quad \beta<\min\left(\frac{1-\frac{1}{n}}{(\frac{3}{2n}+\delta_0)},2\right),
\tag{3.23}
$$

and taking into account that:

$$
\begin{aligned}
|\!|\!|(u_h,-\boldsymbol{\sigma}_h-\frac{\beta}{h}k\tilde{\boldsymbol{\tau}}_h)|\!|\!|^2=&k\|\nabla u_h\|^2+\frac{k}{h}\|u_h\|^2_{L^2(\Gamma)}+\frac{1}{k}\|-\boldsymbol{\sigma}_h-\frac{\beta}{h}k\tilde{\boldsymbol{\tau}}_h\|^2\\
\leq&k\|\nabla u_h\|^2+\frac{k}{h}\|u_h\|^2_{L^2(\Gamma)}+\frac{1}{k}\|\boldsymbol{\sigma}_h\|^2+\frac{k\beta^2}{h^2}\|\tilde{\boldsymbol{\tau}}_h\|^2\\
=&k\|\nabla u_h\|^2+\frac{k}{h}\|u_h\|^2_{L^2(\Gamma)}+\frac{1}{k}\|\boldsymbol{\sigma}_h\|^2+\frac{k\beta^2}{h^2}\|u_h\|^2\\
\lesssim&k(1+\beta^2)\|\nabla u_h\|^2+\frac{k}{h}\left(1+\beta^2\right)\|u_h\|^2_{L^2(\Gamma)}+\frac{1}{k}\|\boldsymbol{\sigma}_h\|^2,
\end{aligned}
\tag{3.24}
$$

we obtain the result we wished to prove:

Theorem For all $[u_h,\boldsymbol{\sigma}_h]$ there exist $[v_h,\boldsymbol{\tau}_h]$ and $\alpha>0$ such that:

$$
B([u_h,\boldsymbol{\sigma}_h],[v_h,\boldsymbol{\tau}_h])\geq\alpha|\!|\!|(u_h,\boldsymbol{\sigma}_h)|\!|\!||\!|\!|(v_h,\boldsymbol{\tau}_h)|\!|\!|.
\tag{3.25}
$$

We have now demonstrated that our symmetric bilinear form is stable for the Poisson problem for any $n>1$. We consider $n=2$ in the following.

3.2.4 Implementation and comparison to Nitsche's method

A key feature of the presented method is that since the stress field is discontinuous across interelement edges, it can be eliminated from the final equations. We will show in this section that after eliminating the stress variables some of the terms cancel out, and the final expression of the terms to be implemented is very similar to that of Nitsche's method, but with the important feature that no penalty parameters need to be estimated, since we have already found a value for n for which the method is stable.

Let $\boldsymbol{U}$ and $\boldsymbol{\Sigma}$ be the arrays of nodal unknowns of u_h and $\boldsymbol{\sigma}_h$ and let us define the matrices $\boldsymbol{K}_{uu}$, $\boldsymbol{K}_{\sigma\sigma}$, $\boldsymbol{K}_{\sigma u}$ and $\boldsymbol{K}_{u\sigma}$ related to the weak form integrals:

$$\begin{aligned}
\boldsymbol{K}_{uu}\cdot\boldsymbol{U} \text{ which comes from the term } &+k(\nabla v_h,\nabla u_h),\\
\boldsymbol{K}_{\sigma\sigma}\cdot\boldsymbol{\Sigma} \text{ which comes from the term } &-\frac{1}{nk}(\boldsymbol{\tau}_h,\boldsymbol{\sigma}_h),\\
\boldsymbol{K}_{\sigma u}\cdot\boldsymbol{U} \text{ which comes from the term } &+\frac{1}{n}(\boldsymbol{\tau}_h,\nabla u_h),\\
\boldsymbol{K}_{u\sigma}\cdot\boldsymbol{\Sigma} \text{ which comes from the term } &+\frac{1}{n}(\nabla v_h,\sigma_h),
\end{aligned} \tag{3.26}$$

as well as $\boldsymbol{G}_{u\sigma}$, $\boldsymbol{G}_{\sigma u}$, $\boldsymbol{g}_{\sigma\bar{u}}$ and $\boldsymbol{f}$:

$$\begin{aligned}
\boldsymbol{G}_{u\sigma}\cdot\boldsymbol{\Sigma} \text{ which comes from the term } &-\langle\boldsymbol{n}\cdot\boldsymbol{\sigma},v_h\rangle_\Gamma,\\
\boldsymbol{G}_{\sigma u}\cdot\boldsymbol{U} \text{ which comes from the term } &-\langle\boldsymbol{n}\cdot\boldsymbol{\tau}_h,u_h\rangle_\Gamma,\\
\boldsymbol{g}_{\sigma\bar{u}} \text{ which comes from the term } &-\langle\boldsymbol{n}\cdot\boldsymbol{\tau}_h,\bar{u}\rangle_\Gamma,\\
\boldsymbol{f} \text{ which comes from the term } &+(f,v_h).
\end{aligned} \tag{3.27}$$

The problem written in matrix form is:

$$\begin{bmatrix}(1-\frac{1}{n})\boldsymbol{K}_{uu} & \boldsymbol{K}_{u\sigma}+\boldsymbol{G}_{u\sigma}\\ \boldsymbol{K}_{\sigma u}+\boldsymbol{G}_{\sigma u} & \boldsymbol{K}_{\sigma\sigma}\end{bmatrix}\begin{bmatrix}\boldsymbol{U}\\ \boldsymbol{\Sigma}\end{bmatrix}=\begin{bmatrix}\boldsymbol{f}\\ \boldsymbol{g}_{\sigma\bar{u}}\end{bmatrix}. \tag{3.28}$$

We can compute the fluxes as:

$$\boldsymbol{\Sigma}=\boldsymbol{K}_{\sigma\sigma}^{-1}(-(\boldsymbol{K}_{\sigma u}+\boldsymbol{G}_{\sigma u})\cdot\boldsymbol{U}+\boldsymbol{g}_{\sigma\bar{u}}). \tag{3.29}$$

In elements cut by Γ, $\boldsymbol{K}_{\sigma\sigma}^{-1}$ is block diagonal, and therefore easy to invert due to the element-wise discontinuous stress approximation. This allows the condensation of the stress unknowns at the element level and we are left with only the original unknowns of the problem:

$$[(1-\frac{1}{n})\boldsymbol{K}_{uu}-(\boldsymbol{G}_{u\sigma}+\boldsymbol{K}_{u\sigma})\boldsymbol{K}_{\sigma\sigma}^{-1}(\boldsymbol{K}_{\sigma u}+\boldsymbol{G}_{\sigma u})]\cdot\boldsymbol{U}=[\boldsymbol{f}-(\boldsymbol{G}_{u\sigma}+\boldsymbol{K}_{u\sigma})\boldsymbol{K}_{\sigma\sigma}^{-1}\boldsymbol{g}_{\sigma\bar{u}}], \tag{3.30}$$

We now define the matrices $\boldsymbol{G}_{uu}^1$, $\boldsymbol{G}_{uu}^2$, $\boldsymbol{G}_{uu}^\alpha$ and the vectors $\boldsymbol{g}_{u\bar{u}}$ and $\boldsymbol{g}_{u\bar{u}}^\alpha$:

$$\begin{aligned}
\boldsymbol{G}_{uu}^1\cdot\boldsymbol{U} \text{ which comes from the term } &-k\langle\boldsymbol{n}\cdot\nabla u_h,v_h\rangle_\Gamma,\\
\boldsymbol{G}_{uu}^2\cdot\boldsymbol{U} \text{ which comes from the term } &-k\langle\boldsymbol{n}\cdot\nabla v_h,u_h\rangle_\Gamma,\\
\boldsymbol{G}_{uu}^\alpha\boldsymbol{U} \text{ which comes from the term } &+k\frac{\alpha}{h}\langle v_h,u_h\rangle_\Gamma,\\
\boldsymbol{g}_{u\bar{u}} \text{ which comes from the term } &-k\langle\boldsymbol{n}\cdot\nabla v_h,\bar{u}\rangle_\Gamma,\\
\boldsymbol{g}_{u\bar{u}}^\alpha \text{ which comes from the term } &+k\frac{\alpha}{h}\langle v_h,\bar{u}\rangle_\Gamma,
\end{aligned} \tag{3.31}$$

We will now see that the following equalities hold:

1.
$$\frac{1}{n}\boldsymbol{K}_{uu}\cdot\boldsymbol{U}=-[\boldsymbol{K}_{u\sigma}\boldsymbol{K}_{\sigma\sigma}^{-1}\boldsymbol{K}_{\sigma u}]\cdot\boldsymbol{U}. \tag{3.32}$$

Let us check this. If S_h is taken to be piecewise-discontinuous and rich enough we can deduce that $[\boldsymbol{K}_{\sigma\sigma}^{-1}\boldsymbol{K}_{\sigma u}]\cdot\boldsymbol{U}$ arises from the term :

$$-kP_{S_h}(\nabla u_h)=-k\nabla u_h, \tag{3.33}$$

where P_{S_h} is the L^2 projection into the stress space. And we can see now that $-[\boldsymbol{K}_{u\sigma}\boldsymbol{K}_{\sigma\sigma}^{-1}\boldsymbol{K}_{\sigma u}]\cdot\boldsymbol{U}$ arises from $\frac{1}{n}k(\nabla v_h,\nabla u_h)$, that is to say, is equal to $\frac{1}{n}\boldsymbol{K}_{uu}\cdot\boldsymbol{U}$.

2. Similarly:
$$\begin{aligned}\boldsymbol{G}_{uu}^{2}\cdot\boldsymbol{U}&=-[\boldsymbol{K}_{u\sigma}\boldsymbol{K}_{\sigma\sigma}^{-1}\boldsymbol{G}_{\sigma u}]\cdot\boldsymbol{U},\\ \boldsymbol{g}_{u\bar{u}}&=-[\boldsymbol{K}_{u\sigma}\boldsymbol{K}_{\sigma\sigma}^{-1}\boldsymbol{g}_{\sigma\bar{u}}].\end{aligned} \tag{3.34}$$

3. and:
$$\boldsymbol{G}_{uu}^{1}\cdot\boldsymbol{U}=-[\boldsymbol{G}_{u\sigma}\boldsymbol{K}_{\sigma\sigma}^{-1}\boldsymbol{K}_{\sigma u}]\cdot\boldsymbol{U}, \tag{3.35}$$

Taking these equalities into account we can write the matrix form (3.30) as:

$$[\boldsymbol{K}_{uu}+\boldsymbol{G}_{uu}^{1}+\boldsymbol{G}_{uu}^{2}-\boldsymbol{G}_{u\sigma}\boldsymbol{K}_{\sigma\sigma}^{-1}\boldsymbol{G}_{\sigma u}]\cdot\boldsymbol{U}=[\boldsymbol{f}+\boldsymbol{g}_{u\bar{u}}-\boldsymbol{G}_{u\sigma}\boldsymbol{K}_{\sigma\sigma}^{-1}\boldsymbol{g}_{\sigma\bar{u}}]. \tag{3.36}$$

Let us now write Nitsche's method for Poisson's problem, which is:

$$\begin{aligned}k(\nabla u_h,\nabla v_h)-k\langle\partial_n u_h,v_h\rangle_\Gamma-k\langle\partial_n v_h,u_h\rangle_\Gamma+k\frac{\alpha}{h}\langle u_h,v_h\rangle_\Gamma&\\ =\langle f,v_h\rangle_\Omega-k\langle\partial_n v_h,\bar{u}\rangle_\Gamma+k\frac{\alpha}{h}\langle\bar{u},v_h\rangle_\Gamma,&\end{aligned} \tag{3.37}$$

which we can write in matrix form as:

$$[\boldsymbol{K}_{uu}+\boldsymbol{G}_{uu}^{1}+\boldsymbol{G}_{uu}^{2}+\boldsymbol{G}_{uu}^{\alpha}]\cdot\boldsymbol{U}=[\boldsymbol{f}+\boldsymbol{g}_{u\bar{u}}+\boldsymbol{g}_{u\bar{u}}^{\alpha}]. \tag{3.38}$$

We can conclude that the only difference between the presented method and Nitsche's method is that we have replaced:

$$\boldsymbol{G}_{uu}^{\alpha}\cdot\boldsymbol{U}\text{ and }\boldsymbol{g}_{u\bar{u}}^{\alpha}, \tag{3.39}$$

by:

$$-\boldsymbol{G}_{u\sigma}\boldsymbol{K}_{\sigma\sigma}^{-1}\boldsymbol{G}_{\sigma u}\cdot\boldsymbol{U}\text{ and }-\boldsymbol{G}_{u\sigma}\boldsymbol{K}_{\sigma\sigma}^{-1}\boldsymbol{g}_{\sigma\bar{u}}. \tag{3.40}$$

We end up with a symmetric method which is identical to Nitsche's method (for a rich enough discontinuous stress field), except for the so called penalty term. The main advantage of our formulation is that no large penalty parameter has to be used in order to ensure stability since we can define a value for n for which the method is stable. Note that the penalty terms in (3.39) involve boundary terms, whereas in (3.40) they involve volume integrals.

3.3 Introducing convection

3.3.1 Problem statement

In this section we deal with the convection-diffusion equation. The problem to be solved is no longer symmetric, and additional terms are needed in order to ensure the stability of the final weak form. The problem we are considering in this section is:

$$-\nabla \cdot \boldsymbol{\sigma} + \boldsymbol{a} \cdot \nabla u = f \quad \text{in } \Omega, \tag{3.41}$$

$$\frac{1}{k}\boldsymbol{\sigma} = \nabla u \quad \text{in } \Omega, \tag{3.42}$$

$$u = \bar{u} \quad \text{on } \Gamma = \partial\Omega, \tag{3.43}$$

where $k > 0$, $\boldsymbol{a}$ is the advection velocity, f is a given forcing function and $\bar{u}$ is the given Dirichlet boundary condition. We have already used the two-field formulation presented for Poisson's problem.

3.3.2 Weak form

The variational form of the problem consists of finding $u_h \in V_h$ and $\boldsymbol{\sigma}_h \in S_h$ such that:

$$(1-\frac{1}{n})k(\nabla u_h, \nabla v_h) - \langle \boldsymbol{\sigma}_h \cdot \boldsymbol{n}, v_h\rangle_\Gamma + (\boldsymbol{a}\cdot\nabla u_h, v_h) + \frac{1}{n}(\nabla v_h, \boldsymbol{\sigma}_h) + \frac{1}{2}\langle a v_h, u_h\rangle_\Gamma$$
$$= \langle f, v_h\rangle_\Omega + \frac{1}{2}\langle a v_h, \bar{u}\rangle_\Gamma, \quad \forall v_h \in V_h, \tag{3.44}$$

$$-\frac{1}{nk}(\boldsymbol{\tau}_h, \boldsymbol{\sigma}_h) + \frac{1}{n}(\boldsymbol{\tau}_h, \nabla u_h) - \langle \boldsymbol{\tau}_h \cdot \boldsymbol{n}, u_h\rangle_\Gamma = -\langle \boldsymbol{\tau}_h \cdot \boldsymbol{n}, \bar{u}\rangle_\Gamma, \quad \forall \boldsymbol{\tau}_h \in S_h \tag{3.45}$$

Note that in the previous weak form we have replaced (3.8) with:

$$k(\nabla u_h, \nabla v_h) - \langle \boldsymbol{\sigma}_h \cdot \boldsymbol{n}, v_h\rangle_\Gamma + (\boldsymbol{a}\cdot\nabla u_h, v_h) = \langle f, v_h\rangle_\Omega, \tag{3.46}$$

and we have added:

$$\frac{1}{2}\langle a v_h, u_h\rangle_\Gamma = \frac{1}{2}\langle a v_h, \bar{u}\rangle_\Gamma, \tag{3.47}$$

which is weakly enforcing (3.3) tested against $\frac{a}{2}v_h$. We will see how to define a in the following sections.

It is observed that, apart from the way to impose the boundary conditions, (3.44)-(3.45) is based on the standard Galerkin method to solve the convection-diffusion-reaction equation. This method is stable only for high values of the diffusion coefficient k. Even though in the examples we will consider convection dominated flows solved using a stabilized formulation, for the sake of conciseness the exposition will be developed in the diffusion dominated case. Likewise, we will consider $\boldsymbol{a}$ constant, for simplicity.

3.3.3 Stability

In this subsection we prove that the formulation given by (3.44)-(3.45) is stable, and as a consequence has a unique solution.

We define the bilinear form on $[V_h, S_h] \times [V_h, S_h]$:

$$\begin{aligned} B^c([u_h, \boldsymbol{\sigma}_h], [v_h, \boldsymbol{\tau}_h]) =& (1 - \frac{1}{n})k(\nabla u_h, \nabla v_h) - \langle \boldsymbol{\sigma}_h \cdot \boldsymbol{n}, v_h \rangle_\Gamma + (\boldsymbol{a} \cdot \nabla u_h, v_h) + \frac{1}{n}(\nabla v_h, \boldsymbol{\sigma}_h) \\ &+ \frac{1}{2}\langle a v_h, u_h \rangle_\Gamma - \frac{1}{nk}(\boldsymbol{\tau}_h, \boldsymbol{\sigma}_h) + \frac{1}{n}(\boldsymbol{\tau}_h, \nabla u_h) - \langle \boldsymbol{\tau}_h \cdot \boldsymbol{n}, u_h \rangle_\Gamma \end{aligned} \tag{3.48}$$

Taking $[v_h, \boldsymbol{\tau}_h] = [u_h, -\boldsymbol{\sigma}_h - \frac{\beta}{h}k\tilde{\boldsymbol{\tau}}_h]$, where we have defined $\tilde{\boldsymbol{\tau}}_h$ as in the previous section, we have:

$$\begin{aligned} B^c([u_h, \boldsymbol{\sigma}_h],&[u_h, -\boldsymbol{\sigma}_h - \frac{\beta}{h}k\tilde{\boldsymbol{\tau}}_h]) = \\ &= (1 - \frac{1}{n})k(\nabla u_h, \nabla u_h) - \langle \boldsymbol{\sigma}_h \cdot \boldsymbol{n}, u_h \rangle_\Gamma + (\boldsymbol{a} \cdot \nabla u_h, u_h) + \frac{1}{n}(\nabla \boldsymbol{u}_h, \boldsymbol{\sigma}_h) \\ &\quad + \frac{1}{2}\langle a u_h, u_h \rangle_\Gamma + \frac{1}{nk}(\boldsymbol{\sigma}_h, \boldsymbol{\sigma}_h) - \frac{1}{n}(\boldsymbol{\sigma}_h, \nabla u_h) + \langle \boldsymbol{\sigma}_h \cdot \boldsymbol{n}, u_h \rangle_\Gamma \\ &\quad + \frac{\beta}{nh}(\tilde{\boldsymbol{\tau}}_h, \boldsymbol{\sigma}_h) - \frac{\beta k}{nh}(\tilde{\boldsymbol{\tau}}_h, \nabla u_h) + \frac{\beta k}{h}\langle \tilde{\boldsymbol{\tau}}_h \cdot \boldsymbol{n}, u_h \rangle_\Gamma \\ &\gtrsim \left(1 - \frac{1}{n} - \beta(\frac{3}{2n} + \delta_0)\right) k\|\nabla u_h\|^2 + \left(\frac{1}{kn} - \frac{\beta}{2kn}\right)\|\boldsymbol{\sigma}_h\|^2 + (1 - \frac{1}{n})\frac{\beta k}{h}\|u_h\|^2_{L^2(\Gamma)} \\ &\quad + \frac{1}{2}\int_\Gamma \boldsymbol{n} \cdot \boldsymbol{a} u_h^2 + \frac{1}{2}\int_\Gamma a u_h^2 \\ &= \left(1 - \frac{1}{n} - \beta(\frac{3}{2n} + \delta_0)\right) k\|\nabla u_h\|^2 + \left(\frac{1}{kn} - \frac{\beta}{2kn}\right)\|\boldsymbol{\sigma}_h\|^2 + (1 - \frac{1}{n})\frac{\beta k}{h}\|u_h\|^2_{L^2(\Gamma)} \\ &\quad + \frac{1}{2}\int_\Gamma (\boldsymbol{n} \cdot \boldsymbol{a} + a) u_h^2, \end{aligned} \tag{3.49}$$

where the same steps as in (3.22) have been carried out.

From here we can deduce that, as long as the following relation holds:

$$a + \boldsymbol{a} \cdot \boldsymbol{n} \geq 0, \tag{3.50}$$

then:

Theorem For all $[u_h, \boldsymbol{\sigma}_h]$there exist $[v_h, \boldsymbol{\tau}_h]$ and $\alpha > 0$ such that:

$$B([u_h, \boldsymbol{\sigma}_h], [v_h, \boldsymbol{\tau}_h]) \geq \alpha |\!|\!| (u_h, \boldsymbol{\sigma}_h) |\!|\!| |\!|\!| (v_h, \boldsymbol{\tau}_h) |\!|\!|. \tag{3.51}$$

Taking into account (3.50) the obvious definition for a is:

$$\begin{aligned} a = -\boldsymbol{a} \cdot \boldsymbol{n}, \quad \text{if } \boldsymbol{a} \cdot \boldsymbol{n} < 0 \\ a = 0, \quad \text{otherwise} \end{aligned} \tag{3.52}$$

The definition of the weighting term a is very similar to the one used in the weak imposition of boundary conditions in [15], the main difference being that in our case it is accompanied with a $\frac{1}{2}$ factor.

3.4 Extension to the Stokes problem

In this section we extend the previous ideas to the Stokes problem, for which we obtain a symmetric method again. Once the method is defined for the Stokes problem we can deal with the Navier-Stokes equations just by putting together to the formulation presented in Section 3.3 the one in the current section.

3.4.1 Problem statement

Let us consider the three-field formulation for the Stokes problem:

$$-\nu\Delta\boldsymbol{u} + \nabla p = \boldsymbol{f} \qquad \text{in } \Omega, \tag{3.53}$$

$$\nabla \cdot \boldsymbol{u} = \boldsymbol{0} \qquad \text{in } \Omega, \tag{3.54}$$

$$\frac{1}{\nu}\boldsymbol{\sigma} = \nabla\boldsymbol{u} \qquad \text{in } \Omega, \tag{3.55}$$

$$\boldsymbol{u} = \bar{\boldsymbol{u}} \qquad \text{on } \Gamma = \partial\Omega, \tag{3.56}$$

where $\nu > 0$, $\boldsymbol{f}$ is a given forcing function and $\bar{\boldsymbol{u}}$ is the given Dirichlet boundary condition. Note that $\boldsymbol{\sigma}$ only accounts for the deviatoric part of the pseudo-stresses (we could also formulate the method in terms of the strain rate tensor $\nabla^s\boldsymbol{u}$).

3.4.2 Weak form

Let us consider the finite element spaces $V_h \subset H^1(\Omega_h)^d$, $Q_h \subset L^2(\Omega_h)$ (we will consider V_h and Q_h made of continuous functions) and $S_h \subset L^2(\Omega_h)^{d\times d}$ (we will consider S_h made of element-wise discontinuous functions). As stated in the previous chapter the standard finite element approximation of the Stokes problem is not stable for an arbitrary $\boldsymbol{u}$, p interpolation. This is the reason why we add stabilization terms to the original weak form of the problem, which allow us to use equal interpolation spaces for velocity and pressure. Our stabilized symmetric variational form of the problem consists of finding $\boldsymbol{u}_h \in V_h$, $p_h \in Q_h$ and $\boldsymbol{\sigma}_h \in S_h$ such that:

$$\begin{aligned}(1-\frac{1}{n})\nu(\nabla\boldsymbol{u}_h,\nabla\boldsymbol{v}_h) - (\nabla\cdot\boldsymbol{v}_h,p_h) - \langle\boldsymbol{\sigma}_h\cdot\boldsymbol{n},\boldsymbol{v}_h\rangle_\Gamma + \langle\boldsymbol{n}\cdot\boldsymbol{v}_h,p_h\rangle_\Gamma \\ - \sum_K \tau_K(\nu\Delta\boldsymbol{v}_h,\nu\Delta\boldsymbol{u}_h)_K + \sum_K \tau_K(\nu\Delta\boldsymbol{v}_h,\nabla p_h)_K + \frac{1}{n}(\nabla\boldsymbol{v}_h,\boldsymbol{\sigma}_h) \\ = \langle\boldsymbol{f},\boldsymbol{v}_h\rangle_\Omega + \sum_K \tau_K(\nu\Delta\boldsymbol{v}_h,\boldsymbol{f})_K, \quad \forall\boldsymbol{v}_h\in V_h,\end{aligned} \tag{3.57}$$

$$\begin{aligned}-(q_h,\nabla\cdot\boldsymbol{u}_h) + \sum_K \tau_K(\nabla q_h,\nu\Delta\boldsymbol{u}_h)_K - \sum_K \tau_K(\nabla q_h,\nabla p_h)_K + \langle q_h,\boldsymbol{n}\cdot\boldsymbol{u}_h\rangle_\Gamma \\ = -\sum_K \tau_K(\nabla q_h,\boldsymbol{f})_K + \langle q_h,\boldsymbol{n}\cdot\bar{\boldsymbol{u}}\rangle_\Gamma, \quad \forall q_h\in Q_h,\end{aligned} \tag{3.58}$$

$$-\frac{1}{n\nu}(\boldsymbol{\tau}_h,\boldsymbol{\sigma}_h) + \frac{1}{n}(\boldsymbol{\tau}_h,\nabla\boldsymbol{u}_h) - \langle\boldsymbol{\tau}_h\cdot\boldsymbol{n},\boldsymbol{u}_h\rangle_\Gamma = -\langle\boldsymbol{\tau}_h\cdot\boldsymbol{n},\bar{\boldsymbol{u}}\rangle_\Gamma, \quad \forall\boldsymbol{\tau}_h\in S_h \tag{3.59}$$

In this terms $(\cdot,\cdot)_K \equiv (\cdot,\cdot)_{L^2(K)}$ represents integrals over each element, and we define the stabilizing parameter τ_K as:

$$\tau_K = \left(c_1 \frac{\nu}{h^2}\right)^{-1}, \tag{3.60}$$

in each element, where h is the element size. For the numerical experiments we have taken $c_1 = 4$.

Note that there are several overlapping dependent equations in the previous variational form:

$$\nu(\nabla \boldsymbol{u}_h, \nabla \boldsymbol{v}_h) - (\nabla \cdot \boldsymbol{v}_h, p_h) - \langle \boldsymbol{\sigma}_h \cdot \boldsymbol{n}, \boldsymbol{v}_h\rangle_\Gamma + \langle \boldsymbol{n} \cdot \boldsymbol{v}_h, p_h\rangle_\Gamma = \langle \boldsymbol{f}, \boldsymbol{v}_h\rangle_\Omega, \tag{3.61}$$

$$-\sum_K \tau_K(\nu\Delta \boldsymbol{v}_h, \nu\Delta \boldsymbol{u}_h)_K + \sum_K \tau_K(\nu\Delta \boldsymbol{v}_h, \nabla p_h)_K = \sum_K \tau_K(\nu\Delta \boldsymbol{v}_h, \boldsymbol{f})_K, \tag{3.62}$$

$$\frac{1}{n}(\nabla \boldsymbol{v}_h, \boldsymbol{\sigma}_h) - \frac{1}{n}\nu(\nabla \boldsymbol{u}_h, \nabla \boldsymbol{v}_h) = \boldsymbol{0}, \tag{3.63}$$

$$-(q_h, \nabla \cdot \boldsymbol{u}_h) = 0, \tag{3.64}$$

$$\sum_K \tau_K(\nabla q_h, \nu\Delta \boldsymbol{u}_h)_K - \sum_K \tau_K(\nabla q_h, \nabla p_h)_K = -\sum_K \tau_K(\nabla q_h, \boldsymbol{f})_K, \tag{3.65}$$

$$\langle q_h, \boldsymbol{n} \cdot \boldsymbol{u}_h\rangle_\Gamma = \langle q_h, \boldsymbol{n} \cdot \bar{\boldsymbol{u}}\rangle_\Gamma \tag{3.66}$$

$$-\frac{1}{n\nu}(\boldsymbol{\tau}_h, \boldsymbol{\sigma}_h) + \frac{1}{n}(\boldsymbol{\tau}_h, \nabla \boldsymbol{u}_h) = \boldsymbol{0}, \tag{3.67}$$

$$-\langle \boldsymbol{\tau}_h \cdot \boldsymbol{n}, \boldsymbol{u}_h\rangle_\Gamma = -\langle \boldsymbol{\tau}_h \cdot \boldsymbol{n}, \bar{\boldsymbol{u}}\rangle_\Gamma. \tag{3.68}$$

(3.61) is weakly enforcing (3.53) tested against $\boldsymbol{v}_h$. (3.64) is weakly enforcing (3.54) tested against $-q_h$. (3.63) and (3.67) are weakly enforcing (3.55) tested against $\frac{1}{n}\nu\nabla \boldsymbol{v}_h$ and $-\frac{1}{n}\boldsymbol{\tau}_h$ respectively. (3.68) is weakly enforcing (3.56) tested against $-\boldsymbol{\tau}_h \cdot \boldsymbol{n}$.

(3.66) is weakly imposing (3.56) tested against $q_h\boldsymbol{n}$. This term is added in order to keep the method symmetric and also in order to be able to prove stability.

Finally (3.62) and (3.65) are the stabilizing terms for the Stokes problem, which are independent of the way boundary conditions are imposed.

3.4.3 Stability

In this subsection we prove that the formulation given by (3.57)-(3.59) is stable, and as a consequence has a unique solution. We define the norm:

$$|\!|\!|(\boldsymbol{u}, p, \boldsymbol{\sigma})|\!|\!|^2 = \nu\|\nabla \boldsymbol{u}\|^2 + \frac{\nu}{h}\|\boldsymbol{u}\|^2_{L^2(\Gamma)} + \frac{h^2}{\nu}\|\nabla p\|^2 + \frac{1}{\nu}\|\boldsymbol{\sigma}\|^2. \tag{3.69}$$

We define the bilinear form on $[V_h, Q_h, S_h] \times [V_h, Q_h, S_h]$:

$$
\begin{aligned}
&B^s([\boldsymbol{u}_h, p_h, \boldsymbol{\sigma}_h], [\boldsymbol{v}_h, q_h, \boldsymbol{\tau}_h]) = \\
&+ (1 - \frac{1}{n})\nu(\nabla \boldsymbol{u}_h, \nabla \boldsymbol{v}_h) - (\nabla \cdot \boldsymbol{v}_h, p_h) - \langle \boldsymbol{\sigma}_h \cdot \boldsymbol{n}, \boldsymbol{v}_h \rangle_\Gamma + \langle \boldsymbol{n} \cdot \boldsymbol{v}_h, p_h \rangle_\Gamma - \sum_K \tau_K (\nu \Delta \boldsymbol{v}_h, \nu \Delta \boldsymbol{u}_h)_K \\
&+ \sum_K \tau_K (\nu \Delta \boldsymbol{v}_h, \nabla p_h)_K + \frac{1}{n}(\nabla \boldsymbol{v}_h, \boldsymbol{\sigma}_h) - (q_h, \nabla \cdot \boldsymbol{u}_h) + \sum_K \tau_K (\nabla q_h, \nu \Delta \boldsymbol{u}_h)_K \\
&- \sum_K \tau_K (\nabla q_h, \nabla p_h)_K + \langle q_h, \boldsymbol{n} \cdot \boldsymbol{u}_h \rangle_\Gamma - \frac{1}{n\nu}(\boldsymbol{\tau}_h, \boldsymbol{\sigma}_h) + \frac{1}{n}(\boldsymbol{\tau}_h, \nabla \boldsymbol{u}_h) - \langle \boldsymbol{\tau}_h \cdot \boldsymbol{n}. \boldsymbol{u}_h \rangle_\Gamma \quad (3.70)
\end{aligned}
$$

We now suppose that V_h and S_h are such that the following conditions hold for all the elements cut by the boundary Γ:

$$
\forall \boldsymbol{v}_h \in V_h \quad \exists \boldsymbol{\tau}_h \in S_h | \quad \|\boldsymbol{v}_h\|^2_{L^2(\Gamma)} \lesssim \langle \boldsymbol{\tau}_h \cdot \boldsymbol{n}, \boldsymbol{v}_h \rangle_\Gamma + \delta_0 h \|\nabla \boldsymbol{v}_h\|^2, \quad \|\boldsymbol{\tau}_h\|_{L^2(K)} = \|\boldsymbol{v}_h\|_{L^2(K)} \tag{3.71}
$$

Taking $[\boldsymbol{v}_h, q_h, \boldsymbol{\tau}_h] = [\boldsymbol{u}_h, -p_h, -\boldsymbol{\sigma}_h - \frac{\beta}{h}\nu\tilde{\boldsymbol{\tau}}_h]$, where $\tilde{\boldsymbol{\tau}}_h$ is the counterpart of $\boldsymbol{u}_h$ in (3.71), we have:

$$
\begin{aligned}
&B^s([\boldsymbol{u}_h, p_h, \boldsymbol{\sigma}_h], [\boldsymbol{u}_h, -p_h, -\boldsymbol{\sigma}_h - \frac{\beta}{h}\nu\tilde{\boldsymbol{\tau}}_h]) = \\
&+ (1 - \frac{1}{n})\nu(\nabla \boldsymbol{u}_h, \nabla \boldsymbol{u}_h) - (\nabla \cdot \boldsymbol{u}_h, p_h) - \langle \boldsymbol{\sigma}_h \cdot \boldsymbol{n}, \boldsymbol{u}_h \rangle_\Gamma + \langle \boldsymbol{n} \cdot \boldsymbol{u}_h, p_h \rangle_\Gamma - \sum_K \tau_K (\nu \Delta \boldsymbol{u}_h, \nu \Delta \boldsymbol{u}_h)_K \\
&+ \sum_K \tau_K (\nu \Delta \boldsymbol{u}_h, \nabla p_h)_K + \frac{1}{n}(\nabla \boldsymbol{u}_h, \boldsymbol{\sigma}_h) + (p_h, \nabla \cdot \boldsymbol{u}_h) - \sum_K \tau_K (\nabla p_h, \nu \Delta \boldsymbol{u}_h)_K \\
&+ \sum_K \tau_K (\nabla p_h, \nabla p_h)_K - \langle p_h, \boldsymbol{n} \cdot \boldsymbol{u}_h \rangle_\Gamma + \frac{1}{n\nu}(\boldsymbol{\sigma}_h, \boldsymbol{\sigma}_h) - \frac{1}{n}(\boldsymbol{\sigma}_h, \nabla \boldsymbol{u}_h) + \langle \boldsymbol{\sigma}_h \cdot \boldsymbol{n}, \boldsymbol{u}_h \rangle_\Gamma \\
&+ \frac{\beta}{nh}(\tilde{\boldsymbol{\tau}}_h, \boldsymbol{\sigma}_h) - \frac{\beta\nu}{nh}(\tilde{\boldsymbol{\tau}}_h, \nabla \boldsymbol{u}_h) + \frac{\beta\nu}{h}\langle \tilde{\boldsymbol{\tau}}_h \cdot \boldsymbol{n}, \boldsymbol{u}_h \rangle_\Gamma \\
\gtrsim &\left(1 - \frac{1}{n} - \beta(\frac{3}{2n} + \delta_0)\right) \nu \|\nabla \boldsymbol{u}_h\|^2 - \sum_K \tau_K \|\nu \Delta \boldsymbol{u}_h\|^2_K + (1 - \frac{1}{n})\frac{\beta\nu}{h}\|\boldsymbol{u}_h\|^2_{L^2(\Gamma)} \\
&+ \sum_K \tau_K \|\nabla p_h\|^2_K + \left(\frac{1}{\nu n} - \frac{\beta}{2\nu n}\right) \|\boldsymbol{\sigma}_h\|^2 \\
\geq &\left(1 - \frac{1}{n} - \beta(\frac{3}{2n} + \delta_0) - C_1\right) \nu \|\nabla \boldsymbol{u}_h\|^2 + (1 - \frac{1}{n})\frac{\beta\nu}{h}\|\boldsymbol{u}_h\|^2_{L^2(\Gamma)} \\
&+ C_2 \frac{h^2}{\nu}\|\nabla p_h\|^2 + \left(\frac{1}{\nu n} - \frac{\beta}{2\nu n}\right) \|\boldsymbol{\sigma}_h\|^2, \qquad (3.72)
\end{aligned}
$$

where C_1 is such that, making use of the inverse estimates:

$$
\sum_K \tau_K \|\nu \Delta \boldsymbol{u}_h\|^2_K \leq C_1 \nu \|\nabla \boldsymbol{u}_h\|^2, \tag{3.73}
$$

and C_2 is defined as $\frac{1}{c_1}$ in (3.60). Imposing:

$$n > 1, \quad \beta < \min\left(\frac{1-\frac{1}{n}}{(\frac{3}{2n}+\delta_0)}, 2\right), \tag{3.74}$$

we can now define the constant:

$$\alpha_0 = \min\left(1 - \frac{1}{n} - \beta(\frac{3}{2n}+\delta_0) - C_1, C_2, \frac{1}{n} - \frac{\beta}{2n}, 1 - \frac{1}{n}\right), \tag{3.75}$$

which is positive for a sufficiently small stability parameter τ_K. We have now proved that:

$$B([\boldsymbol{u}_h, p_h, \boldsymbol{\sigma}_h], [\boldsymbol{u}_h, -p_h, -\boldsymbol{\sigma}_h - \frac{\beta}{h}\nu\tilde{\boldsymbol{\tau}}_h]]) \gtrsim \alpha_0 |\!|\!| (\boldsymbol{u}_h, p_h, \boldsymbol{\sigma}_h) |\!|\!|^2, \tag{3.76}$$

We now take into account that:

$$\begin{aligned}
|\!|\!| (\boldsymbol{u}_h, p_h, -\boldsymbol{\sigma}_h - \frac{\beta}{h}k\tilde{\boldsymbol{\tau}}_h) |\!|\!|^2 &= k\|\nabla \boldsymbol{u}_h\|^2 + \frac{k}{h}\|\boldsymbol{u}_h\|^2_{L^2(\Gamma)} + \frac{h^2}{\nu}\|\nabla p_h\|^2 + \frac{1}{k}\| - \boldsymbol{\sigma}_h - \frac{\beta}{h}k\tilde{\boldsymbol{\tau}}_h\|^2 \\
&\leq k\|\nabla \boldsymbol{u}_h\|^2 + \frac{k}{h}\|\boldsymbol{u}_h\|^2_{L^2(\Gamma)} + \frac{h^2}{\nu}\|\nabla p_h\|^2 + \frac{1}{k}\|\boldsymbol{\sigma}_h\|^2 + \frac{k\beta^2}{h^2}\|\tilde{\boldsymbol{\tau}}_h\|^2 \\
&= k\|\nabla \boldsymbol{u}_h\|^2 + \frac{k}{h}\|\boldsymbol{u}_h\|^2_{L^2(\Gamma)} + \frac{h^2}{\nu}\|\nabla p_h\|^2 + \frac{1}{k}\|\boldsymbol{\sigma}_h\|^2 + \frac{k\beta^2}{h^2}\|\boldsymbol{u}_h\|^2 \\
&\lesssim k(1+\beta^2)\|\nabla \boldsymbol{u}_h\|^2 + \frac{k}{h}\left(1+\beta^2\right)\|\boldsymbol{u}_h\|^2_{L^2(\Gamma)} + \frac{h^2}{\nu}\|\nabla p_h\|^2 + \frac{1}{k}\|\boldsymbol{\sigma}_h\|^2,
\end{aligned} \tag{3.77}$$

which allows us to obtain the result we were looking for:

Theorem For all $[\boldsymbol{u}_h, p_h, \boldsymbol{\sigma}_h]$ there exist $[\boldsymbol{v}_h, q_h, \boldsymbol{\tau}_h]$ and $\alpha > 0$ such that:

$$B([\boldsymbol{u}_h, p_h, \boldsymbol{\sigma}_h], [\boldsymbol{v}_h, q_h, \boldsymbol{\tau}_h]) \geq \alpha |\!|\!| (\boldsymbol{u}_h, p_h, \boldsymbol{\sigma}_h) |\!|\!| |\!|\!| (\boldsymbol{v}_h, q_h, \boldsymbol{\tau}_h) |\!|\!|. \tag{3.78}$$

3.4.4 Implementation and comparison to Nitsche's method

Equalities similar to (3.32)-(3.35) hold also for the Stokes problem. We can now write down the weak form of the problem as it results if Nitsche's method is used to impose boundary conditions, which is:

$$\begin{aligned}
\nu(\nabla \boldsymbol{u}_h, \nabla \boldsymbol{v}_h) - (\nabla \cdot \boldsymbol{v}_h, p_h) - \nu\langle \nabla \boldsymbol{u}_h \cdot \boldsymbol{n}, \boldsymbol{v}_h\rangle_\Gamma - \nu\langle \nabla \boldsymbol{v}_h \cdot \boldsymbol{n}, \boldsymbol{u}_h\rangle_\Gamma + \langle \boldsymbol{n}\cdot\boldsymbol{v}_h, p_h\rangle_\Gamma \\
- \sum_K \tau_K(\nu\Delta \boldsymbol{v}_h, \nu\Delta \boldsymbol{u}_h)_K + \sum_K \tau_K(\nu\Delta \boldsymbol{v}_h, \nabla p_h)_K + \nu\frac{\alpha}{h}\langle \boldsymbol{u}_h, \boldsymbol{v}_h\rangle_\Gamma \\
= \langle \boldsymbol{f}, \boldsymbol{v}_h\rangle_\Omega + \sum_K \tau_K(\nu\Delta \boldsymbol{v}_h, \boldsymbol{f})_K - \nu\langle \nabla \boldsymbol{v}_h \cdot \boldsymbol{n}, \bar{\boldsymbol{u}}_h\rangle_\Gamma + \nu\frac{\alpha}{h}\langle \bar{\boldsymbol{u}}_h, \boldsymbol{v}_h\rangle_\Gamma, \quad \forall \boldsymbol{v}_h \in V_h,
\end{aligned} \tag{3.79}$$

$$\begin{aligned}
-(q_h, \nabla \cdot \boldsymbol{u}_h) + \sum_K \tau_K(\nabla q_h, \nu\Delta \boldsymbol{u}_h)_K - \sum_K \tau_K(\nabla q_h, \nabla p_h)_K + \langle q_h, \boldsymbol{n}\cdot\boldsymbol{u}_h\rangle_\Gamma \\
= -\sum_K \tau_K(\nabla q_h, \boldsymbol{f})_K + \langle q_h, \boldsymbol{n}\cdot\bar{\boldsymbol{u}}\rangle_\Gamma, \quad \forall q_h \in Q_h.
\end{aligned} \tag{3.80}$$

Making use of (3.32)-(3.35) we can conclude that after condensing the stress field at the element level, the presented method is very similar to Nitsche's method, the main difference being in the definition of the so-called penalty term, for which we do not need a large penalty parameter in order to ensure stability.

3.5 Transient Navier-Stokes equations

The proposed method for imposing boundary conditions in the transient Navier-Stokes equations consists simply in putting together the terms appearing in the convection-diffusion equation with the ones in the Stokes problem. As in the Stokes problem a stabilized formulation is required so that equal interpolations for the velocity and the pressure can be used. Moreover, additional stabilization terms are added so that we can deal with convection-dominated problems.

3.5.1 Problem statement

Let us consider the three-field formulation for the transient Navier-Stokes equations:

$$\partial_t \boldsymbol{u} - \nu \Delta \boldsymbol{u} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} + \nabla p = \boldsymbol{f} \qquad \text{in } \Omega, \tag{3.81}$$

$$\nabla \cdot \boldsymbol{u} = \boldsymbol{0} \qquad \text{in } \Omega, \tag{3.82}$$

$$\frac{1}{\nu}\boldsymbol{\sigma} = \nabla \boldsymbol{u} \qquad \text{in } \Omega, \tag{3.83}$$

$$\boldsymbol{u} = \bar{\boldsymbol{u}} \qquad \text{on } \Gamma = \partial\Omega, \tag{3.84}$$

in Ω and for $t > 0$, where $\boldsymbol{f}$ is the vector of body forces and ν the kinematic viscosity and $\partial_t \boldsymbol{u}$ is the local time derivative of the velocity field. Appropriate initial conditions have to be appended to this problem.

3.5.2 Weak form

The variational form of the problem consists of finding $\boldsymbol{u}_h \in V_h$, $p_h \in Q_h$ and $\boldsymbol{\sigma}_h \in S_h$ such that:

$$
\begin{aligned}
(\boldsymbol{v}_h, \partial_t \boldsymbol{u}_h) + (1-\frac{1}{n})\nu(\nabla \boldsymbol{u}_h, \nabla \boldsymbol{v}_h) + (\boldsymbol{v}_h, \boldsymbol{u}_h \cdot \nabla \boldsymbol{u}_h) - (\nabla \cdot \boldsymbol{v}_h, p_h) - \langle \boldsymbol{\sigma}_h \cdot \boldsymbol{n}, \boldsymbol{v}_h \rangle_\Gamma \\
+\langle \boldsymbol{n} \cdot \boldsymbol{v}_h, p_h \rangle_\Gamma + \sum_K \tau_K \langle \nu \Delta \boldsymbol{v}_h + \boldsymbol{u}_h \cdot \nabla \boldsymbol{v}_h, \partial_t \boldsymbol{u}_h - \nu \Delta \boldsymbol{u}_h + \boldsymbol{u}_h \cdot \nabla \boldsymbol{u}_h + \nabla p_h \rangle_K \\
+\frac{1}{n}(\nabla \boldsymbol{v}_h, \boldsymbol{\sigma}_h) + \frac{1}{2}\langle a \boldsymbol{v}_h, \boldsymbol{u}_h \rangle_\Gamma \\
= \langle \boldsymbol{f}, \boldsymbol{v}_h \rangle_\Omega + \sum_K \tau_K \langle \nu \Delta \boldsymbol{v}_h + \boldsymbol{u}_h \cdot \nabla \boldsymbol{v}_h, \boldsymbol{f} \rangle_K + \frac{1}{2}\langle a \boldsymbol{v}_h, \bar{\boldsymbol{u}} \rangle_\Gamma, \quad \forall \boldsymbol{v}_h \in V_h, \\
-(q_h, \nabla \cdot \boldsymbol{u}_h) + \sum_K \tau_K (\nabla q_h, \partial_t \boldsymbol{u}_h - \nu \Delta \boldsymbol{u}_h + \boldsymbol{u}_h \cdot \nabla \boldsymbol{u}_h + \nabla p_h)_K + \langle q_h, \boldsymbol{n} \cdot \boldsymbol{u}_h \rangle_\Gamma \\
= -\sum_K \tau_K (\nabla q_h, \boldsymbol{f})_K + \langle q_h, \boldsymbol{n} \cdot \bar{\boldsymbol{u}} \rangle_\Gamma, \quad \forall q_h \in Q_h, \\
-\frac{1}{n\nu}(\boldsymbol{\tau}_h, \boldsymbol{\sigma}_h) + \frac{1}{n}(\boldsymbol{\tau}_h, \nabla \boldsymbol{u}_h) - \langle \boldsymbol{\tau}_h \cdot \boldsymbol{n}, \boldsymbol{u}_h \rangle_\Gamma = -\langle \boldsymbol{\tau}_h \cdot \boldsymbol{n}, \bar{\boldsymbol{u}} \rangle_\Gamma, \quad \forall \boldsymbol{\tau}_h \in S_h
\end{aligned}
\tag{3.85}
$$

and now the stabilization parameter is computed as

$$
\tau_K = \left(c_1 \frac{\nu}{h^2} + c_2 \frac{|\boldsymbol{u}_h|_K}{h} \right)^{-1},
$$

where $|\boldsymbol{u}_h|_K$ is the mean velocity modulus in element K. The stability constants are defined as $c_1 = 4$ and $c_2 = 2$. Any finite difference scheme can be used to approximate the time derivative $\partial_t \boldsymbol{u}_h$.

3.6 Numerical examples

3.6.1 Convection-diffusion equation

In this subsection we illustrate the behavior of the proposed method for the scalar convection-diffusion equation. The Poisson, and convection-diffusion equations are solved in a domain Ω enclosed in a circle of radius $R < 1$. We choose the *hold-all domain* $B = (-1, 1) \times (-1, 1)$, where a system of Cartesian coordinates (x, y) with its origin at the center of the circle has been adopted. A structured mesh of right-angled linear triangular elements is constructed in *B*, *h* being the length of the edges corresponding to the cathetus.

The Poisson equation

Let us start solving the Poisson equation with $k = 1$, $\boldsymbol{a} = \boldsymbol{0}$, $f = 1$ to check the performance and convergence of the proposed method. The analytical solution for this case is known to be

$$
u(x, y) = \frac{1}{4}(R^2 - x^2 - y^2).
$$

Fig. 3.1 shows the errors $\|u - u_h\|_{L^2(\Omega)}$ versus the element size h. The coarsest mesh is built of 1250 elements, while the finer ones is built out of 320000 elements. The method shows quadratic convergence when linear elements are used. When compared to the strong imposition of boundary conditions described in Chapter 2, the error obtained is smaller in the current method although both methods show quadratic convergence.

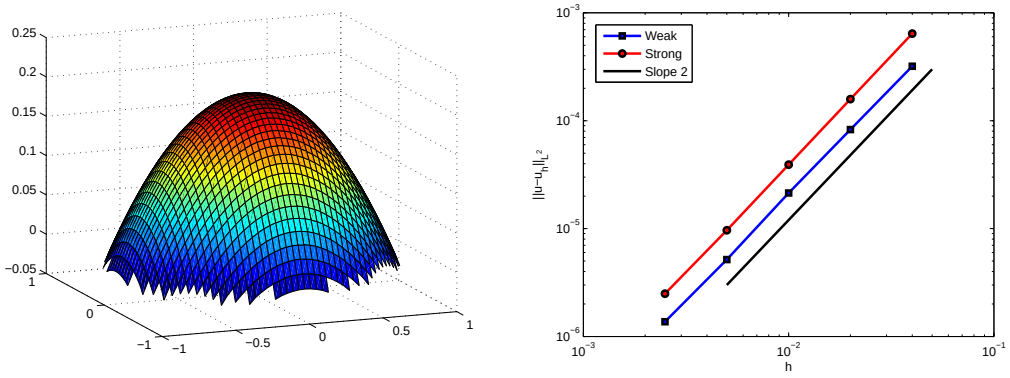


Figure 3.1: Results and error convergence for the solution of the diffusion equation

Convection-diffusion

Fig. 3.2 shows the behaviour of the method in the convection-diffusion problem, where $k = 10^{-2}$, $\boldsymbol{a} = (1, 0)$, $f = 1$. A stabilized formulation similar to the one described in Section 3.5 has been used. In order to obtain the error we have computed the solution for a very fine mesh (h = 2/800, 1200000 elements) which we have used as the reference solution. Again, the method shows quadratic convergence, although the strong imposition of boundary conditions performs slightly better in the convection-diffusion equation.

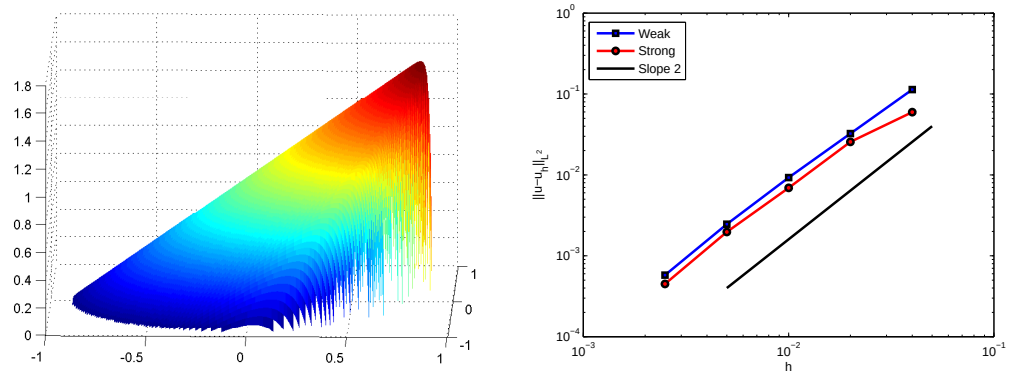


Figure 3.2: Results and error convergence for the solution of the convection-diffusion equation

It is also interesting to observe how the method behaves in strongly convection dominated problems. In Fig. 3.3 the solution for the problem with different viscosities are compared. We can observe that when convection grows larger and the corresponding boundary layer becomes

thinner, the mesh is no longer capable of capturing the boundary layer geometry. Due to the fact that the weak formulation does not weight the boundary conditions against convection velocity in the outflow boundary, we can observe that there are no spurious oscillations in the outflow boundary layer. As viscosity grows smaller the solution of the problem resembles the solution of the pure transport equation, where no boundary conditions are imposed on the outflow.

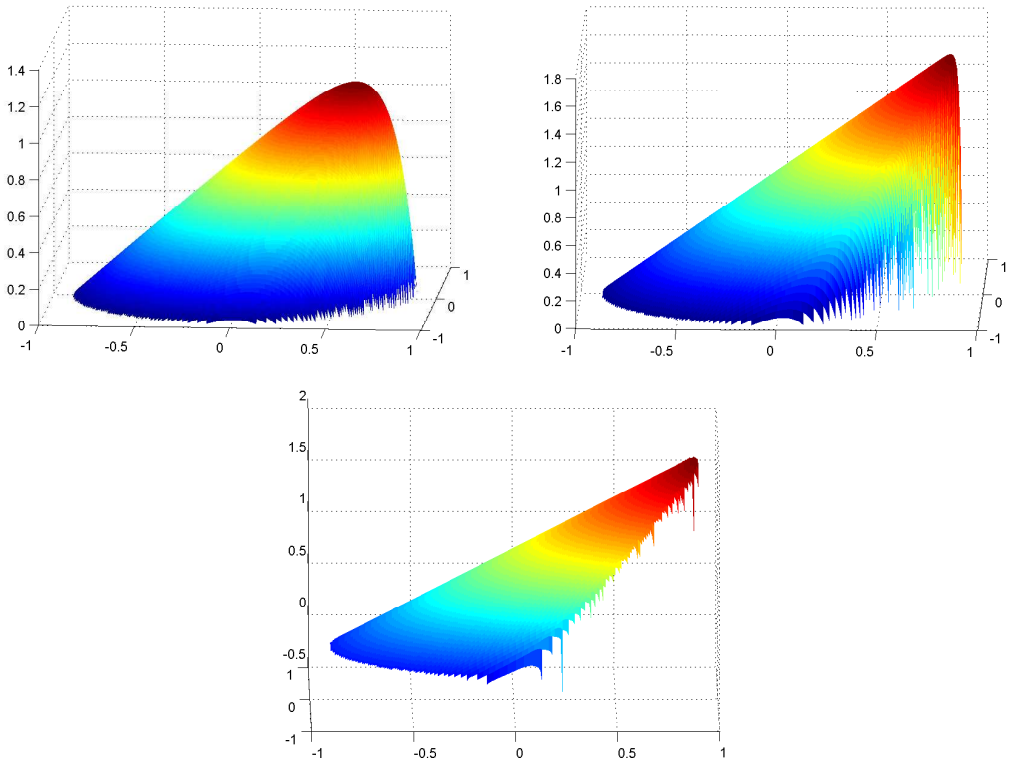


Figure 3.3: Solution comparison for the convection-diffusion equation with viscosities 10^{-1}, 10^{-2} and 10^{-5} .

3.6.2 Stokes problem

In this section we solve the Stokes problem and we check the convergence properties of the proposed method. We study the stationary Stokes flow around a cylinder. We use linear interpolations both for the velocity and for the pressure and the stabilized formulation proposed in the previous sections. The setting of the problem is shown in Fig. 3.4. A parabolic inflow profile with unitary mean horizontal velocity is set on $x = 0$. Velocity is prescribed to zero on $y = 0$ and $y = 1$ and on the cylindric boundary. The proposed method for weakly imposing boundary conditions has been used both in the immersed cylindrical interface and in the external grid matching boundaries.

In Fig. 3.5 velocity and pressure fields for a fine mesh are shown. In Fig. 3.6 we have plotted the error versus the mesh size, both for the velocity and for the pressure fields. The coarsest

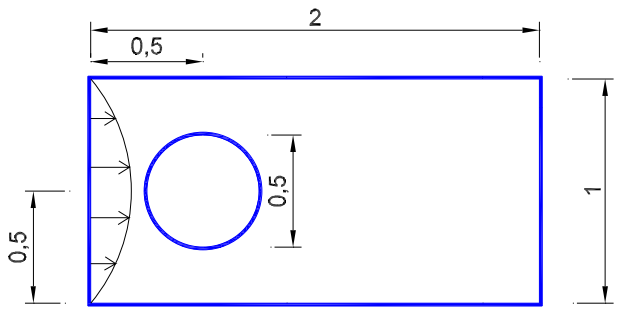


Figure 3.4: Geometry and boundary conditions for the Stokes flow around a cylinder.

mesh is built of 625 elements, while the finer one is built out of 40000 elements. Again, results for each mesh size have been compared against results in a much finer mesh (160000 elements). We can see that quadratic convergence rates are obtained in both cases. When compared to the strong imposition of boundary conditions, we can conclude that both methods perform equally well, except for the coarsest mesh case, in which the strong imposition of boundary conditions performs slightly better.

3.6.3 Transient Navier-Stokes equations

In this section we deal with the transient incompressible Navier-Stokes equations. As in the previous subsection, we will solve the flow around a cylinder, although the overall domain is larger in this case in order to allow the development of the vortices which arise behind the cylinder. The setting of the problem is depicted in Fig. 3.7. A parabolic inflow profile with mean horizontal velocity equal to 1 is set on $x = 0$. Velocity is prescribed to zero at $y = 0$, $y = 8$ and the cylindric boundary. The proposed method for weakly imposing boundary conditions has been used both in the immersed cylindrical interface and in the external grid matching boundaries. Viscosity has been set to $\nu = 10^{-2}$, which yields a Reynolds number $Re = 100$ based on the cylinder diameter and the mean inflow velocity. A backward Euler scheme has been used for the time integration with time step $\delta t = 0.2$. A 12566 linear element mesh has been used to solve the problem. The mesh has been refined in the area around the cylinder, but it is still a rather coarse mesh in which the length of the cylinder is only 12 times the element length.

In Fig. 3.8 velocity and pressure fields at the end of the simulation ($t = 100$) are shown. Fully developed vortices behind the cylinder, and a smooth solution around the immersed boundary can be appreciated. Fig. 3.9 shows the time history of the vertical velocity at a point behind the cylinder $(10, 4)$. After the initial transitory stage, an oscillatory pattern of amplitude 0.6 and period 4.9 is established. When compared to the results of the strong imposition of boundary conditions, we can see that both methods yield very similar results, the weak method presenting slightly larger amplitudes in both the initial transitory and the periodic stages.

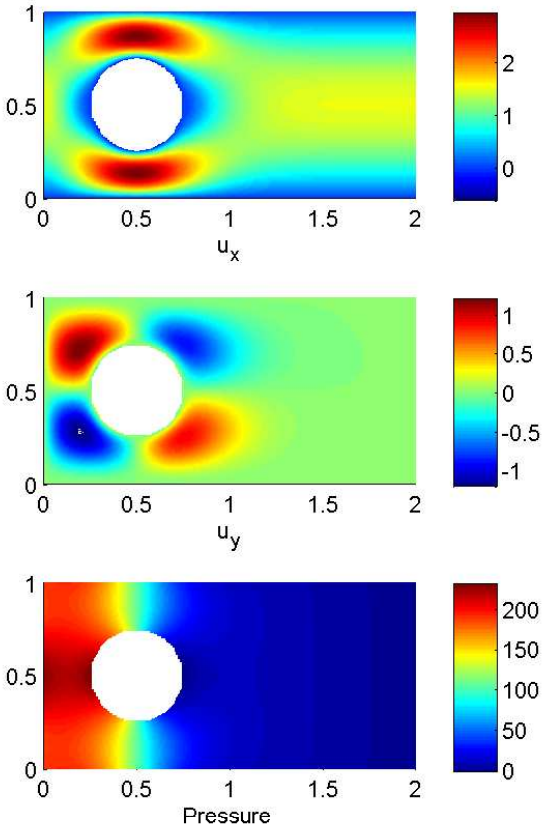


Figure 3.5: Velocity and pressure fields for the Stokes flow around a cylinder.

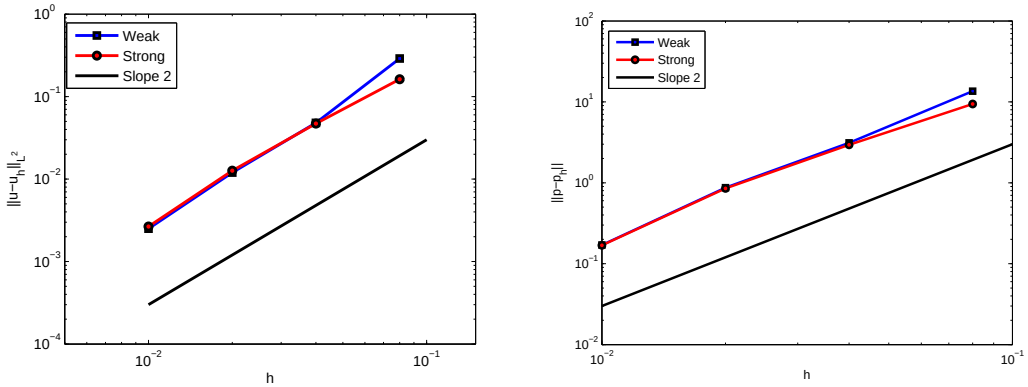


Figure 3.6: Convergence plots for the velocity (left) and pressure (right) fields in the Stokes flow around a cylinder.

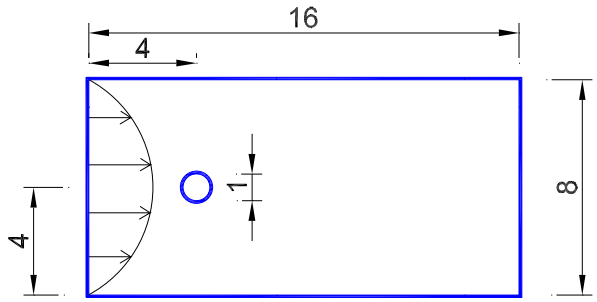


Figure 3.7: Geometry and boundary conditions for the transient Navier-Stokes flow around a cylinder.

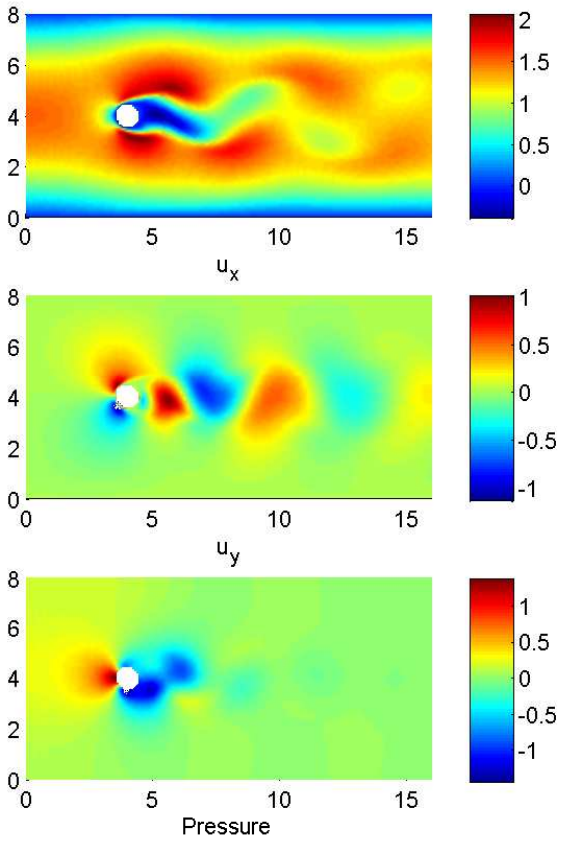


Figure 3.8: Velocity and pressure fields for the transient Navier-Stokes flow around a cylinder. Results at $t = 100$.

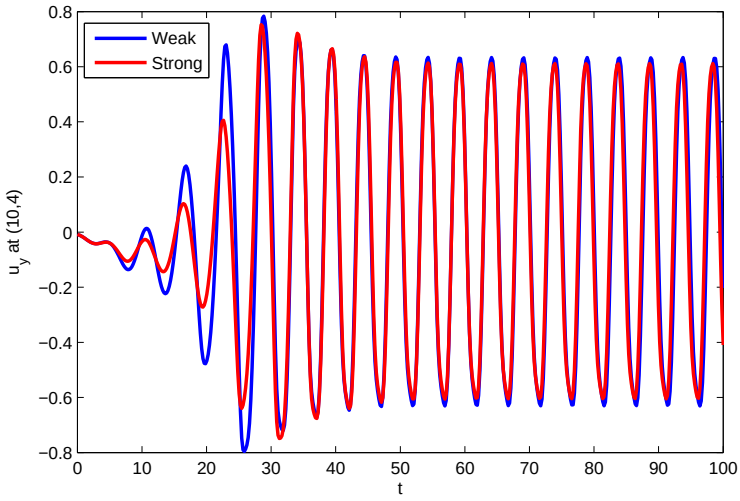


Figure 3.9: Vertical velocity evolution at $(10, 4)$. Comparison between weak and strong enforcement of boundary conditions.

3.6.4 Weak imposition of boundary conditions in the transport equation

In this subsection we study the pure transport equation in which only boundary conditions on the inflow are needed. We solve the problem described in the convection-diffusion subsection, but we only impose boundary conditions on the inflow. Linear convergence is obtained with both the strong imposition of boundary conditions described in Chapter 2 and the strategy described in this chapter, although quadratic convergence is obtained for the strong imposition of boundary conditions in boundary matching grids. For the method described in this chapter, the problem does not seem to be in the outflow, but in the inflow, where the method does not impose boundary conditions strongly enough.

In Fig. 3.10 we can observe the error for different meshes in the pure transport equation. It can be seen that the error diminishes linearly with the mesh size but also, and most importantly, that the computed solution is displaced, that is, the error does not oscillate around 0, but around 0.08 (h = 1/50), 0.04 (h = 1/100) and 0.02 (h = 1/200). This suggests that the boundary conditions are not strongly enough imposed.

Fig. 3.11 shows the convergence rates for the solution of the pure transport equation for different methods. We can see that convergence is linear for both weak and strong boundary conditions if a stabilized formulation is used. If no stabilization is used, *and the mesh nodes are aligned with the advection direction*, the convergence is closer to quadratic. This suggests that the incorrect weighting of the boundary conditions is due to the stabilization terms.

Since the stabilization terms add numerical viscosity in the direction of the streamlines, we can try considering a different viscosity in the term which weights the boundary conditions in the Poisson problem:

$$k* = k + \alpha \tau a^2, \tag{3.86}$$

where α is a dimensionless constant. In this way we are able to recover a close to quadratic convergence. We use this viscosity for the computation of the terms imposing the boundary

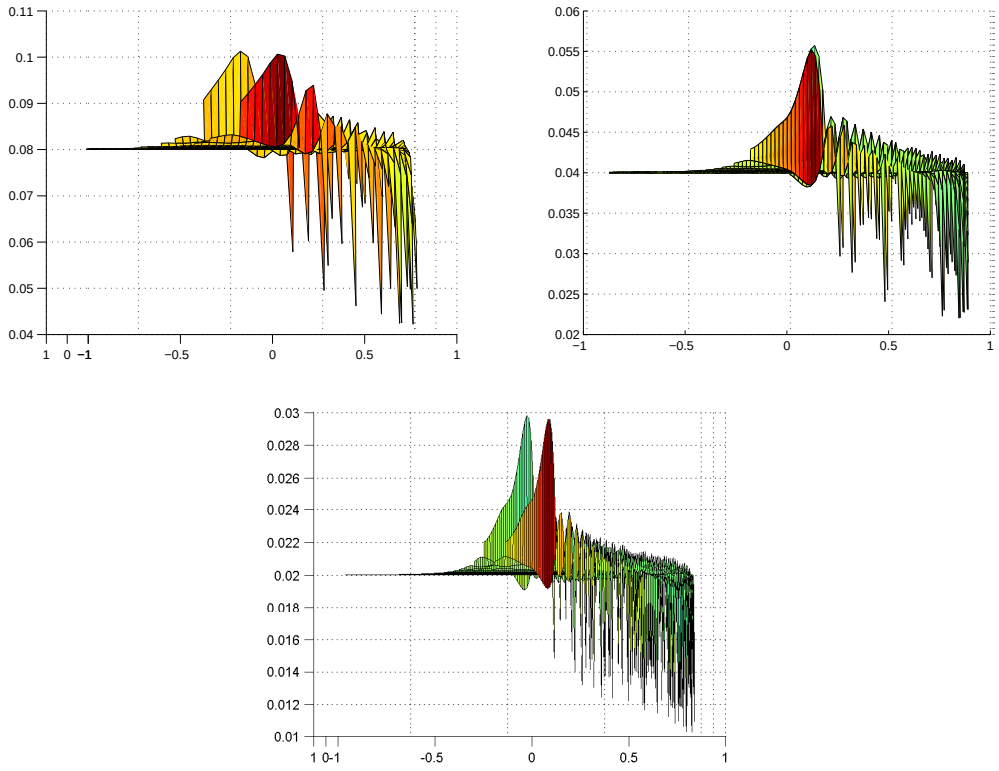


Figure 3.10: Error comparison (u_h - analytical) for the pure transport equation for h = 1/50, 1/100, 1/200. We can observe that the error is diminishing linearly with h. Moreover it is displaced 0.08, 0.04 and 0.02 from the 0 position. A stronger imposition of boundary conditions would improve the solution.

conditions:

$$-\boldsymbol{G}_{u\sigma}\boldsymbol{K}_{\sigma\sigma}^{-1}\boldsymbol{G}_{\sigma u}\cdot\boldsymbol{U} \text{ and } -\boldsymbol{G}_{u\sigma}\boldsymbol{K}_{\sigma\sigma}^{-1}\boldsymbol{g}_{\sigma\bar{u}} \tag{3.87}$$

which we substitute by:

$$-\frac{k*}{k}\boldsymbol{G}_{u\sigma}\boldsymbol{K}_{\sigma\sigma}^{-1}\boldsymbol{G}_{\sigma u}\cdot\boldsymbol{U} \text{ and } -\frac{k*}{k}\boldsymbol{G}_{u\sigma}\boldsymbol{K}_{\sigma\sigma}^{-1}\boldsymbol{g}_{\sigma\bar{u}} \tag{3.88}$$

The optimal value for α happens to be around $\alpha = 200$, which does not have any physical meaning. Further work needs to be done in order to find a proper definition of the weighting terms for the boundary conditions in the pure transport equation.

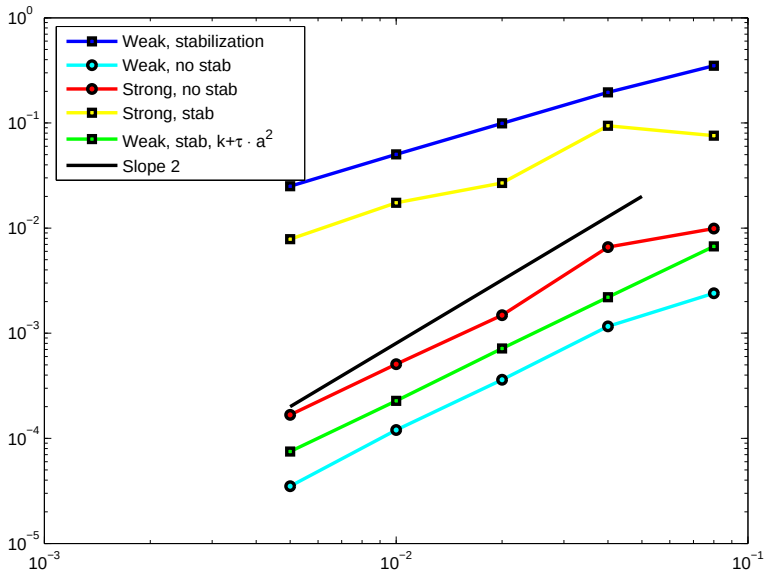


Figure 3.11: Convergence plots for the transport equation, comparison between several methods

3.7 Conclusions

In this chapter we have proposed a way to weakly prescribe Dirichlet boundary conditions in embedded grids. The key feature of the proposed method is that we can ensure stability without the need of a large penalty parameter and that it is symmetric for symmetric problems. In the Poisson problem this is achieved by introducing an additional element-discontinuous stress variable. Additional terms are required in order to guarantee stability in the convection-diffusion equation, in which we weight the boundary conditions with a particular norm of the convection velocity, and the Stokes problem, where we test the imposition of boundary conditions against the pressure test functions. The proposed strategy is then easily extended to the transient Navier-Stokes equations.

The method turns out to be accurate (second order for linear elements) and robust for all the problems tested except for the pure transport equation, in which we are not able to recover quadratic convergence. Further work needs to be developed to find a proper definition of the weighting terms for the imposition of boundary conditions in the pure transport equation. From the implementation point of view, the method satisfies the main design condition of using only the degrees of freedom of the mesh of Ω_h. Moreover, the final resulting method is very easy to implement, since it only requires some additional boundary integrals to be added to the original variational form.

When compared to the method described in Chapter 2 we can conclude that both methods perform similarly well, and are both equally suitable for flow problems. However, recent research suggests that the weak imposition of boundary conditions (both in matching and non-matching grids) could be more suitable for highly turbulent flows (see [15]). On the other hand, strong boundary conditions seem more suitable for problems which require a sharp tracking of the domain movement, such as the free surface problems to be described in the following chapters.

Chapter 4

The Fixed-Mesh ALE approach applied to flows in moving domains

In this chapter we propose a method to approximate flow problems in moving domains using always a given grid for the spatial discretization, and therefore the formulation to be presented falls within the category of fixed-grid methods. Our main concern is to properly account for the advection of information as the domain boundary evolves. To achieve this, we use an arbitrary Lagrangian-Eulerian framework, the distinctive feature being that at each time step results are projected onto a fixed, background mesh, that is where the problem is actually solved.

4.1 Introduction

In many coupled problems of practical interest the domain of at least one of the problems evolves in time. The Arbitrary Eulerian Lagrangian (ALE) approach is a tool very often employed to cope with this domain motion. In this work we aim at describing a particular version of the ALE formulation that can be used in different coupled problems. In this chapter we will particularize it to flow problems.

In the classical ALE approach to solve problems in computational fluid dynamics, the mesh in which the computational domain is discretized is deformed (see for example [45, 79, 74]). This is done according to a prescribed motion of part of its boundary, which is transmitted to the interior nodes in a way as smooth as possible so as to avoid mesh distortion. In this work we present an ALE-type strategy with a different motivation. Instead of assuming that the computational domain is defined by the mesh boundary, we assume that there is a function that defines the boundary of the domain where the flow takes place. We will refer to it as the *boundary function*. It may be given, for example, by the shape of a body that moves within the fluid, or it may need to be computed, as in the case of level set functions. It may be also defined discretely, by a set of points. When this boundary function moves, the flow domain changes, and that must be taken into account at the moment of writing the conservation equations that govern the flow, which need to be cast in the ALE format. However, our purpose here is to explain how to use always a background *fixed* mesh. That requires a virtual motion of the mesh nodes followed by a projection of the new node positions onto the fixed mesh.

The basic numerical formulation we will use consists of a stabilized finite element method

to solve the ALE flow equations and finite difference time integration schemes. However, other discretization techniques could be applied, since the idea we want to expose is independent of the numerical method being used. This idea consists in projecting the results of the ALE deformed mesh onto a fixed background mesh *at each time step*, prior to solving the flow equations. It will be shown that at the end all the calculations can be performed on the fixed mesh, and in fact the ALE deformed mesh does not need to be explicitly built.

We want to stress that this idea is independent on the way to impose boundary conditions on the moving boundary. The way to impose this prescription is often used to classify a particular fixed-mesh method. Since the physical boundary is contained in the domain actually discretized, these methods are often called *immersed boundary methods*. Moreover, since the fixed grid used is often Cartesian, these formulations can be found under the keywords *Cartesian grid methods* (see for example the reviews [127, 103, 100]). These methods are developed for constant-in-time domains, and then extended in a more or less ad-hoc way to time dependent domains. In spite of the fact that we want to distinguish between the way to deal with moving domains from the way of approximately imposing the boundary conditions on the moving boundary, we will briefly describe the particular approach we use, which corresponds to the method described in Chapter 2.

The chapter is organized as follows. A general overview of the FM-ALE method is presented in Section 4.2, starting with the discretization of the classical ALE formulation and then describing the algorithmic steps of the FM-ALE alternative. These steps are further elaborated in Section 4.3. Even though they are not intrinsic to the main idea of the method, there are three numerical ingredients that are essential for the success of the formulation. These are the definition and updating of the moving boundary, the approximate imposition of boundary conditions and the projection of data between two different finite element meshes. These "side ingredients" are here particularized to the FM-ALE method. They are described in Section 4.4. A simple numerical example, but containing all the features of the formulation, is presented in Section 4.5. For other applications of the FM-ALE method see [73], where the simulation of lost foam casting is carried out, or [41], were classical free surface problems are solved with the FM-ALE method. Some conclusions close the chapter in Section 4.6.

4.2 The Fixed-Mesh ALE method

In this section we describe the essential idea of the FM-ALE method. However, we start with the classical ALE formulation of the incompressible Navier-Stokes equations and their numerical approximation.

4.2.1 The classical ALE method and its finite element approximation

Problem statement

Let us consider a region $\Omega^0 \subset \mathbb{R}^d$ ($d = 2, 3$) where a flow will take place during a time interval $[0, T]$. However, we consider the case in which the fluid at time t occupies only a subdomain $\Omega(t) \subset \Omega^0$ (note in particular that $\Omega(0) \subset \Omega^0$). Suppose also that the boundary of $\Omega(t)$ is defined by part of $\partial\Omega^0$ and a moving boundary that we call $\Gamma_{\text{free}}(t) = \partial\Omega(t) \setminus \partial\Omega^0 \cap \partial\Omega(t)$.

This moving part of $\partial\Omega(t)$ may correspond to the boundary of a moving solid immersed in the fluid or can be determined by a level set function.

In order to cope with the time-dependency of $\Omega(t)$, we use the ALE approach, with the particular feature of considering a variable definition of the domain velocity. Let $\boldsymbol{\chi}_t$ be a family of invertible mappings, which for all $t \in [0,T]$ map a point $\boldsymbol{X} \in \Omega(0)$ to a point $\boldsymbol{x} = \boldsymbol{\chi}_t(\boldsymbol{X}) \in \Omega(t)$, with $\boldsymbol{\chi}_0 = \boldsymbol{I}$, the identity. If $\boldsymbol{\chi}_t$ is given by the motion of the particles, the resulting formulation would be Lagrangian, whereas if $\boldsymbol{\chi}_t = \boldsymbol{I}$ for all t, $\Omega(t) = \Omega(0)$ and the formulation would be Eulerian.

Let now $t' \in [0,T]$, with $t' \leq t$, and consider the mapping

$$\begin{aligned} \boldsymbol{\chi}_{t,t'} : \Omega(t') &\longrightarrow \Omega(t) \\ \boldsymbol{x}' &\mapsto \boldsymbol{x} = \boldsymbol{\chi}_t \circ \boldsymbol{\chi}_{t'}^{-1}(\boldsymbol{x}'). \end{aligned}$$

Given a function $f : \Omega(t) \times (0,T) \longrightarrow \mathbb{R}$ we define

$$\left.\frac{\partial f}{\partial t}\right|_{\boldsymbol{x}'}(\boldsymbol{x},t) := \frac{\partial (f \circ \boldsymbol{\chi}_{t,t'})}{\partial t}(\boldsymbol{x}',t), \quad \boldsymbol{x} \in \Omega(t),\ \boldsymbol{x}' \in \Omega(t').$$

In particular, the domain velocity taking as a reference the coordinates of $\Omega(t')$ is given by

$$\boldsymbol{u}_{\text{dom}} := \left.\frac{\partial \boldsymbol{x}}{\partial t}\right|_{\boldsymbol{x}'}(\boldsymbol{x},t). \tag{4.1}$$

The incompressible Navier-Stokes formulated in $\Omega(t)$, accounting also for the motion of this domain, can be written as follows: find a velocity $\boldsymbol{u} : \Omega(t) \times (0,T) \longrightarrow \mathbb{R}^d$ and a pressure $p : \Omega(t) \times (0,T) \longrightarrow \mathbb{R}$ such that

$$\rho \left[\left.\frac{\partial \boldsymbol{u}}{\partial t}\right|_{\boldsymbol{x}'}(\boldsymbol{x},t) + (\boldsymbol{u} - \boldsymbol{u}_{\text{dom}}) \cdot \nabla \boldsymbol{u} \right] - \nabla \cdot (2\mu \nabla^S \boldsymbol{u}) + \nabla p = \rho \boldsymbol{f}, \tag{4.2}$$

$$\nabla \cdot \boldsymbol{u} = 0, \tag{4.3}$$

where $\nabla^S \boldsymbol{u}$ is the symmetrical part of the velocity gradient, ρ is the fluid density, μ is the viscosity and $\boldsymbol{f}$ is the vector of body forces.

Initial and boundary conditions have to be appended to problem (4.2)-(4.3). The boundary conditions on $\Gamma_{\text{free}}(t)$ can be of two different types: a) p (or the normal stress) given, $\boldsymbol{u}$ unknown on Γ_{free}; b) $\boldsymbol{u}$ given, p (or the normal stress) unknown on Γ_{free}. On the rest of the boundary of $\Omega(t)$ the usual boundary conditions can be considered. In general, we consider these boundary conditions of the form

$$\begin{aligned} \boldsymbol{u} &= \bar{\boldsymbol{u}} \quad \text{on } \Gamma_D, \\ \boldsymbol{n} \cdot \boldsymbol{\sigma} &= \bar{\boldsymbol{t}} \quad \text{on } \Gamma_N, \end{aligned}$$

where $\boldsymbol{n}$ is the external normal to the boundary, $\boldsymbol{\sigma} = -p\boldsymbol{I} + 2\mu\nabla^S \boldsymbol{u}$ is the Cauchy stress tensor and $\bar{\boldsymbol{u}}$ and $\bar{\boldsymbol{t}}$ are the given boundary data. The components of the boundary Γ_D and Γ_N are disjoint and such that $\Gamma_D \cup \Gamma_N = \partial\Omega$, and therefore time-dependent.

The time-discrete problem

Let us start introducing some notation. Consider a uniform partition of $[0,T]$ into N time intervals of length δt. Let us denote by f^n the approximation of a time dependent function f at time level $t^n = n\delta t$. We will also denote

$$\begin{aligned} \delta f^{n+1} &= f^{n+1} - f^n, \\ \delta_t f^{n+1} &= \frac{f^{n+1} - f^n}{\delta t}, \\ f^{n+\theta} &= \theta f^{n+1} + (1-\theta) f^n, \quad \theta \in [1/2, 1]. \end{aligned}$$

Even though other options are possible, we will use the simple trapezoidal rule to discretize problem (4.2)-(4.3) in time. Suppose we are given a computational domain at time t^n, with spatial coordinates labeled $\boldsymbol{x}^n$, and $\boldsymbol{u}^n$ and p^n are known in this domain. The velocity $\boldsymbol{u}^{n+1}$ and the pressure p^{n+1} can then be found as the solution to the problem

$$\rho \left[\delta_t \boldsymbol{u}^{n+1}\big|_{\boldsymbol{x}^n} + (\boldsymbol{u}^{n+\theta} - \boldsymbol{u}^{n+\theta}_{\rm dom}) \cdot \nabla \boldsymbol{u}^{n+\theta} \right] - \nabla \cdot (2\mu \nabla^S \boldsymbol{u}^{n+\theta}) + \nabla p^{n+1} = \rho \boldsymbol{f}^{n+1}, \tag{4.4}$$

$$\nabla \cdot \boldsymbol{u}^{n+\theta} = 0, \tag{4.5}$$

where now $\delta_t \boldsymbol{u}^{n+1}|_{\boldsymbol{x}^n} = (\boldsymbol{u}^{n+1}(\boldsymbol{x}) - \boldsymbol{u}^n(\boldsymbol{x}^n))/\delta t$, being $\boldsymbol{x} = \boldsymbol{\chi}_{t^{n+\theta},t^n}(\boldsymbol{x}^n)$ the spatial coordinates in $\Omega(t^{n+\theta})$. The domain velocity given by (4.1), with $\boldsymbol{x}' = \boldsymbol{x}^n$, is approximated as

$$\boldsymbol{u}^{n+\theta}_{\rm dom} = \frac{1}{\theta \delta t} \left(\boldsymbol{\chi}_{t^{n+\theta},t^n}(\boldsymbol{x}^n) - \boldsymbol{x}^n \right). \tag{4.6}$$

Note that the order of accuracy of this approximation is consistent with the order of accuracy of (4.4)-(4.5), that is to say, it is 2 for $\theta = 1/2$ and 1 otherwise. We are interested only in the cases $\theta = 1/2$ and $\theta = 1$ (implicit schemes are required).

Remark 1 The trapezoidal rule considered for the time integration, with a single mesh, satisfies the so called *geometric conservation law* (GCL) condition (see, e.g. [20, 52, 90]). However, there are second order accurate schemes based on multi-step time discretizations that do not satisfy it. The price to be paid is that these schemes are usually only conditionally stable, although stability conditions are often very mild and not encountered in practice (see for example the analyses in [7, 20, 52, 53, 109]). We will use one of such schemes in Section 4.5. $\triangle$

The fully discrete problem

The next step is to consider the spatial discretization of problem (4.4)-(4.5). As for the time discretization, different options are possible. Here we simply describe the stabilized finite element formulation employed in our numerical simulations.

Let $\{\Omega^e\}^{n+1}$ be a finite element partition of the domain $\Omega(t^{n+1})$, with index e ranging from 1 to the number of elements $n_{\rm el}$ (which may be different at different time steps). We denote with a subscript h the finite element approximation to the unknown functions, and by $\boldsymbol{v}_h$ and q_h the velocity and pressure test functions associated to $\{\Omega^e\}^{n+1}$, respectively.

An important point is that we are interested in using equal interpolation for the velocity and the pressure. Therefore, the corresponding finite element spaces are assumed to be built up using the standard continuous interpolation functions.

In order to overcome the numerical problems of the standard Galerkin method, a stabilized finite element formulation is applied. This formulation is presented in [31]. It is based on the subgrid scale concept introduced in [76], although when linear elements are used it reduces to the Galerkin/least-squares method described for example in [56]. We apply this stabilized formulation together with the finite difference approximation in time (4.4)-(4.5).

The bottom line of the method is to test the continuous equations by the standard Galerkin test functions plus perturbations that depend on the operator representing the differential equation being solved. In our case, this operator corresponds to the linearized form of the time discrete Navier-Stokes equations (4.4)-(4.5). In this case, the method consists of finding $\boldsymbol{u}_h^{n+1}$ and p_h^{n+1} such that

$$
\begin{aligned}
& m_1^{n+\theta}\left(\delta_t \boldsymbol{u}_h^{n+1}\big|_{\boldsymbol{x}^n}, \boldsymbol{v}_h\right) + a^{n+\theta}(\boldsymbol{u}_h, \boldsymbol{v}_h) \\
& \qquad + c^{n+\theta}(\boldsymbol{u}_h - \boldsymbol{u}_{\mathrm{dom}}; \boldsymbol{u}_h, \boldsymbol{v}_h) + b_1^{n+\theta}(p_h, \boldsymbol{v}_h) = l_1^{n+\theta}(\boldsymbol{v}_h), \qquad (4.7) \\
& m_2^{n+\theta}\left(q_h, \delta_t \boldsymbol{u}_h^{n+1}\big|_{\boldsymbol{x}^n}\right) + b_2^{n+\theta}(q_h, \boldsymbol{u}_h) + s^{n+\theta}(q_h, p_h) = l_2^{n+\theta}(q_h), \qquad (4.8)
\end{aligned}
$$

for all test functions $\boldsymbol{v}_h$ and q_h, the former vanishing on the Dirichlet part of the boundary Γ_D. The different forms appearing in these equations are given by

$$
\begin{aligned}
& m_1(\delta_t \boldsymbol{u}_h, \boldsymbol{v}_h) = \int_\Omega \boldsymbol{v}_h \cdot \rho\, \delta_t \boldsymbol{u}_h + \sum_{e=1}^{n_{\mathrm{el}}} \int_{\Omega^e} \boldsymbol{\zeta}_{u1} \cdot \rho\, \delta_t \boldsymbol{u}_h, \\
& a(\boldsymbol{u}_h, \boldsymbol{v}_h) = \int_\Omega 2\nabla^S \boldsymbol{v}_h : \mu \nabla^S \boldsymbol{u}_h + \sum_{e=1}^{n_{\mathrm{el}}} \int_{\Omega^e} \boldsymbol{\zeta}_{u1} \cdot \left(-2\nabla \cdot (\mu \nabla^S \boldsymbol{u}_h)\right) + \sum_{e=1}^{n_{\mathrm{el}}} \int_{\Omega^e} \zeta_{u2} \nabla \cdot \boldsymbol{u}_h, \\
& c(\boldsymbol{a}; \boldsymbol{u}_h, \boldsymbol{v}_h) = \int_\Omega \boldsymbol{v}_h \cdot (\rho\, \boldsymbol{a} \cdot \nabla \boldsymbol{u}_h) + \sum_{e=1}^{n_{\mathrm{el}}} \int_{\Omega^e} \boldsymbol{\zeta}_{u1} \cdot (\rho\, \boldsymbol{a} \cdot \nabla \boldsymbol{u}_h), \\
& b_1(p_h, \boldsymbol{v}_h) = -\int_\Omega p_h \nabla \cdot \boldsymbol{v}_h + \sum_{e=1}^{n_{\mathrm{el}}} \int_{\Omega^e} \boldsymbol{\zeta}_{u1} \cdot \nabla p_h, \\
& m_2(q_h, \delta_t \boldsymbol{u}_h) = \sum_{e=1}^{n_{\mathrm{el}}} \int_{\Omega^e} \boldsymbol{\zeta}_p \cdot \rho\, \delta_t \boldsymbol{u}_h, \\
& b_2(q_h, \boldsymbol{u}_h) = \int_\Omega q_h \nabla \cdot \boldsymbol{u}_h + \sum_{e=1}^{n_{\mathrm{el}}} \int_{\Omega^e} \boldsymbol{\zeta}_p \cdot \left(\rho\, \boldsymbol{a} \cdot \nabla \boldsymbol{u}_h - 2\nabla \cdot (\mu \nabla^S \boldsymbol{u}_h)\right), \\
& s(q_h, p_h) = \sum_{e=1}^{n_{\mathrm{el}}} \int_{\Omega^e} \boldsymbol{\zeta}_p \cdot \nabla p_h, \\
& l_1(\boldsymbol{v}_h) = \int_\Omega \boldsymbol{v}_h \cdot \boldsymbol{f} + \sum_{e=1}^{n_{\mathrm{el}}} \int_{\Omega^e} \boldsymbol{\zeta}_{u1} \cdot \boldsymbol{f} + \int_{\Gamma_{\mathrm{N}}} \boldsymbol{v}_h \cdot \bar{\boldsymbol{t}}, \\
& l_2(q_h) = \sum_{e=1}^{n_{\mathrm{el}}} \int_{\Omega^e} \boldsymbol{\zeta}_p \cdot \boldsymbol{f},
\end{aligned}
$$

where the functions $\boldsymbol{\zeta}_{u1}$, ζ_{u2} and $\boldsymbol{\zeta}_p$ are computed within each element as

$$\boldsymbol{\zeta}_{u1} = \tau_u \left[\rho \left(\boldsymbol{u}_h - \boldsymbol{u}_{\rm dom}\right) \cdot \nabla \boldsymbol{v}_h + 2\nabla \cdot \left(\mu \nabla^S \boldsymbol{v}_h\right)\right], \tag{4.9}$$

$$\zeta_{u2} = \tau_p \nabla \cdot \boldsymbol{v}_h, \tag{4.10}$$

$$\boldsymbol{\zeta}_p = \tau_u \nabla q_h, \tag{4.11}$$

and the parameters τ_u and τ_p are also computed element-wise as (see [32])

$$\tau_u = \left[\frac{4\mu}{h^2} + \frac{2\rho|\boldsymbol{u}_h - \boldsymbol{u}_{\rm dom}|}{h}\right]^{-1}, \qquad \tau_p = 4\mu + 2\rho|\boldsymbol{u}_h - \boldsymbol{u}_{\rm dom}|h,$$

where h is the element size for linear elements and half of it for quadratics.

Remark 2

- The superscript $n+\theta$ in all the terms in (4.7)-(4.8) indicates that all the forms are evaluated with the unknowns at $n+\theta$, except for the term coming from the temporal derivative, whose superscript is explicitly indicated. Likewise, the integrals are evaluated at $\Omega(t^{n+\theta})$.

- The dependency on the advection velocity $\boldsymbol{a} = \boldsymbol{u}_h - \boldsymbol{u}_{\rm dom}$ has been only indicated in the from coming directly from the convective term of the equations, namely, $c(\boldsymbol{a}; \boldsymbol{u}_h, \boldsymbol{v}_h)$. However, it has to be noted that all the forms listed above depend on the stabilization parameters, and therefore depend on $\boldsymbol{a}$ as well. Moreover, the dependency of $b_2(q_h, \boldsymbol{u}_h)$ on $\boldsymbol{a}$ is even more explicit. However, in order to keep the notation more concise only the above mentioned dependency of $c(\boldsymbol{a}; \boldsymbol{u}_h, \boldsymbol{v}_h)$ has been left.

- As usual, the mesh of $\Omega(t^{n+1})$ is assumed to be obtained from the mesh of $\Omega(t^n)$ by moving the nodes of the latter with the domain velocity $\boldsymbol{u}_{\rm dom}$ (often referred to as mesh velocity). This greatly simplifies the implementation of the ALE method, since in this case the nodal values of $\boldsymbol{u}^{n+1}(\boldsymbol{x})$ and those of $\boldsymbol{u}^n(\boldsymbol{x}^n)$ correspond to the same nodes (at time steps $n+1$ and n, respectively).

- If $\theta = 1/2$, the unknowns of the problem can be taken as $\boldsymbol{u}^{n+1/2}$ and $p^{n+1/2}$, since $\delta_t \boldsymbol{u}_h^{n+1}\big|_{\boldsymbol{x}^n} = 2\delta t^{-1}(\boldsymbol{u}^{n+1/2}(\boldsymbol{x}) - \boldsymbol{u}^n(\boldsymbol{x}^n))$. All the calculations to be performed are the same as for $\theta = 1$, with the only modification that once $\boldsymbol{u}^{n+1/2}$ is computed $\boldsymbol{u}^{n+1}$ has to be updated to go to the next time step. This analogy includes the updating of the computational domain. When $\theta = 1/2$ we need to update this domain from $n - 1/2$ to $n + 1/2$ to compute $\boldsymbol{u}^{n+1/2}$ and $p^{n+1/2}$, whereas when $\theta = 1$ we need to update it from n to $n+1$ to compute $\boldsymbol{u}^{n+1}$ and p^{n+1}. For conciseness, the latter situation is considered in the following.

- From (4.9)-(4.11) it is observed that these terms are precisely the adjoints of the (linearized) operators of the differential equations to be solved applied to the test functions (observe the sign of the viscous term in (4.9)). This method corresponds to the algebraic version of the subgrid scale approach ([76]) and circumvents the stability problems of the Galerkin method. In particular, in this case it is possible to use equal velocity pressure interpolations, that is, we are not tied to the satisfaction of the inf-sup stability condition. For more details about this formulation, see for example [76, 31].

△

4.2.2 The fixed-mesh ALE approach: algorithmic steps

The purpose of this subsection is to give an overview of the FM-ALE method and to describe the main idea, leaving for the next section a more detailed description of the different steps involved.

Suppose Ω^0 is meshed with a finite element mesh M^0 and that at time level t^n the domain $\Omega(t^n)$ is meshed with a finite element mesh M^n (as we will see, close to M^0). Let $\boldsymbol{u}^n$ be the velocity already computed on $\Omega(t^n)$. The purpose is to obtain the fluid region $\Omega(t^{n+1})$ and the velocity field $\boldsymbol{u}^{n+1}$. The former may move according to a prescribed kinematics, for example due to the motion of a solid, or can be an unknown of the problem. If the classical ALE method is used, M^n would deform to another mesh defined at t^{n+1}. The key idea is not to use this mesh to compute $\boldsymbol{u}^{n+1}$ and p^{n+1}, but to re-mesh in such a way that the new mesh is, essentially, M^0 once again.

The steps of the algorithm to achieve the goal described are the following:

1. Define $\Gamma_{\text{free}}^{n+1}$ by updating the function that defines it.
2. Deform *virtually* the mesh M^n to M_{virt}^{n+1} using the classical ALE concepts and compute the mesh velocity $\boldsymbol{u}_m^{n+1}$.
3. Write down the ALE Navier-Stokes equations on M_{virt}^{n+1}.
4. *Split the elements* of M^0 cut by $\Gamma_{\text{free}}^{n+1}$ to define a mesh on $\Omega(t^{n+1})$, M^{n+1}.
5. *Project* the ALE Navier-Stokes equations from M_{virt}^{n+1} to M^{n+1}.
6. Solve the equations on M^{n+1} to compute $\boldsymbol{u}^{n+1}$ and p^{n+1}.

In Section 4.3 we describe all these steps in detail. A global idea of the meshes involved in the process is represented in Fig. 4.1. Note in particular that at each time steps two sets of nodes have to be appropriately dealt with, namely, the so called newly created nodes and the boundary nodes. Contrary to other fixed grid methods, some of which are described in the next subsection, newly created nodes are treated in a completely natural way using the FM-ALE approach: the value of the velocity there is directly given by the projection step from M_{virt}^{n+1} to M^{n+1}. Boundary nodes require either additional unknowns with respect to those of mesh M^0 or an appropriate imposition of boundary conditions. This issue is treated in Section 4.4.

4.2.3 Other fixed grid methods

Other possibilities to use a single grid in the whole simulation can be found in the literature, each one having advantages and drawbacks. As the method presented in this chapter, they were designed as an alternative to body fitted meshes and are sometimes referred to as *Embedded Mesh Methods*. They can be divided into two main groups [35], corresponding in fact to two ways of prescribing the boundary conditions on Γ_{free}:

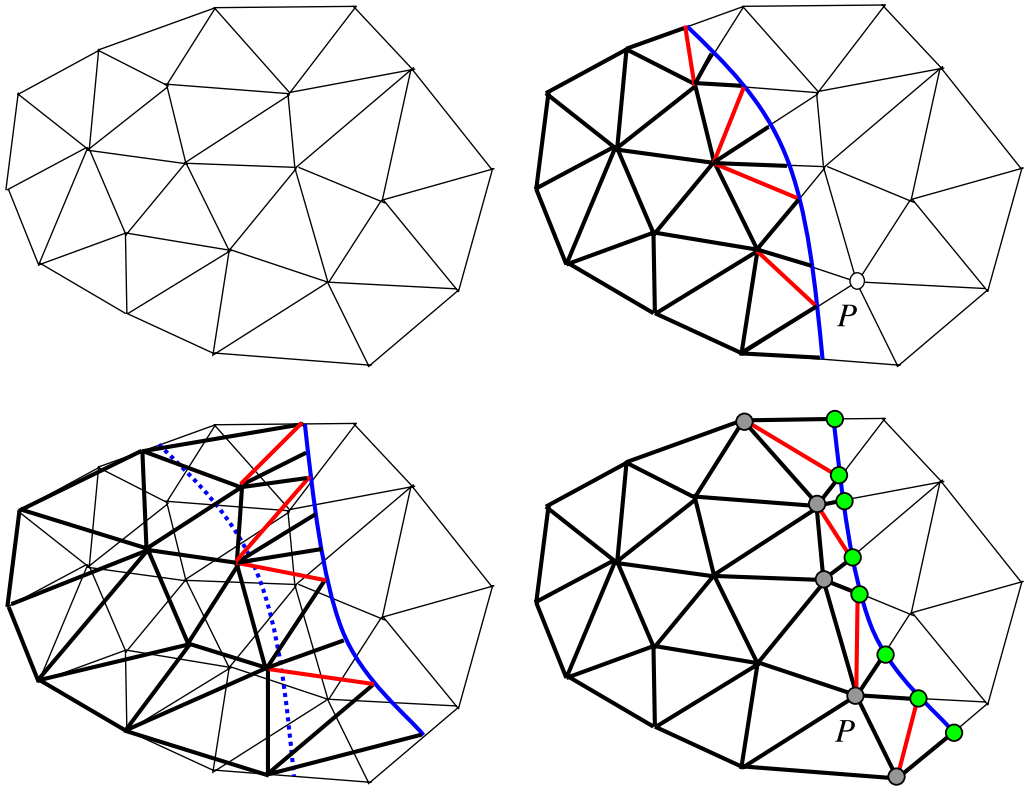


Figure 4.1: Two dimensional FM-ALE schematic. Top-left: original finite element mesh M^0 of Ω^0. Top-right: finite element mesh M^n of $\Omega(t^n)$, with the elements represented by a thick line and the elements of M^0 represented by thin line. The blue line represents Γ^n_{free} and the red edges indicate the splitting of M^0 to obtain M^n. Bottom-left: updating of M^n to M^{n+1}_{virt} using the classical ALE strategy. The position of $\Gamma^{n+1}_{\text{free}}$ is again shown using a solid blue line and the previous position Γ^n_{free} using a dotted blue line. Bottom-right: Mesh M^{n+1} of $\Omega(t^{n+1})$, represented by a thick line. The edges that split elements of M^0 are again indicated in red. Boundary nodes, where approximate boundary conditions need to be imposed, are drawn in green, whereas newly created nodes are drawn in gray.

- *Force term*. The interaction of the fluid and the solid is taken into account through a force term, which appears either in the strong or in the weak form of the flow equations. Therefore, the boundary conditions on Γ_{free} are neither imposed as Dirichlet nor as Neumann boundary conditions. Among this type of methods, let us cite for example the Immersed Boundary method as a variant of the Penalty method, where punctual forces are added to the momentum equation, and the Fictitious Domain method, where the solid boundary conditions are imposed through a Lagrange multiplier.

- *Approximate boundary conditions*. Instead of adding a force term, these methods impose the boundary conditions 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.

The *Immersed Boundary Method* in its original form [113] consists in adding punctual penalty forces in the domain boundary so that the boundary conditions are fulfilled. The forces are computed from a fluid-structure (elastic) interaction problem at the interface. The method is first order accurate even if second order approximation schemes are used, although *formal second order accuracy* has been reported in [87]. The more recent *Immersed Interface Method* achieves higher order accuracy by avoiding the use of the Dirac delta distribution to define the forcing terms (see [91, 92, 137]).

The *Penalty method* is similar to the previous one in the sense that a force term is added to the momentum equations. The difference raises in the fact that the penalty parameter is not computed from a fluid-structure interaction as in the original immersed boundary method, but it is simply required to be large enough to enforce the boundary conditions approximately. The force terms can be of two types, depending on whether they are imposed as boundary or as volume forces [133] .

Another approach is the use of Lagrange multipliers to enforce the boundary conditions. However, the finite element subspaces for the bulk and Lagrange multiplier fields must satisfy the classical inf-sup condition, which usually leads to the need for stabilization (see [70, 14, 82]). Moreover, additional degrees of freedom must be added to the problem. The use of Lagrange multipliers is the basis of the *Fictitious Domain Method* [62, 63].

Recently, *hybrid Cartesian/immersed boundary methods* have been developed for Cartesian grids, which use the grid nodes closest to the boundary to enforce boundary conditions [60, 139, 104]. The method is second order accurate.

Most of these methods have been well tested in the literature for both steady and moving interfaces. Generally, the last case is treated by applying directly the former at each time step. However, very few authors have described the full formulation for moving interfaces, sometimes simply by ignoring the problem. The fact that the boundary moves and the subsequent advection of unknowns is often not taken into account.

To explain an obvious consequence of the boundary motion, let us discuss the treatment of the newly created nodes. To explain the problem, let us consider point P in Fig. 4.1. Suppose that the boundary Γ_{free} corresponds in this case to the rigid boundary of a moving object. Physically, it is clear that the solution in the fluid cannot depend on what happens inside the solid. Mathematically, this means that the values of the unknowns at the fluid nodes are uncoupled from those at the solid nodes. Therefore, the velocity and the pressure at the solid nodes (apart from those participating to the enforcing of the boundary conditions) can be

whatever at a certain time step n, in particular their value at node P (see Fig. 4.1, top-right). Now we move on to the next time step $n+1$ as the solid moves. Some solid nodes can therefore become fluid nodes, such as node P (see Fig. 4.1, bottom-right). The velocity at this node at time step n is in fact needed in the temporal term of the momentum equations and cannot be *whatever*. In the case of fractional step techniques, the situation can even be worse as the previous time step pressure could also be needed at these nodes.

A special treatment is needed for the newly created fluid nodes. In many publications, the previous time step values are computed using ad hoc arguments, that sometimes lead to good approximations from the practical point of view when small time steps are used. As an example, in [100] the authors extrapolate the velocity and pressure from the nearest fluid nodes at the previous time step. In [27], the Navier-Stokes equations are correctly expressed in an ALE framework, but the velocity is taken as the solid velocity. It is worth to note that if the solid is deformable and has been solved together with the fluid in a coupled way (as in the original immersed boundary method [113] or in the fluid-solid approach in [141]), this velocity is physically meaningful. This is not the case, however, is the case of rigid bodies or bodies with rigid boundaries. A possibility to deal with this situation is to write the Navier-Stokes equations in a non-inertial frame of reference attached to the body, as in [72] in the context of Chimera meshes or in [86], where an immersed boundary method is used.

We explain in the following what we believe is a consistent way of treating moving interfaces based on a fixed-mesh ALE approach.

4.3 Developing the Fixed-Mesh ALE method

In this section we describe the steps enumerated previously, concentrating on those specific of the FM-ALE method and leaving for Section 4.4 those that can be considered side numerical ingredients.

4.3.1 Step 1. Boundary function update

This step is completely problem dependent. The motion of $\Gamma_{\text{free}}(t)$ may be determined by different ways. In a typical fluid-structure interaction problem, $\Gamma_{\text{free}}(t)$ will be part of the solid boundary, and therefore its kinematics will be determined by the dynamics of the solid under the action exerted by the fluid. As a particular case, the motion of the solid boundary may be directly prescribed. This is the simplest situation and the one corresponding to the validating numerical example presented in Section 4.5.

In a wide variety of applications, $\Gamma_{\text{free}}(t)$ may be represented by a level set function. The peculiarities of the levelset function update in the context of the FM-ALE approach are described in Section 4.4.

4.3.2 Step 2. Mesh velocity

Updating the boundary function defines the deformation of the domain from $\Omega(t^n)$ to $\Omega(t^{n+1})$ (recall that we are considering the case $\theta = 1$, see Remark 2). Consequently, the mesh M^n

used at time step n has to be deformed to adapt to the domain $\Omega(t^{n+1})$. This mesh deformation has to be defined by means of a mesh velocity.

The mesh velocity on the boundary points can be computed from their position $\boldsymbol{x}_b^{n+1}$ and $\boldsymbol{x}_b^n$, where subscript b refers to points on Γ_{free}. Using approximation (4.6), this mesh velocity would be $\boldsymbol{u}_{\text{dom},b}^{n+1} = (\boldsymbol{x}_b^{n+1} - \boldsymbol{x}_b^n)/\delta t$. Once the velocity at the nodes of Γ_{free} is known, it has to be extended to the rest of the nodes. A classical possibility is to solve the Laplace problem $\Delta \boldsymbol{u}_{\text{dom}} = \boldsymbol{0}$ using $\boldsymbol{u}_{\text{dom},b}^{n+1}$ as Dirichlet boundary conditions. However, it is also possible to restrict $\boldsymbol{u}_{\text{dom}} \neq \boldsymbol{0}$ to the nodes next to $\Gamma_{\text{free}}^{n+1}$, since in our approach mesh distortion does not accumulate from one time step to another (see Fig. 4.1 for a schematic of the mesh deformation). This is in practice what we do. The condition we use in order to choose which of the nodes of the mesh are allowed to move is:

$$\text{dist}(\boldsymbol{x}_{node}, \Gamma_{\text{free}}) < K \cdot \max |\boldsymbol{u}_{\text{dom},b}| \cdot \delta t$$

where $K > 1$ is a user defined constant which adjusts the size of the region of Ω in which the mesh is deformed. This ensures that the mesh deformation is smooth enough for large values of $\frac{\max |\boldsymbol{u}_{\text{dom},b}| \cdot \delta t}{h}$, where h is the element size. An example of mesh deformation from M^n to M_{virt}^{n+1} is represented in Fig. 4.2.

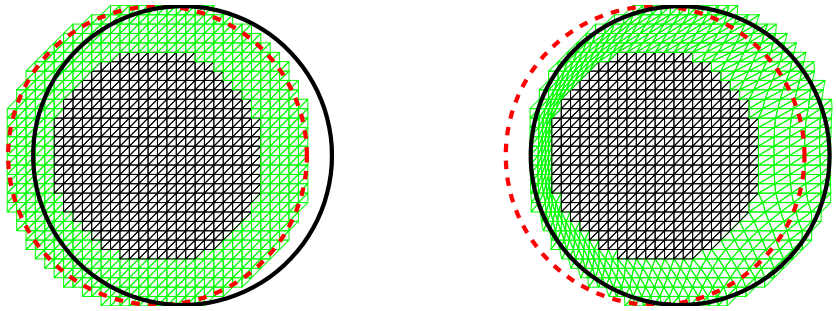


Figure 4.2: Mesh deformation. Left: M^n. Right: M_{virt}^{n+1}. The red dotted line represents Γ_{free} at time t^n, the black line corresponds to Γ_{free} at time t^{n+1}- Green elements are deformed from M^n to M_{ALE}^{n+1} while black elements remain undeformed.

Remark 3 As only nodes close to $\Gamma_{\text{free}}^{n+1}$ are displaced, the projection operations between meshes need only to be carried out in the deformed region of the mesh (green region in Fig. 4.2).

4.3.3 Step 3. Solving the flow equations I: Equations on the deformed mesh

The previous procedure defines the domain $\Omega(t^{n+1})$ and a mesh that we call $M^{n+1}_{\rm virt}$, obtained from a deformation of the mesh M^n. The equations to be solved there are (see (4.7)-(4.8)):

$$m_1^{n+1}\left(\frac{1}{\delta t}\left(\boldsymbol{u}^{n+1}_{h,\rm virt}(\boldsymbol{x})-\boldsymbol{u}^{n}_{h,\rm virt}(\boldsymbol{x}^n)\right),\boldsymbol{v}_h\right)+a^{n+1}(\boldsymbol{u}_{h,\rm virt},\boldsymbol{v}_h)$$
$$+c^{n+1}(\boldsymbol{u}_{h,\rm virt}-\boldsymbol{u}_{\rm dom,virt};\boldsymbol{u}_{h,\rm virt},\boldsymbol{v}_h)+b_1^{n+1}(p_{h,\rm virt},\boldsymbol{v}_h)=l_1^{n+1}(\boldsymbol{v}_h), \tag{4.12}$$
$$b_2^{n+1}(q_h,\boldsymbol{u}_{h,\rm virt})+s^{n+1}(q_h,p_{h,\rm virt})=l_2^{n+1}(q_h), \tag{4.13}$$

where subscript "virt" refers to the mesh $M^{n+1}_{\rm virt}$ on which these equations should now be solved using the space discretization described in Subsection 4.2.1. Let us stress once again that, as it is well known in the classical ALE approach, $\boldsymbol{u}^n(\boldsymbol{x}^n)$ is known on $M^{n+1}_{\rm virt}$ because the nodes of this mesh are obtained from the motion of the nodes of M^n with the mesh velocity $\boldsymbol{u}^{n+1}_{\rm dom,virt}$.

4.3.4 Step 4. Splitting of elements

The key idea of the FM-ALE method is *not to use* $M^{n+1}_{\rm virt}$ to solve the flow equations at time t^{n+1}, but to use instead another mesh M^{n+1} that will be a *a minor modification of the background mesh* M^0. This mesh M^{n+1} is obtained by splitting the elements of M^0 cut by $\Gamma^{n+1}_{\rm free}$, as shown in Fig. 4.1. Meshes M^{n+1} and M^0 only differ in the subelements created after the splitting just mentioned.

Mesh M^{n+1} could be thought as a local refinement of mesh M^0 to make it conform the boundary $\Gamma^{n+1}_{\rm free}$. This is certainly a possibility that can be implemented as such. Let us note however that this requires the introduction of boundary nodes at each step, as shown in Fig. 4.1, and the subsequent change in the mesh graph and in the sparsity pattern of the matrix of the final algebraic system to be solved for the arrays of nodal unknowns. As in other fixed grid methods, this computational complication can be avoided *by prescribing boundary conditions on* $\Gamma^{n+1}_{\rm free}$ *in an approximate way*. Nevertheless, this issue, in spite of its major practical importance, is not an essential concept of the FM-ALE method, and we defer its description to Section 4.4.

The local refinement from M^0 to M^{n+1} is needed also to perform the numerical integration of the different terms appearing in (4.7)-(4.8). The impact of this in the computational cost of the overall calculation is minimum.

The splitting of elements is a strictly algorithmic step that shall not be discussed here. In the case of 2D linear elements, Fig. 4.3 shows how the splitting can be done and the numerical integration points (red points) required in each triangle resulting from this splitting.

4.3.5 Step 5. Solving the flow equations II: Equations on the background mesh

Let P^{n+1} be the projection of finite element functions defined on $M^{n+1}_{\rm virt}$ to M^{n+1}. To define it, for each node of M^{n+1} the element in $M^{n+1}_{\rm virt}$ where it is placed has to be identified. Once this is done, the value of any unknown at this node can be obtained through interpolation, possibly

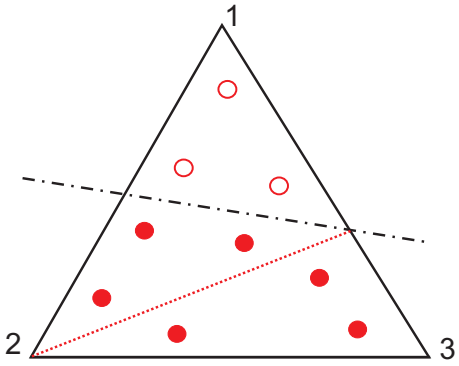


Figure 4.3: Splitting of elements

with restrictions. The way to construct this projection operator is a problem common to different situations in which transfer of information between finite element meshes is required. We describe our approach in Section 4.4.

The velocity $\boldsymbol{u}^n$ in M^{n+1}_{virt} is known because its nodal values correspond to those of mesh M^n. However, its nodal values on M^{n+1} have to be computed using the projection just described. The same happens with the mesh velocity $\boldsymbol{u}_{\text{dom}}$.

If now we define

$$\boldsymbol{u}_h^{n+1} := P^{n+1}(\boldsymbol{u}_{h,\text{virt}}^{n+1}),$$

the problem to be solved at time step $n+1$ is to find a velocity $\boldsymbol{u}_h^{n+1}$ and a pressure p_h^{n+1} such that

$$\begin{aligned} & m_1^{n+1}\left(\delta t^{-1}\left(\boldsymbol{u}_h^{n+1}(\boldsymbol{x}) - P^{n+1}(\boldsymbol{u}_{h,\text{virt}}^{n}(\boldsymbol{x}^n))\right), \boldsymbol{v}_h\right) + a^{n+1}(\boldsymbol{u}_h, \boldsymbol{v}_h) \\ & \qquad + c^{n+1}(\boldsymbol{u}_h - P^{n+1}(\boldsymbol{u}_{\text{dom,virt}}); \boldsymbol{u}_h, \boldsymbol{v}_h) + b_1^{n+1}(p_h, \boldsymbol{v}_h) = l_1^{n+1}(\boldsymbol{v}_h), \qquad (4.14) \\ & b_2^{n+1}(q_h, \boldsymbol{u}_h) + s^{n+1}(q_h, p_h) = l_2^{n+1}(q_h), \qquad (4.15) \end{aligned}$$

which again must hold for all velocity test functions $\boldsymbol{v}_h$ and pressure test functions q_h.

Note that $p_h^{n+1} \neq P^{n+1}(p_{h,\text{virt}}^{n+1})$. Pressure p_h^{n+1} is determined by imposing that $\boldsymbol{u}_h^{n+1}$ is divergence free, which at the discrete level is not equivalent to impose that $\boldsymbol{u}_{h,\text{virt}}^{n+1}$ is divergence free.

Problem (4.14)-(4.15) is posed on M^{n+1} which, as it has been said, coincides with M^0 except for the splitting of the elements crossed by the interface. Even this difference can be avoided if instead of prescribing exactly the boundary conditions an approximation is performed, for example using Nitsche's method, Lagrange multipliers or the strategy described in Section 4.4. Therefore, the goal of using a fixed mesh during the whole simulation has been achieved.

It is observed that the projection P^{n+1} has to be applied to

- $P^{n+1}(\boldsymbol{u}_{h,\text{virt}}^{n}(\boldsymbol{x}^n))$. This clarifies the effect of the mesh motion in the context of fixed-mesh methods. In particular, there is no doubt about the velocity at previous time steps of newly created nodes.

- $P^{n+1}(\boldsymbol{u}^{n+1}_{\mathrm{dom,virt}})$. The mesh velocity is computed on M^{n+1}_{virt}, and therefore needs to be projected to compute on M^{n+1}.

4.3.6 Comparison with the classical ALE approach

To conclude this section, it is important to highlight the differences between our FM-ALE approach and a classical ALE formulation:

- Given a position of the fluid front on the fixed mesh, elements cut by the front are split into subelements (only for integration purposes), so that the front coincides with the edges of the subelements.
- After deforming the mesh from one time step to the other using classical ALE procedures, results are projected back to the original mesh .
- The front is represented by a boundary function, and not by the position of the material points at Γ_{free} as in a classical ALE method.

4.4 Side numerical ingredients

In this section we describe some numerical ingredients that, in spite of being essential in the development of the FM-ALE method, are not inherent to its main concept. In other words, these ingredients may be changed without altering the main concept of the method.

4.4.1 Level set function update

In the applications, there are several ways to define Γ_{free}. In general, we assume that this part of the boundary of the flow domain is defined by what we have called generically a *boundary function*. This function may be defined analytically or by discrete means, for example through interpolation from some nodes that define the location of Γ_{free}. That would be a natural way to deal with fluid-structure interaction problems.

In some applications, it is convenient to represent Γ_{free} by a *level set function* (see [112] for an overview of these methods). This function, say ψ, will be the solution of the problem

$$\begin{aligned} \partial_t \psi + \boldsymbol{u} \cdot \nabla \psi &= 0 && \text{in } \Omega^0 \times (0,T), \\ \psi &= \overline{\psi} && \text{on } \Gamma_{\mathrm{inf}} \times (0,T), \\ \psi(\boldsymbol{x},0) &= \psi_0(\boldsymbol{x}) && \text{in } \Omega^0, \end{aligned} \tag{4.16}$$

where $\Gamma_{\mathrm{inf}} := \{\boldsymbol{x} \in \partial\Omega^0 \mid \boldsymbol{u}\cdot\boldsymbol{n} < 0\}$ is the inflow part of the domain boundary. In free surface simulations, the initial condition ψ_0 is chosen in order to define the initial position of the fluid front to be analyzed. The boundary condition $\overline{\psi}$ determines whether fluid enters or not through a certain point of the inflow boundary.

Due to the pure convective type of the equation for ψ, we use the SUPG technique for the spatial discretization. Again, the temporal evolution is treated via the standard trapezoidal rule.

If ψ is taken as a step function, numerical problems may be encountered when it is transported. It is known that small oscillations in the vicinity of sharp gradients still remain using the SUPG formulation. These oscillations may propagate and yield to distorted front shapes, specially near corners. Compared to similar methods, such as the volume-of-fluid (VOF) method [69], one particularity of the level set method is that it uses a smooth function ψ. As the smoothness can be lost as the simulation evolves, the level set function must be redefined for each mesh node as explained for example in [37].

Once ψ is computed, $\Gamma_{\text{free}}(t)$ is defined as

$$\Gamma_{\text{free}}(t) = \{\boldsymbol{x} \in \Omega^0 \mid \psi(\boldsymbol{x}, t) = 0\}.$$

Thus, $\Gamma_{\text{free}}(t)$ is simply updated by solving the problem for $\psi(\boldsymbol{x}, t)$.

The important point to be noted is that the system is solved on the whole domain Ω^0. As mentioned earlier, we approximate this problem using a stabilized finite element method. For the discrete problem it is necessary to extrapolate the velocity defined on $\Omega(t)$ to the rest of Ω^0. The question is how to perform this extrapolation. In principle, the advection velocity $\boldsymbol{u}$ in (4.16) is only needed in the neighborhood of $\Gamma_{\text{free}}(t)$, since the precise transport of ψ is not needed, except for the transport of the isovalue that defines $\Gamma_{\text{free}}(t)$. In our calculations, we have found useful to extrapolate $\boldsymbol{u}$ by solving a Stokes problem on $\Omega(t)^c = \Omega^0 \setminus \Omega(t)$. This has two main advantages with respect to a simpler extrapolation procedure, namely, the extrapolated velocity is weakly divergence free in $\Omega(t)^c$ and we can impose the correct boundary conditions for it.

4.4.2 Approximate imposition of boundary conditions

Even though we have not formulated it as such, the FM-ALE method can be considered an immersed boundary method, in the sense that $\Gamma_{\text{free}}(t)$ is a boundary that moves within a fixed domain Ω^0. From the conceptual point of view, there is no problem in imposing exactly Dirichlet boundary conditions on this part of the boundary. However, this requires the dynamic addition of mesh nodes (see Fig. 4.1, where these nodes are drawn in green), with the associated change in the sparsity of the matrix of the algebraic system to be solved mentioned earlier. This is why it is very convenient from the implementation standpoint to avoid the explicit introduction of such nodes and to prescribe boundary conditions *approximately*, for example with the method described in Chapter 2. It is important to note that this implementation maintains the connectivity of the background mesh.

4.4.3 Data transfer between finite element meshes

The last crucial ingredient in the FM-ALE approach is the transfer of information between meshes M_{virt}^{n+1} and M^{n+1} for each time step n (see Fig. 4.1). In principle, it would be possible to use a simple interpolation operator. However, it is well known that this interpolation, for example when it is of Lagrangian type, may suffer from overdiffusivity, in the sense that results on the new mesh may be damped from those of the original one. Another possibility could be to use the L^2 projection as transfer operator. We explain here how to incorporate *restrictions* to the projection between meshes. The idea described in the following was introduced in [71] in the context of transmission of information through boundaries in domain decomposition

methods. For a method particularly designed in the context of immersed boundary methods for the transfer of forces, see [141].

Let us consider two meshes, M_1 and M_2, of a domain Ω. For simplicity, we assume that both are conforming (matching $\partial\Omega$). Let n_i ($i = 1, 2$) be the number of nodes in M_i and let $\boldsymbol{\Phi}_i \in \mathbb{R}^{n_i}$ be the array of nodal values of a scalar variable ϕ. Suppose that $\boldsymbol{\Phi}_1$ is known and we want to project it onto M_2 to obtain $\boldsymbol{\Phi}_2$. If $\boldsymbol{P}_{21} \in \text{Mat}_{\mathbb{R}}(n_2, n_1)$ is the transfer operator from M_1 to M_2 (for example the standard interpolation or the L^2 projection), a simple choice would be $\boldsymbol{\Phi}_2 = \boldsymbol{P}_{21}\boldsymbol{\Phi}_1$. However, suppose that we require $\boldsymbol{\Phi}_2$ to inherit a set of properties from $\boldsymbol{\Phi}_1$, written in the form

$$\boldsymbol{R}_2\boldsymbol{\Phi}_2 = \boldsymbol{R}_1\boldsymbol{\Phi}_1, \quad \boldsymbol{R}_i \in \text{Mat}_{\mathbb{R}}(n_r, n_i), \tag{4.17}$$

where n_r is the number of restrictions to be imposed. The idea we propose is to take $\boldsymbol{\Phi}_2$ as close as possible to $\boldsymbol{P}_{21}\boldsymbol{\Phi}_1$ but satisfying (4.17). A possibility is to solve the optimization problem

$$\begin{aligned} \text{minimize} \qquad & \frac{1}{2}|\boldsymbol{\Phi}_2 - \boldsymbol{P}_{21}\boldsymbol{\Phi}_1|^2, \\ \text{under the constraint} \qquad & \boldsymbol{R}_2\boldsymbol{\Phi}_2 = \boldsymbol{R}_1\boldsymbol{\Phi}_1. \end{aligned}$$

This problem can be solved by optimizing the Lagrangian $L(\boldsymbol{\Phi}_2, \boldsymbol{\lambda})$, where $\boldsymbol{\lambda} \in \mathbb{R}^{n_r}$, given by

$$L(\boldsymbol{\Phi}_2, \boldsymbol{\lambda}) = \frac{1}{2}|\boldsymbol{\Phi}_2 - \boldsymbol{P}_{21}\boldsymbol{\Phi}_1|^2 - \boldsymbol{\lambda}^{\mathrm{t}}(\boldsymbol{R}_2\boldsymbol{\Phi}_2 - \boldsymbol{R}_1\boldsymbol{\Phi}_1).$$

This leads to the system

$$\begin{aligned} \boldsymbol{\Phi}_2 - \boldsymbol{R}_2^{\mathrm{t}}\boldsymbol{\lambda} &= \boldsymbol{P}_{21}\boldsymbol{\Phi}_1, \\ \boldsymbol{R}_2\boldsymbol{\Phi}_2 &= \boldsymbol{R}_1\boldsymbol{\Phi}_1, \end{aligned}$$

which after solving for $\boldsymbol{\Phi}_2$ yields

$$\boldsymbol{\Phi}_2 = \boldsymbol{P}_{21}\boldsymbol{\Phi}_1 + \boldsymbol{R}_2^{\mathrm{t}}(\boldsymbol{R}_2\boldsymbol{R}_2^{\mathrm{t}})^{-1}(\boldsymbol{R}_1 - \boldsymbol{R}_2\boldsymbol{P}_{21})\boldsymbol{\Phi}_1.$$

In the applications, the number of restrictions n_r is small, so that inverting $\boldsymbol{R}_2\boldsymbol{R}_2^{\mathrm{t}} \in \text{Mat}_{\mathbb{R}}(n_r, n_r)$ is computationally affordable. In the case of the FM-ALE method, a typical restriction would be for example to impose global conservation of momentum and of mass when projecting velocities from mesh M_{virt}^{n+1} to M^{n+1} for each n. In this case, $n_r = d + 1$.

4.5 A numerical example

In this section we will solve the flow over a moving cylinder with the proposed FM-ALE strategy. The objective is to apply this methodology to this simple validating example.

The corresponding flow equations are those described in Section 4.2, although in this case a multi-step time discretization will be used. In particular, we will use the second order backward differentiation scheme (BDF2), in which the time derivative at time $n + 1$ is approximated as:

$$\left.\frac{\partial u}{\partial t}\right|^{n+1} \approx \frac{1}{\delta t}\left(\frac{3}{2}u^{n+1} - 2u^n + \frac{1}{2}u^{n-1}\right).$$

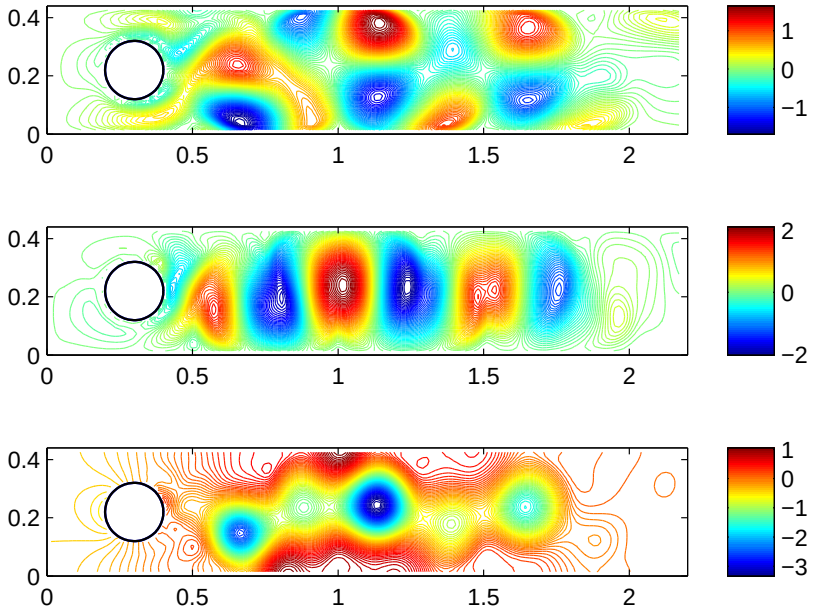


Figure 4.4: Solution at $t = 3$. From top to bottom: x-velocity, y-velocity, pressure.

The strategy described in Chapter 2 will be used to prescribe Dirichlet type boundary conditions on the surface of the moving solid, in this case the cylinder.

The *hold-all domain* is the rectangle $B = [0, 2.2] \times [0, 0.44]$. A background mesh of 9000 linear triangles has been used. The considered solid is a cylinder of diameter $D = 0.2$, its trajectory being defined by the position of its center:

$$x_c(t) = 1.1 + 0.8 \sin\left(\frac{2\pi}{3}(t - 0.75)\right),$$
$$y_c(t) = 0.22.$$

The velocity is prescribed to $(0, 0)$ on the walls of the rectangular domain, except for the wall corresponding to $x = 2.2$, where it is left free, whereas it matches the cylinder velocity on the cylinder surface. Note that the flow is due only to the cylinder movement. Viscosity is set to 0.001, so that the maximum Reynolds number is $\mathrm{Re} \approx 300$ based on the cylinder diameter and the (maximum) velocity when the cylinder is located at the central section of the rectangle. The time step size has been set to $\delta t = 0.05$, and 60 time steps (a full period) have been performed, after which the flow is considered to be fully developed.

Fig. 4.4 shows the results obtained at time $t = 3$. We would like to remark the smoothness of the velocity field close to the cylinder surface.

It is also interesting to see which are the differences between the treatment of the newly created nodes in the proposed FM-ALE approach and other usual procedures. To this end we compare nodal values for newly created nodes at time t^n (in the time step which goes from t^n to t^{n+1}) for the FM-ALE approach (information is convected and projected) and for the more

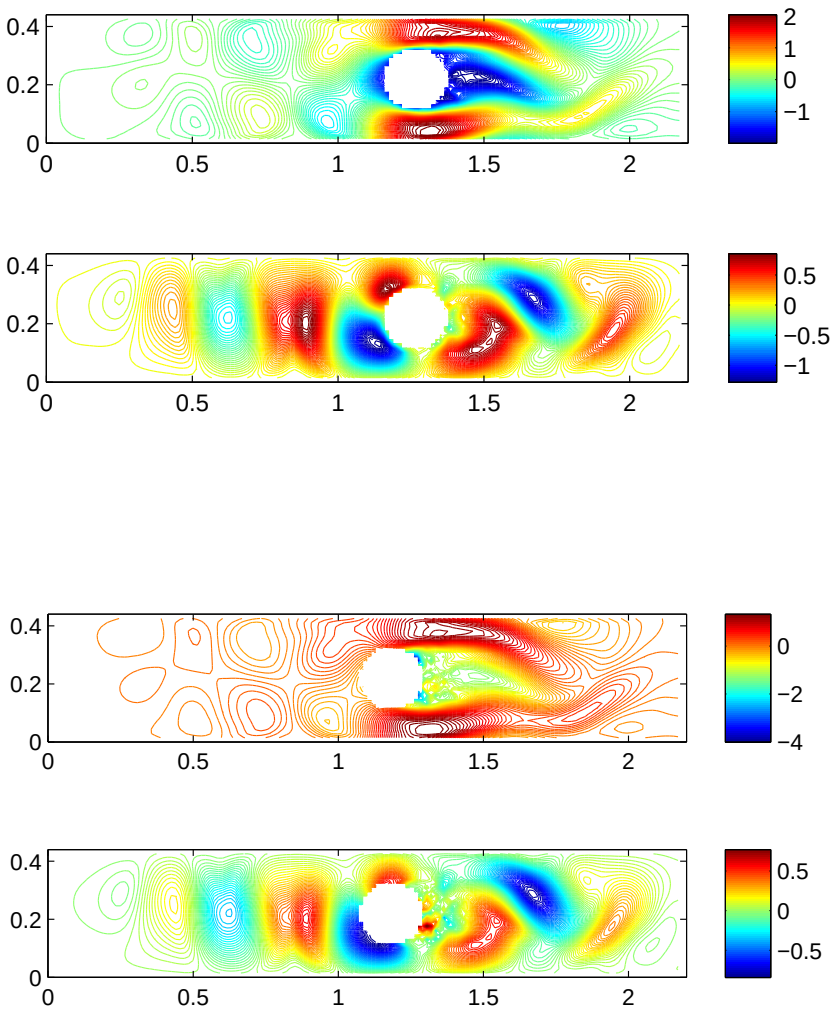


Figure 4.5: Solution at $t = 2.25$, extrapolation procedure. From the top to the bottom: x-velocity before extrapolating, y-velocity before extrapolating, x-velocity after extrapolating, y-velocity after extrapolating.

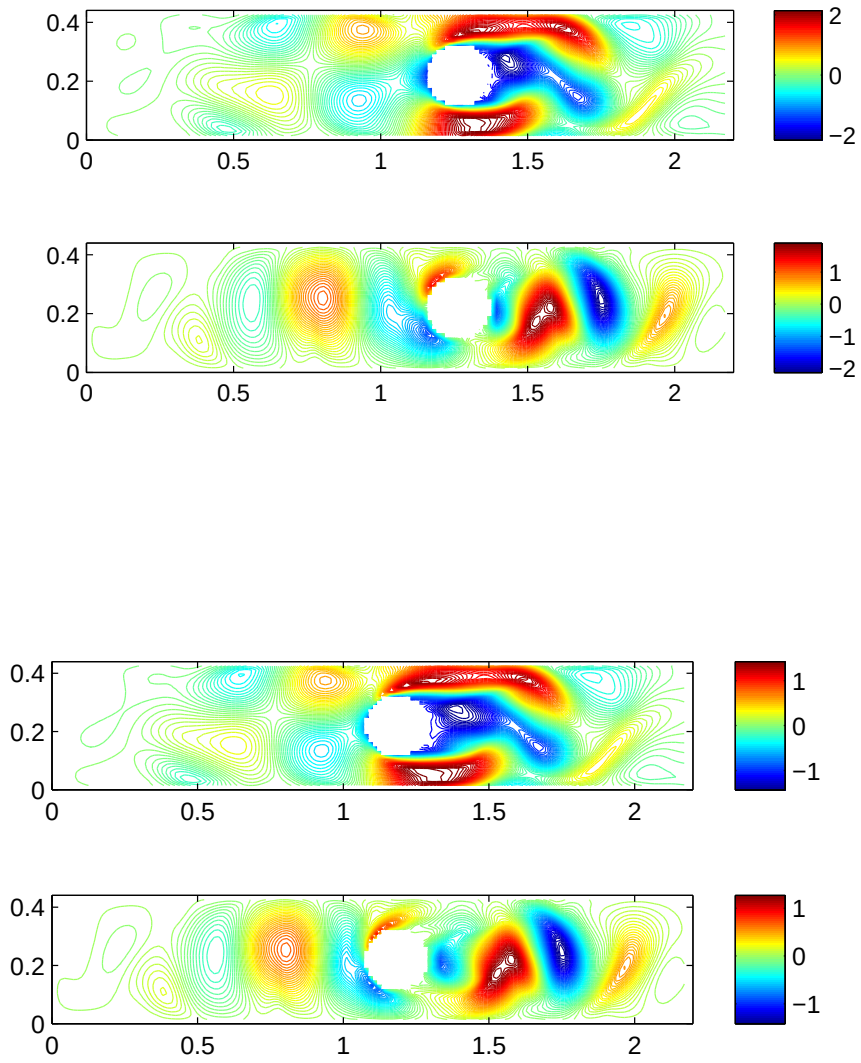


Figure 4.6: Solution at $t = 2.25$, FM-ALE procedure. From the top to the bottom: x-velocity before convection-projection, y-velocity before convection-projection, x-velocity after convection-projection, y-velocity after convection-projection.

usual procedure of extrapolating values from neighboring nodes mentioned earlier.

Fig. 4.5 and Fig. 4.6 show velocity values (before and after the convection-projection or the extrapolation procedures) at $t^n = 2.25$. It can be seen that for large incremental displacements, as those of the time step we are considering, extrapolated values differ significantly from convected-projected values, and are much less smooth. Also, the values before the convection-projection or extrapolation procedure are smoother for the FM-ALE approach. We would like to stress that, contrary to the convection-projection of the FM-ALE method, the extrapolation procedure lacks physical grounds.

4.6 Conclusions

In this chapter we have introduced in detail the concept of the FM-ALE approach. Succinctly, it consists in using the standard ALE method but "remeshing" at each time step so as to use always the same given mesh, which discretizes the whole region where the flow takes place.

The first benefit is conceptual. Ad-hoc approximations to account for the advection of information that can be found in several fixed-grid methods are avoided. This is in particular reflected by the treatment of the so called newly created nodes. When a node "dry" in one time step becomes part of the flow region in the next time step, the value of the flow variables to be assigned there to approximate (local) time derivatives is perfectly determined.

It has been our intention to clearly distinguish the main concept of the formulation from other related issues, and in particular from the approximate imposition of boundary conditions. Nevertheless, the way to carry out this imposition is essential for the success of the method. We have described our particular approach. Some remarks concerning the transfer of information between meshes have also been made, and the possibility to model the moving surface by level set functions has been explained.

A numerical example has been presented which shows the performance of the method in a simple validating example. Results have been compared to those of other fixed grid methods, showing the need of correctly computing the advection of information between time steps.

Another natural application of the FM-ALE approach is the numerical approximation of fluid-structure interaction problems, which we will deal with in the following chapter.

Chapter 5

The Fixed-Mesh ALE approach applied to Solid Mechanics and FSI problems

In this chapter we propose a method to solve Solid Mechanics and Fluid-Structure Interaction problems using always a fixed background mesh for the spatial discretization. The main feature of the method is that it properly accounts for the advection of information as the domain boundary evolves. To achieve this, we use an arbitrary Lagrangian-Eulerian framework, the distinctive characteristic being that at each time step results are projected onto a fixed, background mesh. For solid mechanics problems subject to large strains the Fixed Mesh - ALE method avoids the element stretching found in fully Lagrangian approaches. For FSI problems FM-ALE allows for the use of a single background mesh to solve both the fluid and the structure. We also apply the FM-ALE method to the problem of floating solids, in which it is used together with the level set function method.

5.1 Introduction

The Fixed Mesh ALE method (FM-ALE from now on) is a fixed grid method its main feature being that the domain movement is taken into account when computing the temporal derivatives. The basic idea consists in using an ALE (Arbitrary Lagrangian-Eulerian) strategy and remeshing at each time step in such a way that the original fixed mesh is recovered. This has two main advantages when compared to other fixed grid methods:

1. Since an ALE formulation is used, temporal derivatives can be correctly computed, including the convective terms arising due to the domain movement.

2. The values of the variables in previous time steps are clearly defined in the so-called *newly created nodes*, an issue of particular controversy in most fixed grid methods.

ALE formulations were initially developed for fluid dynamics problems, in which they were necessary to cope with Fluid Structure Interaction (FSI) and free surface problems (see References [45, 74, 79]). In classical ALE methods remeshing is often necessary after a certain number of time steps in order to avoid element stretching. The FM-ALE method avoids this need by projecting the results from the ALE deformed mesh onto a fixed background mesh

at each time step, prior to solving the flow equations. At the end all the calculations can be performed on the fixed mesh, and in fact the ALE deformed mesh does not need to be explicitly built.

The FM-ALE method for flow problems in moving domains is extensively described in Chapter 4: the main algorithmic steps of the method are set a numerical example in the field of flow problems are presented. Here we take the same ideas and we apply them to problems in solid mechanics and Fluid-Structure Interaction.

The most usual approach to solve solid mechanics problems is the use of Lagrangian formulations. This means that equations are written for *material* points following the movement of particles. This is a natural choice since in solid mechanics we are interested in tracking the behavior of structures in time (contrary to many problems of fluid mechanics where one is interested on the *effect* of the flow in a certain region, leading to Eulerian formulations). However, there are certain problems in which fully Lagrangian formulations cannot be used or lead to numerical difficulties: when a solid body is subject to large strains the shape of the elements which form the mesh can change a lot, resulting in stretched elements. Stretched elements cause that the system of equations to be solved is ill-conditioned, an inconvenient of particular importance if iterative methods are to be used. In this case ALE formulations are used and the mesh is no longer deformed following the particles but is given an arbitrary movement which avoids the stretching of elements.

ALE methods for solid mechanics problems have been extensively developed (see [99, 18, 59]). The main concern in these works is to correctly compute the stress and plastic history variables update, since values of history variables at the previous time step are not available at the quadrature points unless a fully Lagrangian approach is used. In the framework of ALE strategies for solid mechanics problems, the FM-ALE method can be understood as an ALE method in which the mesh velocity is set to zero in all the domain except in the region close to the body surface. In [111] an Eulerian formulation for large deformation solid dynamics is presented. However, it is not clear how the issue of newly created nodes near the boundary is treated.

Once the FM-ALE strategy has been applied to both flow and solid mechanics problems it is very natural to consider its use in the area of Fluid-Structure Interaction (FSI). Several fixed grid strategies to solve FSI problems have been developed in the past years. As a first example, the immersed boundary method ([113, 87, 91, 92, 137]) consists in adding punctual penalty forces in the domain boundary so that boundary conditions are fulfilled. Another possible approach is the use of Lagrange multipliers to enforce boundary conditions (see [70, 14, 82]). Both approaches are fictitious domain methods ([62, 63]) in the sense that the fluid-structure interface divides the fluid domain in a physical flow field and a fictitious field, which may be discretized and solved, but has no physical meaning to the FSI problem. Usually the unknown fields in this fictitious domain are used to assign values to the *newly created nodes* in the computation of time derivatives. In the *extended* finite element method, special functions are used to enrich the finite element space near the interface. In [89] a fixed mesh is used to solve the fluid while the solid is treated by a Lagrangian description. The description of the fluid-solid interface is done by means of a level set function. In all these works, a fully Lagrangian approach is used to deal with the solid.

An interesting feature of using the FM-ALE method to solve FSI problems is that since the regions occupied by the fluid and the solid do not superimpose, a single mesh can be used,

giving some of the elements to the solid mechanics problem and the others to the flow problem. Special care has to be given to the coupling conditions between fluid and structure: the usual partitioned methods can be used with the FM-ALE method, although due to the fact that the same mesh is used to solve both problems a monolithic approach seems more suitable.

We finally use the FM-ALE method to solve the problem of solid bodies falling into water. This involves the additional need of tracking the free surface of the fluid by means of a level set function. Several works have been already developed in the field of the simulation of floating solids. In [138] floating bodies are simulated by means of the QALE-FEM method, but the solid displacements in the numerical examples are small and there is no need to remesh. In [136] the finite element method is used to simulate the interaction between waves and a floating body, but again it focuses in the case in which the solid body displacements are small. A fixed grid strategy for the simulation of solids falling into water has been used in [98], where the impact of a cylindrical object on a water surface is studied, the main difference with respect to the present approach being the way newly created nodes are treated. Floating bodies can also be treated with the Chimera strategy described in [72], provided the free surface is considered as the interface between the fluid analyzed and a fictitious one, for example air. The flow problem would become in this case a two-phase flow rather than a free surface problem, and a possible way to deal with it is explained in [40]. ALE approaches are also possible for the simulation of free surface - fluid structure interaction problems, as done for example in [94], but they require rebuilding the finite element mesh when this mesh gets too distorted. In [43] the phase-field method is used to analyze the wetting phenomena of the impact of a sphere with a free surface, The novelty of the present work with respect to the previous ones is the use of a fixed mesh strategy which correctly takes into account the movement of the fluid domain at the time of computing the ALE convective terms and time derivatives.

The chapter is organized as follows. A review of ALE methods applied to solid mechanics problems is presented in Section 5.2. Firstly the general ALE formulation is presented and particularized to the solid mechanics conservation laws. Afterwards the two possible approaches to face the equations are discussed: the monolithic approach deals with the arising equations in a classical manner, while in fractional - step methods the equations are solved in two steps: the material and the convective phases. This allows for the use of specific numerical methods to solve each of the phases. In Section 5.3 the FM-ALE method is described. Since a detailed explanation of the method can be found in the previous chapter, only the general algorithm and the particular features of its application to solid mechanics problems are presented. Stress is put in critical issues such as the imposition of boundary conditions or the tracking of the solid body surface. Section 5.4 deals with the FM-ALE method applied to FSI problems. The equations for the coupled problem are presented. A description of some of the most common coupling strategies and their particularization to FM-ALE follows. Section 5.5 particularizes the application of the FM-ALE method to the problem of solid bodies falling into water. In this section we describe how the tracking of the free surface is done, and which are the additional computational challenges in the interaction of the free surface with the solid body.

Finally in Section 5.6 some numerical examples and validation tests are carried out, showing the behavior of the proposed methodology. Some conclusions close the chapter in Section 5.7.

5.2 ALE methods applied to solid mechanics

5.2.1 Problem statement

Let us consider a region $\Omega^0 \subset \mathbb{R}^d$ ($d = 2, 3$) where a solid body moves through during a time interval $[0, T]$. The solid at time t occupies only a subdomain $\Omega(t) \subset \Omega^0$. The boundary of $\Omega(t)$ is defined by part of $\partial\Omega^0$ and a moving boundary that we call $\Gamma_{\text{free}} = \partial\Omega(t) \setminus \partial\Omega^0 \cap \partial\Omega(t)$.

In order to cope with the time-dependency of $\Omega(t)$, we use the ALE approach, with the particular feature of considering a variable definition of the domain velocity. Let $\boldsymbol{\chi}_t$ be a family of invertible mappings, which for all $t \in [0, T]$ map a point $\boldsymbol{X} \in \Omega(0)$ to a point $\boldsymbol{x} = \boldsymbol{\chi}_t(\boldsymbol{X}) \in \Omega(t)$, with $\boldsymbol{\chi}_0 = \boldsymbol{I}$, the identity. If $\boldsymbol{\chi}_t$ is given by the motion of the particles, the resulting formulation would be Lagrangian, whereas if $\boldsymbol{\chi}_t = \boldsymbol{I}$ for all t, $\Omega(t) = \Omega(0)$ and the formulation would be Eulerian.

Let now $t' \in [0, T]$, with $t' \leq t$, and consider the mapping

$$\begin{aligned} \boldsymbol{\chi}_{t,t'} : \Omega(t') &\longrightarrow \Omega(t) \\ \boldsymbol{x}' &\mapsto \boldsymbol{x} = \boldsymbol{\chi}_t \circ \boldsymbol{\chi}_{t'}^{-1}(\boldsymbol{x}'). \end{aligned}$$

Given a function $f : \Omega(t) \times (0, T) \longrightarrow \mathbb{R}$ we define

$$\left.\frac{\partial f}{\partial t}\right|_{\boldsymbol{x}'} (\boldsymbol{x}, t) := \frac{\partial (f \circ \boldsymbol{\chi}_{t,t'})}{\partial t}(\boldsymbol{x}', t), \quad \boldsymbol{x} \in \Omega(t),\ \boldsymbol{x}' \in \Omega(t').$$

In particular, the domain velocity taking as a reference the coordinates of $\Omega(t')$ is given by

$$\boldsymbol{u}_{\text{dom}} := \left.\frac{\partial \boldsymbol{x}}{\partial t}\right|_{\boldsymbol{x}'} (\boldsymbol{x}, t). \tag{5.1}$$

Three conservation laws are fundamental in solid mechanics, namely mass, momentum and energy balance. Let us make the assumption that mechanical effects are uncoupled from thermal effects. In this case, equations for mass and momentum balance can be solved independently from the energy balance equation. The solid mechanics problem formulated in $\Omega(t)$, accounting also for the motion of this domain, can be written as follows:

$$\left.\frac{\partial \rho}{\partial t}\right|_{\boldsymbol{x}'} + (\boldsymbol{u} - \boldsymbol{u}_{\text{dom}}) \cdot \nabla \rho = -\rho \nabla \cdot \boldsymbol{u}, \tag{5.2}$$

$$\rho \left.\frac{\partial \boldsymbol{u}}{\partial t}\right|_{\boldsymbol{x}'} + \rho(\boldsymbol{u} - \boldsymbol{u}_{\text{dom}}) \cdot \nabla \boldsymbol{u} = \nabla \cdot \boldsymbol{\sigma} + \rho \boldsymbol{b}, \tag{5.3}$$

where ρ is the solid density, $\boldsymbol{u}$ is the particle velocity, $\boldsymbol{\sigma}$ is the Cauchy stress tensor and $\boldsymbol{b}$ is the vector of body forces.

It is usual in the field of solid mechanics to use the following equation which relates the density ρ in a given configuration with the density ρ_0 at the undeformed configuration:

$$\rho J = \rho_0, \tag{5.4}$$

at each material point, where

$$\boldsymbol{F} = \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{X}}, \quad J = \det(\boldsymbol{F}).$$

As long as the material surfaces which compose the boundary of the solid are tracked with enough accuracy , this allows to avoid solving (5.2), and solving only for (5.3). An additional constitutive equation that relates $\boldsymbol{\sigma}$ and $\boldsymbol{u}$ will be needed so that the problem is well posed.

If path dependent constitutive equations are to be used, material derivatives of the plastic internal variables have to account for the advection effects, leading to an equation for them of the form:

$$\left.\frac{\partial \boldsymbol{\alpha}}{\partial t}\right|_{\boldsymbol{X}} = \left.\frac{\partial \boldsymbol{\alpha}}{\partial t}\right|_{\boldsymbol{x}'} + (\boldsymbol{u} - \boldsymbol{u}_{\text{dom}}) \cdot \nabla \boldsymbol{\alpha} = \mathcal{F}(\boldsymbol{\lambda}), \tag{5.5}$$

where $\boldsymbol{\alpha}$ is the set of plastic internal variables and $\boldsymbol{\lambda}$ is the set of variables of interest of the problem, which would typically include the plastic internal variables plus the displacements, velocity and acceleration fields. The right-hand-side of (5.5) denotes a problem-dependent operator $\mathcal{F}$ applied to $\boldsymbol{\lambda}$.

Initial and boundary conditions have to be appended to problem (5.3). Usual boundary conditions are used for both Γ_{free} and $\partial\Omega^0$:

$$\begin{aligned} \boldsymbol{u} &= \bar{\boldsymbol{u}} \quad \text{on } \Gamma_D, \\ \boldsymbol{n} \cdot \boldsymbol{\sigma} &= \bar{\boldsymbol{t}} \quad \text{on } \Gamma_N, \end{aligned} \tag{5.6}$$

where $\boldsymbol{n}$ is the external normal to the boundary and $\bar{\boldsymbol{t}}$ are the given boundary data. Γ_D and Γ_N are respectively the Dirichlet and the Neumann parts of the boundary $\partial\Omega(t)$.

To shorten the notation, we will introduce the convection velocity

$$\boldsymbol{c} = \boldsymbol{u} - \boldsymbol{u}_{\text{dom}}$$

in what follows.

5.2.2 The time-discrete problem

Let us introduce some notation. Consider a uniform partition of $[0, T]$ into Ntime intervals of length δt. Let us denote by f^n the approximation of a time dependent function f at time level $t^n = n\delta t$. We will also denote

$$\begin{aligned} \delta f^{n+1} &= f^{n+1} - f^n, \\ \delta_t f^{n+1} &= \frac{f^{n+1} - f^n}{\delta t}, \\ f^{n+\theta} &= \theta f^{n+1} + (1-\theta) f^n, \quad \theta \in [1/2, 1]. \end{aligned}$$

θ type schemes Suppose we are given a computational domain at time t^n, with spatial coordinates labeled $\boldsymbol{x}^n$, and an equation of the form:

$$\frac{\partial v}{\partial t} + \boldsymbol{c} \cdot \nabla v = \mathcal{G}(v),$$

where v is the unknown function and $\mathcal{G}$ is an operator applied to it. If v^n is known, v^{n+1} can now be found as the solution of the problem:

$$\delta_t v^{n+1}\big|_{\boldsymbol{x}^n} + \boldsymbol{c}^{n+\theta} \cdot \nabla v^{n+\theta} = \mathcal{G}(v^{n+\theta}), \tag{5.7}$$

where now $\delta_t v^{n+1}|_{\boldsymbol{x}^n} = (v^{n+1}(\boldsymbol{x}) - v^n(\boldsymbol{x}^n))/\delta t$, being $\boldsymbol{x} = \boldsymbol{\chi}_{t^{n+\theta},t^n}(\boldsymbol{x}^n)$ the spatial coordinates in $\Omega(t^{n+\theta})$. The domain velocity given by (5.1), with $\boldsymbol{x}' = \boldsymbol{x}^n$, is approximated as

$$\boldsymbol{u}_{\mathrm{dom}}^{n+\theta} = \frac{1}{\theta\delta t}\left(\boldsymbol{\chi}_{t^{n+\theta},t^n}(\boldsymbol{x}^n) - \boldsymbol{x}^n\right). \tag{5.8}$$

which allows us to compute $\boldsymbol{c}^{n+\theta} = \boldsymbol{u}^{n+\theta} - \boldsymbol{u}_{\mathrm{dom}}^{n+\theta}$ in (5.7).

Fractional step methods for solid mechanics There are basically two ways of dealing with the ALE system of equations (5.2) to (5.6) (see [117] and the references therein):

a) solving the fully coupled system of equations, accounting for the various terms simultaneously,

b) using a fractional-step method to treat material and convective effects separately.

Although solving the coupled system of equations is more accurate, the fractional step method offers some very useful advantages. On one hand, each of the equations to be solved is simpler than the ones arising from the coupled problem. On the other, difficulties on the computation of the stress field gradient, which are due to the fact that stresses are usually discontinuous across element edges, are more easily circumvented.

Remark 1 The FM-ALE method presented in this work has no dependence on the way the system of equations (5.2) to (5.6) is dealt with. However, for ease of implementation, fractional step schemes have been chosen in the numerical examples presented in Section 5.6. △

Let us consider the θ type scheme in (5.7). For simplicity we will consider θ = 1. This equation can be solved in a monolithic way, but it can also be divided in two phases:

Material phase (first order splitting)

In the first phase we solve:

$$\frac{v^{n+1}(\boldsymbol{x}_{\mathrm{mat}}) - v^n(\boldsymbol{x}^n)}{\delta t} = \mathcal{G}(v^{n+1}(\boldsymbol{x}_{\mathrm{mat}})), \tag{5.9}$$

where $\boldsymbol{x}_{\mathrm{mat}} = \boldsymbol{X}_{t^{n+1},t^n}(\boldsymbol{x}^n)$ is the mapping given by the motion of the particles. Note that this first phase corresponds to $\boldsymbol{u}_{\mathrm{dom}} = \boldsymbol{u}$ ($\boldsymbol{c} = \boldsymbol{0}$), that is to say, to a fully Lagrangian approach.

Convective phase (first order splitting) In the second phase we solve:

$$\frac{v^{n+1}(\boldsymbol{x}) - v^{n+1}(\boldsymbol{x}_{\mathrm{mat}})}{\delta t} + \boldsymbol{c}^{n+1}(\boldsymbol{x}) \cdot \nabla v^{n+1}(\boldsymbol{x}) = 0, \tag{5.10}$$

where $\boldsymbol{x} = \boldsymbol{\chi}_{t^{n+1},t^n}(\boldsymbol{x}^n)$ are the spatial coordinates in $\Omega(t^{n+1})$.

If we add (5.9) and (5.10) we obtain:

$$\frac{v^{n+1}(\boldsymbol{x}) - v^n(\boldsymbol{x}^n)}{\delta t} + \boldsymbol{c}^{n+1}(\boldsymbol{x}) \cdot \nabla v^{n+1}(\boldsymbol{x}) = \mathcal{G}(v^{n+1}(\boldsymbol{x}_{\mathrm{mat}})), \tag{5.11}$$

which corresponds exactly to (5.7) except for the fact that instead of evaluating $\mathcal{G}(\boldsymbol{v}^{n+1})$ at $\boldsymbol{x}$ we evaluate it at $\boldsymbol{x}_{\mathrm{mat}}$. This introduces an error of $\mathcal{O}(\delta t)$: observe from (5.10) that $\|v^{n+1}(\boldsymbol{x}) - v^{n+1}(\boldsymbol{x}_{\mathrm{mat}})\| = \mathcal{O}(\delta t)$, where $\|\cdot\|$ may be taken for example as the L^2-norm.

If one wants a second order in time scheme, this could be achieved by modifying scheme (5.9) - (5.10) in the following manner:

Material phase (second order splitting)

$$\frac{v^{n+1}(\boldsymbol{x}_{\mathrm{mat}}) - v^n(\boldsymbol{x}^n)}{\delta t} + \boldsymbol{c}^n \cdot \nabla v^n(\boldsymbol{x}) = \mathcal{G}(v^{n+1}(\boldsymbol{x}_{\mathrm{mat}})). \tag{5.12}$$

Convective phase (second order splitting)

$$\frac{v^{n+1}(\boldsymbol{x}) - v^{n+1}(\boldsymbol{x}_{\mathrm{mat}})}{\delta t} + \boldsymbol{c}^{n+1} \cdot \nabla v^{n+1}(\boldsymbol{x}) - \boldsymbol{c}^n \cdot \nabla v^n(\boldsymbol{x}) = 0.$$

Note that $\boldsymbol{c}^{n+1} \cdot \nabla v^{n+1}(\boldsymbol{x}) - \boldsymbol{c}^n \cdot \nabla v^n(\boldsymbol{x})$ is expected to be, formally, of first order in δt, and therefore $\|v^{n+1}(\boldsymbol{x}) - v^{n+1}(\boldsymbol{x}_{\mathrm{mat}})\| = \mathcal{O}(\delta t^2)$. Thus, when $v^{n+1}(\boldsymbol{x}_{\mathrm{mat}})$ is used in (5.12) instead of $v^{n+1}(\boldsymbol{x})$, the resulting splitting error is expected to be $\mathcal{O}(\delta t^2)$. If an overall second order scheme is to be used, $\theta = 1/2$ must be chosen.

These fractional step schemes can be introduced to the system of equations (5.2) to (5.6) and also for the plastic internal variables $\boldsymbol{\alpha}$ whose evolution equation is given by (5.5).

Newmark's method If in the constitutive equation which relates the stress tensor $\boldsymbol{\sigma}$ with the set of variables of interest of the problem there is a dependence on the displacement field $\boldsymbol{d}$, that is to say $\boldsymbol{\sigma} = \boldsymbol{\sigma}(\boldsymbol{d}, \boldsymbol{\alpha})$, (5.3) becomes a second order in time equation. In [107], Newmark presented a method to discretely approximate the velocity and acceleration ($\boldsymbol{a}$) at time t^{n+1} as a function of displacements ($\boldsymbol{d}$), velocity and acceleration at time t^n in a Lagrangian framework. These three fields can be related in the continuous case by means of the equations

$$\left.\frac{\partial \boldsymbol{u}}{\partial t}\right|_{\boldsymbol{X}} = \boldsymbol{a}, \quad \left.\frac{\partial \boldsymbol{d}}{\partial t}\right|_{\boldsymbol{X}} = \boldsymbol{u}.$$

Newmark method reads:

$$\begin{aligned} \boldsymbol{a}^{n+1} &= \frac{1}{\beta \delta t^2}[\boldsymbol{d}^{n+1} - \boldsymbol{d}^n - \boldsymbol{u}^n \delta t] - \left(\frac{1}{2\beta} - 1\right)\boldsymbol{a}^n, \\ \boldsymbol{u}^{n+1} &= \frac{\gamma}{\beta \delta t}[\boldsymbol{d}^{n+1} - \boldsymbol{d}^n] + \left(1 - \frac{1}{2\beta}\right)\delta t \boldsymbol{a}^n, \end{aligned} \tag{5.13}$$

where β and γ are parameters to be chosen. Most usual values are $\beta = 1/4$ and $\gamma = 1/2$, which provide a second order stable and non dissipative scheme. In the case of displacement-dependent stress tensors, this method is to be used instead of θ type schemes. For the sake of conciseness, we will restrict what follows to θ schemes, both in the monolithic and fractional step versions. The dependence of $\boldsymbol{\sigma}$ on the rest of variables of the problem (including internal variables $\boldsymbol{\alpha}$) will be simply indicated by $\boldsymbol{\sigma} = \boldsymbol{\sigma}(\boldsymbol{\lambda})$.

5.2.3 The fully discrete problem

The next step is to consider the spatial discretization of the time discrete problem for both the coupled and fractional-step methods. Here we present the discretization obtained if finite elements are used.

Let $\{\Omega^e\}^{n+1}$ be a finite element partition of the domain $\Omega(t^{n+1})$, with index e ranging from 1 to the number of elements n_{el}. We denote with a subscript h the finite element approximation to the unknown functions. The test functions for the velocity $\boldsymbol{u}_h$ will be denoted by $\boldsymbol{v}_h$, whereas $\boldsymbol{\gamma}_h$ will be the test functions for the discrete internal variables $\boldsymbol{\alpha}_h$, the finite element approximation to the solution of (5.5). All the unknowns and test functions are referred to the current configuration of the solid.

The standard Galerkin method applied to the monolithic time discretized problem reads: find $\boldsymbol{u}_h^{n+1}$ and $\boldsymbol{\alpha}_h^{n+1}$ such that

$$m_s^{n+\theta}\left(\delta_t \boldsymbol{u}_h^{n+1}\big|_{\boldsymbol{x}^n}, \boldsymbol{v}_h\right) + a_s^{n+\theta}(\boldsymbol{\lambda}_h, \boldsymbol{v}_h) + c_s^{n+\theta}(\boldsymbol{c}_h; \boldsymbol{u}_h, \boldsymbol{v}_h) = l_s^{n+\theta}(\boldsymbol{v}_h), \tag{5.14}$$

$$(\delta_t \boldsymbol{\alpha}_h^{n+1}\big|_{\boldsymbol{x}^n}, \boldsymbol{\gamma}_h) + (\boldsymbol{c}_h^{n+\theta} \cdot \nabla \boldsymbol{\alpha}_h^{n+\theta}, \boldsymbol{\gamma}_h) = (\mathcal{F}^{n+\theta}(\boldsymbol{\lambda}_h), \boldsymbol{\gamma}_h), \tag{5.15}$$

for all appropriate test functions $\boldsymbol{v}_h$ and $\boldsymbol{\gamma}_h$. The different forms appearing in (5.14) are given by

$$\begin{aligned}
m_s(\delta_t \boldsymbol{u}_h, \boldsymbol{v}_h) &= \int_\Omega \boldsymbol{v}_h \cdot \rho\, \delta_t \boldsymbol{u}_h, \\
a_s(\boldsymbol{\lambda}_h, \boldsymbol{v}_h) &= \int_\Omega \nabla \boldsymbol{v}_h : \boldsymbol{\sigma}(\boldsymbol{\lambda}_h), \\
c_s(\boldsymbol{c}_h; \boldsymbol{u}_h, \boldsymbol{v}_h) &= \int_\Omega \boldsymbol{v}_h \cdot (\rho\, \boldsymbol{c}_h \cdot \nabla \boldsymbol{u}_h), \\
l_s(\boldsymbol{v}_h) &= \int_{\Gamma_N} \boldsymbol{v}_h \cdot \bar{\boldsymbol{t}} + \int_\Omega \boldsymbol{v}_h \cdot \rho \boldsymbol{b},
\end{aligned}$$

where $\boldsymbol{c}_h$ is the discrete convection velocity, defined as:

$$\boldsymbol{c}_h = \boldsymbol{u}_h - \boldsymbol{u}_{\text{dom}}.$$

The superscript $n+\theta$ in the different terms of (5.14) indicates the time level where unknowns and time dependent functions need to be evaluated, as well as the spatial domain where integrals need to be performed. In (5.15) the symbol $(\cdot,\cdot)$ denotes the L^2-inner product in this spatial domain.

The test functions $\boldsymbol{v}_h$ in (5.14) must vanish at the Dirichlet part of the boundary Γ_D. Since $\mathcal{F}$ in (5.15) is usually an algebraic operator, functions $\boldsymbol{\gamma}_h$ need to vanish only at points $\boldsymbol{x}^n$, at which the temporal derivatives in (5.14)-(5.15) are referred.

Remark 2 When diffusion is small in a convection-diffusion process or, as in the case of (5.15), the process is purely convective (which happens when $\mathcal{F}$ is an algebraic operator), the Galerkin method fails and stabilized methods need to be used. The method we use is SUPG (see [29] for an overview of stabilization methods), which applied for example to (5.15) leads to the modification of this equation to

$$\begin{aligned}
&(\delta_t \boldsymbol{\alpha}_h^{n+1}\big|_{\boldsymbol{x}^n}, \boldsymbol{\gamma}_h + \tau \boldsymbol{c}_h^{n+\theta} \cdot \nabla \boldsymbol{\gamma}_h) + (\boldsymbol{c}_h^{n+\theta} \cdot \nabla \boldsymbol{\alpha}_h^{n+\theta}, \boldsymbol{\gamma}_h + \tau \boldsymbol{c}_h^{n+\theta} \cdot \nabla \boldsymbol{\gamma}_h) \\
&\quad = (\mathcal{F}^{n+\theta}(\boldsymbol{\lambda}_h), \boldsymbol{\gamma}_h + \tau \boldsymbol{c}_h^{n+\theta} \cdot \nabla \boldsymbol{\gamma}_h),
\end{aligned} \tag{5.16}$$

where the so called stabilization parameter τ is computed elementwise as

$$\tau_e = \left(c \frac{|\boldsymbol{c}|_e}{h_e} \right)^{-1}, \quad e = 1, ..., n_{\text{el}},$$

where h_e is the element size for linear elements and half of it for quadratics and $|\boldsymbol{c}|_e$ is a characteristic value of $|\boldsymbol{c}|$ on element e. In the numerical experiments we have taken the algorithmic constant $c = 2$. Stabilization might also be necessary if the coupled method is used. However, it is not needed in most solid mechanics simulations since the convective term is usually not dominant in equation (5.14). This method corresponds to the algebraic version of the subgrid scale approach (see [76]) and circumvents the stability problems of the Galerkin method. In particular, in the case of incompressible materials it is possible to use equal velocity pressure interpolations, that is, we are not tight to the satisfaction of the inf-sup stability condition. $\triangle$

For the fractional-step approach, equations (5.14)-(5.15) may be split into material and convective phases. Using $\theta = 1$ and a first order splitting, the former would consist in finding $\boldsymbol{u}_h^{L,n+1}$ and $\boldsymbol{\alpha}_h^{L,n+1}$ such that

$$\frac{1}{\delta t} m_s^{L,n+1} \left(\boldsymbol{u}_h^{L,n+1} - \boldsymbol{u}_h^n, \boldsymbol{v}_h \right) + a_s^{L,n+1}(\boldsymbol{\lambda}_h, \boldsymbol{v}_h) = l_s^{L,n+1}(\boldsymbol{v}_h) \quad \forall \boldsymbol{v}_h, \tag{5.17}$$

$$\frac{1}{\delta t} (\boldsymbol{\alpha}_h^{L,n+1} - \boldsymbol{\alpha}_h^n, \boldsymbol{\gamma}_h) = (\mathcal{F}^{L,n+1}(\boldsymbol{\lambda}_h), \boldsymbol{\gamma}_h) \quad \forall \boldsymbol{\gamma}_h, \tag{5.18}$$

where superscript L is used to denote that all variables, including domain integrals, are evaluated considering zero convection velocity. The convection step consists in finding $\boldsymbol{u}_h^{n+1}$ and $\boldsymbol{\alpha}_h^{n+1}$ such that

$$\frac{1}{\delta t} m_s^{n+1} \left(\boldsymbol{u}_h^{n+1} - \boldsymbol{u}_h^{L,n+1}, \boldsymbol{v}_h \right) + c_s^{n+1}(\boldsymbol{c}_h; \boldsymbol{u}_h, \boldsymbol{v}_h) = 0 \quad \forall \boldsymbol{v}_h, \tag{5.19}$$

$$\frac{1}{\delta t} (\boldsymbol{\alpha}_h^{n+1} - \boldsymbol{\alpha}_h^{L,n+1}, \boldsymbol{\gamma}_h) + (\boldsymbol{c}_h^{L,n+1} \cdot \nabla \boldsymbol{\alpha}_h^{n+1}, \boldsymbol{\gamma}_h) = 0 \quad \forall \boldsymbol{\gamma}_h. \tag{5.20}$$

In order to take the convective term linear, the convection velocity in this step may be taken as

$$\boldsymbol{c}_h^{L,n+1} = \boldsymbol{u}_h^{L,n+1} - \boldsymbol{u}_{\text{dom}}^{n+1}.$$

Remark 3 Note that when $\mathcal{F}$ is an algebraic operator, (5.18) is in fact an approximation to an ordinary differential equation, which corresponds to the time integration, usually at each numerical integration point, of the evolution equation for the internal variables. Obviously, options better than the simplest backward Euler scheme of (5.18) could be used. On the other hand, (5.20) simply represents the transport of the internal variables from the material configuration to the final configuration at t^{n+1}. There are models in which also the stresses $\boldsymbol{\sigma}_h$ need to be transported. Since these stresses are discontinuous across the element edges for C^0 shape functions, solving equation (5.16) for $\boldsymbol{\sigma}$ is not straightforward. There are a certain number of strategies to deal with this problem which can be found for example in [75, 117, 4]. In the numerical examples of Section 5.6 there is no need to update the stresses, since only elastic materials have been considered. $\triangle$

5.3 The FM-ALE method applied to solid mechanics

In this section we describe how the Fixed-Mesh ALE method can be applied to problems in solid mechanics. An overview of the FM-ALE approach is presented but major attention is given to the particular characteristics of its application to solids. For a more detailed explanation of the FM-ALE method in the general framework of moving domains, see Chapter 4. In this section and the ones that follow it, the numerical schemes will be particularized for $\theta = 1$.

5.3.1 The general algorithm

Suppose Ω^0 is meshed with a finite element mesh M^0 and that at time level t^n the domain $\Omega(t^n)$ is meshed with a finite element mesh M^n. Let $\boldsymbol{u}_h^n$ be the velocity already computed on $\Omega(t^n)$. The purpose is to obtain the region the solid occupies at time t^{n+1}, $\Omega(t^{n+1})$, and to compute the various unknown fields. If the classical ALE method is used, M^n would deform to another mesh defined at t^{n+1}. In the FM-ALE approach we do not use this mesh to compute the unknowns of the problem, but instead we re-mesh in such a way that the new mesh is, essentially, M^0 once again. The main steps of the algorithm have been presented in Chapter 4. Here we present an alternative algorithm which would lead to a very similar result. A global idea of the meshes involved in the process is represented in Fig. 4.1.

1. Define $\Gamma_{\text{free}}^{n+1}$ by updating the function that defines it.
2. Deform the mesh M^n to M_{ALE}^{n+1} using the classical ALE concepts and compute the mesh velocity $\boldsymbol{u}_{\text{dom}}^{n+1}$.
3. Write down the ALE solid mechanics equations on M_{ALE}^{n+1}.
4. Solve the equations on M_{ALE}^{n+1} to compute the unknowns in the deformed mesh.
5. *Split the elements* of M^0 cut by $\Gamma_{\text{free}}^{n+1}$ to define a mesh on $\Omega(t^{n+1})$, M^{n+1}.
6. *Project* the results from M_{ALE}^{n+1} to M^{n+1}.

The conceptual idea of the algorithm in Chapter 4 and the one presented here is basically the same, the only difference being that in the first algorithm the equations are solved on M^{n+1} while in the second algorithm they are solved on M_{ALE}^{n+1}. However, the second approach is more convenient if non-linear systems of equations are to be solved. This is due to the fact that the projection from M_{ALE}^{n+1} to M^{n+1} is done only at the end of the time step, while in the first algorithm this projection has to be carried out at each iteration.

5.3.2 Details on some of the steps

Tracking of Γ_{free}

In the examples presented in Section 5.6 the body surface has been tracked by means of a Lagrangian boundary mesh. The intersection between the finite element mesh and the Lagrangian mesh is found at each time step. After the ALE solid equations have been solved,

the Lagrangian contour mesh is deformed. The transmission of information between the two meshes is done by means of an L^2 projection. There are two possible ways of updating the position of the Lagrangian mesh nodes. The first approach consists in computing:

$$\int_\Gamma \boldsymbol{v}_h \cdot \delta \boldsymbol{d}_L^{n:n+1} = \int_\Gamma \boldsymbol{v}_h \cdot \delta \boldsymbol{d}_{FE}^{n:n+1},$$
$$\boldsymbol{d}_L^{n+1} = \boldsymbol{d}_L^n + \delta \boldsymbol{d}_L^{n:n+1}, \tag{5.21}$$

while in the second approach we compute:

$$\int_\Gamma \boldsymbol{v}_h \cdot \boldsymbol{d}_L^{n+1} = \int_\Gamma \boldsymbol{v}_h \cdot \boldsymbol{d}_{FE}^{n+1}, \tag{5.22}$$

where $\boldsymbol{d}_{FE}$ are the displacements computed on the finite element mesh and $\boldsymbol{d}_L$ are the displacements of the Lagrangian surface mesh, $\delta \boldsymbol{d}^{n:n+1}$ are the incremental displacements from time step n to time step $n+1$ and $\boldsymbol{v}_h$ are now the test functions corresponding to the nodes of the Lagrangian surface mesh.

Although (5.21) could seem a natural choice, since a usual approach in solid mechanics is to solve for the incremental nodal displacements, (5.22) works better than (5.21). This is a consequence of the fact that (5.22) preserves the information of the undeformed geometry, while the incremental approach of (5.21) leads to the loss of this information.

Another possible approach which has not been exploited in this work would be to track the body surface by means of a level set function. For more details on the use of level set functions in the FM-ALE method, see [41]. A method to track initial position of the particles has been developed in [47], which could also be applied to the present formulation.

Approximate imposition of boundary conditions

As done in Chapter 4, the strategy described in Chapter 2 is used in order to impose Dirichlet boundary conditions without changing neither the connectivity nor the sparsity of the final system of equations matrix. There are a number of other methods for imposing Dirichlet boundary conditions on fixed meshes which could have been used, see for example the strategies proposed in the *Immersed Boundary Method* [113], the *Fictitious Domain Method* [62, 63], and the *hybrid Cartesian/immersed boundary methods* [60, 139, 104]. The only difficulties in the imposition of boundary conditions near boundaries with irregular geometry are associated to the fact that we consider the interface between the "inside the domain" region and the "outside the domain" region of each cut element to be a single line segment. This introduces limitations in the tracking of the solid body geometry, specially when sharp corners are present or, in the floating solids problems to be described later, in those elements which are cut by the solid body boundary and the level set function at the same time. This is a common limitation of fixed mesh methods which can be addressed by coding more complex subelement integration subroutines, although this has not been done in the current work.

Splitting of elements

Mesh M^{n+1} is obtained by splitting the elements of M^0 cut by $\Gamma_{\text{free}}^{n+1}$. Meshes M^{n+1} and M^0 only differ in the subelements created after the splitting just mentioned. Mesh M^{n+1} could be

thought as a local refinement of mesh M^0 to make it conform the boundary $\Gamma_{\text{free}}^{n+1}$. As in other fixed grid methods, this computational complication can be avoided by prescribing boundary conditions on $\Gamma_{\text{free}}^{n+1}$ in an approximate way, although the local refinement from M^0 to M^{n+1} is needed also to perform the numerical integration of the different terms appearing in (5.14)-(5.20).

However, depending on how $\Gamma_{\text{free}}^{n+1}$ intersects M^0, the resulting subelements size could be very small compared to the size of elements adjacent to $\Gamma_{\text{free}}^{n+1}$. This results in an ill-conditioning of the system of equations to be solved. In order to avoid this issue we work with a slightly deformed mesh $M_{\text{ALE,def}}^{n+1}$ at each time step constructed as follows: exterior nodes very close to $\Gamma_{\text{free}}^{n+1}$ (closer than $0.1h$ for example) are displaced in a direction orthogonal to $\Gamma_{\text{free}}^{n+1}$ until they match exactly the body surface. The splitting of this mesh will avoid ill-conditioned elements.

Remark 6 Note that since only nodes very close to $\Gamma_{\text{free}}^{n+1}$ are displaced, the stretch of the elements is negligible, as it can be seen in Fig. 5.1. △

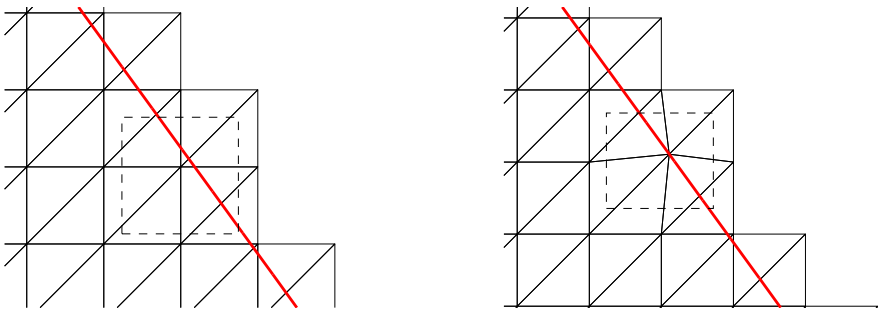


Figure 5.1: Deformation of M^0 for splitting purposes. Left: undeformed mesh. Right: deformed mesh. Element stretch is barely appreciable.

5.4 The FM-ALE method applied to Fluid-Structure Interaction problems

In Chapter 4 the FM-ALE method for solving flow problems in moving domains was presented. In this chapter we have seen how the FM-ALE approach can be used to solve problems in solid mechanics. In this section we will show how to solve Fluid-Structure Interaction problems using the FM-ALE approach for both the fluid and the structure. In this case the same background fixed mesh can be used to solve both the first and the second case, leading to some advantageous features in the coupling between them.

5.4.1 The FM-ALE method for flow problems in moving domains

In this section an incompressible Newtonian fluid will be considered. As in the solid case, the domain movement has to be taken into account. The incompressible Navier - Stokes equations are:

Find a velocity $\boldsymbol{u} : \Omega(t) \times (0,T) \longrightarrow \mathbb{R}^d$ and a pressure $p : \Omega(t) \times (0,T) \longrightarrow \mathbb{R}$ such that

$$\rho \left[\left. \frac{\partial \boldsymbol{u}}{\partial t} \right|_{\boldsymbol{x}'} + (\boldsymbol{u} - \boldsymbol{u}_{\text{dom}}) \cdot \nabla \boldsymbol{u} \right] - \nabla \cdot (2\mu \nabla^S \boldsymbol{u}) + \nabla p = \rho \boldsymbol{f}, \tag{5.23}$$

$$\nabla \cdot \boldsymbol{u} = 0, \tag{5.24}$$

where $\nabla^S \boldsymbol{u}$ is the symmetrical part of the velocity gradient, ρ is the fluid density, μ is the viscosity and $\boldsymbol{f}$ is the vector of body forces. Initial and boundary conditions have to be appended to problem (5.23)-(5.24).

If finite elements are used, the fully discrete stabilized counterpart of this equations is:

Find $\boldsymbol{u}_h^{n+1}$ and p_h^{n+1} such that

$$\begin{aligned} & m_1^{n+\theta} \left(\delta_t \boldsymbol{u}_h^{n+1} \big|_{\boldsymbol{x}^n}, \boldsymbol{v}_h \right) + a^{n+\theta}(\boldsymbol{u}_h, \boldsymbol{v}_h) \\ & \quad + c^{n+\theta}(\boldsymbol{u}_h - \boldsymbol{u}_{\text{dom}}; \boldsymbol{u}_h, \boldsymbol{v}_h) + b_1^{n+\theta}(p_h, \boldsymbol{v}_h) = l_1^{n+\theta}(\boldsymbol{v}_h), \end{aligned} \tag{5.25}$$

$$m_2^{n+\theta} \left(q_h, \delta_t \boldsymbol{u}_h^{n+1} \big|_{\boldsymbol{x}^n} \right) + b_2^{n+\theta}(q_h, \boldsymbol{u}_h) + s^{n+\theta}(q_h, p_h) = l_2^{n+\theta}(q_h), \tag{5.26}$$

for all test functions $\boldsymbol{v}_h$ and q_h, the former vanishing on the Dirichlet part of the boundary Γ_D. The different forms appearing in these have been defined in Chapter 4.

5.4.2 Solving the coupled problem

When dealing with the coupled problem, the domain is divided into a solid part $\Omega_s(t)$ and a fluid part $\Omega_f(t)$, where $\bar{\Omega}^0 = \bar{\Omega}_s(t) \cup \bar{\Omega}_f(t)$ and $\Omega_s(t) \cap \Omega_f(t) = \emptyset$. The boundary of the coupled problem can now be divided into the Dirichlet boundary for the fluid Γ_D^f and the solid Γ_D^s, the Neumann boundary for the fluid Γ_N^f and the solid Γ_N^s, and the common interface boundary between the fluid and the solid Γ_{free}. The boundary of the coupled problem is now $\Gamma = \Gamma_D \cup \Gamma_N \cup \Gamma_{\text{free}}$, where $\Gamma_D = \Gamma_D^f \cup \Gamma_D^s$ and $\Gamma_N = \Gamma_N^f \cup \Gamma_N^s$.

The problem now consists in solving (5.14) or (5.17)-(5.20) in $\Omega_s(t)$ and (5.25)-(5.26) in $\Omega_f(t)$. The key point is obviously the boundary conditions to be applied. On Γ_D and Γ_N boundary conditions are the usual applied to solid and fluid mechanics problems:

$$\begin{aligned} \boldsymbol{u}^s &= \bar{\boldsymbol{u}}^s \quad \text{on } \Gamma_D^s, \\ \boldsymbol{u}^f &= \bar{\boldsymbol{u}}^f \quad \text{on } \Gamma_D^f, \\ \boldsymbol{n} \cdot \boldsymbol{\sigma}^s &= \bar{\boldsymbol{t}}^s \quad \text{on } \Gamma_N^s, \\ \boldsymbol{n} \cdot \boldsymbol{\sigma}^f &= \bar{\boldsymbol{t}}^f \quad \text{on } \Gamma_N^f, \end{aligned}$$

where superscript s has been introduced for the unknowns in the solid and superscript f for the unknowns in the fluid. In Γ_{free} conditions must be applied such that velocity and traction continuity at all time steps is fulfilled:

$$\begin{aligned} \boldsymbol{u}^s &= \boldsymbol{u}^f \quad \text{on } \Gamma_{\text{free}}, \\ \boldsymbol{n} \cdot \boldsymbol{\sigma}^s &= \boldsymbol{n} \cdot \boldsymbol{\sigma}^f \quad \text{on } \Gamma_{\text{free}}. \end{aligned}$$

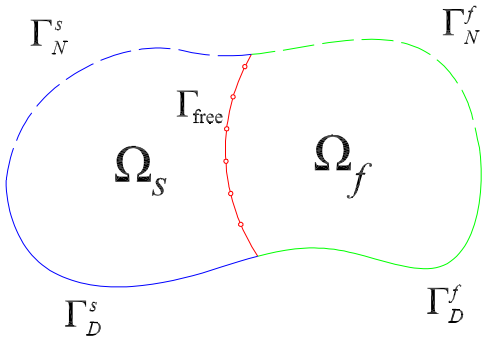


Figure 5.2: Domain and domain boundary subdivision in FSI problems. Blue: Γ^s. Green: Γ^f. Red: Γ_{free}. Dashed line: Γ_N. Continuous line: Γ_D

Satisfying the kinematic continuity leads to mass conservation, whereas satisfying the dynamic continuity yields conservation of linear momentum.

Note that since $\bar{\Omega}^0 = \bar{\Omega}_s(t) \cup \bar{\Omega}_f(t)$ and M^0 is a mesh covering Ω^0, it is possible at each time step to divide M^0 into M^t_s and M^t_f such that:

$$M^0 = M^t_s \cup M^t_f,$$

where M^t_s and M^t_f are meshes covering $\Omega_s(t)$ and $\Omega_f(t)$ respectively, and not necessarily disjoint. *This allows us to use a single mesh* M^0 to solve both the fluid and the solid mechanics problems for $t \in [0, T]$. However, if boundary conditions are prescribed in an approximate way, for example following the strategy proposed in Chapter 2, there will be some nodes of M^0 which will belong to both M^t_s and M^t_f. At these nodes degrees of freedom need to be duplicated so that unknowns for both the fluid and the solid can be obtained.

There are basically two ways of dealing with the coupled Fluid-Structure Interaction problem: the *partitioned* and the *monolithic* approaches. In partitioned methods the solid and fluid problems are solved independently and coupling between both is achieved iteratively by means of the so called *coupling algorithms*. The major advantage of this approach is that specific codes can be used for each of the two problems to be solved. Its drawback is that convergence is difficult to achieve under certain circumstances. In the monolithic approach both problems are solved simultaneously and coupling between them is imposed in an *implicit* manner, which avoids the need of coupling iterations. The dimension of the system to be solved is larger in the monolithic case. However, if iterations within each time step yield convergence of the partitioned solution to the monolithic one, the distinction between both is blurred. In fact, those iterations can be understood as a certain preconditioner to solve iteratively the monolithic problem.

Although both strategies can be used together with the FM-ALE method, the monolithic approach is the one which suits it best. We have already seen how M^0 can be divided into M^t_s and M^t_f. Moreover, with the formulation we use to solve the incompressible Navier - Stokes equations, it is possible to use the same interpolation functions for the unknowns corresponding to the solid problem and for the ones corresponding to the flow problem. It is very easy in this case to implicitly write the coupling conditions between fluid and structure. To this purpose equations corresponding to fluid velocity unknowns in nodes belonging to L^f_{-1} (L^s_0)

are used to prescribe implicitly $\boldsymbol{u}^s = \boldsymbol{u}^f$. On the other hand, traction continuity is imposed in equations corresponding to solid velocity/displacement in L_0 and L_{-1} simply by adding the corresponding boundary terms to the momentum conservation equations.

The final system to be solved is: find $\boldsymbol{u}_h^{n+1,s}$, $\boldsymbol{u}_h^{n+1,f}$ and p_h^{n+1} such that

$$\begin{aligned} m_1^{n+\theta}\left(\delta_t \boldsymbol{u}_h^f, \boldsymbol{v}_h^f\right) + a^{n+\theta}(\boldsymbol{u}_h^f, \boldsymbol{v}_h^f) + c^{n+\theta}(\boldsymbol{u}_h^f - \boldsymbol{u}_{\mathrm{dom}}; \boldsymbol{u}_h^f, \boldsymbol{v}_h^f) + b_1^{n+\theta}(p_h, \boldsymbol{v}_h^f) &= l_1^{n+\theta}(\boldsymbol{v}_h^f), \\ m_2^{n+\theta}\left(q_h, \delta_t \boldsymbol{u}_h^f\right) + b_2^{n+\theta}(q_h, \boldsymbol{u}_h^f) + s^{n+\theta}(q_h, p_h) &= l_2^{n+\theta}(q_h), \\ m_s^{n+\theta}\left(\delta_t \boldsymbol{u}_h^s, \boldsymbol{v}_h^s\right) + a_s^{n+\theta}(\boldsymbol{\lambda}_h, \boldsymbol{v}_h^s) + c_s^{n+\theta}(\boldsymbol{u}_h^s - \boldsymbol{u}_{\mathrm{dom}}; \boldsymbol{u}_h^s, \boldsymbol{v}_h^s) &= l_s^{n+\theta}(\boldsymbol{v}_h^s), \end{aligned} \tag{5.27}$$

for all test functions $\boldsymbol{v}_h^f$ and $\boldsymbol{v}_h^s$ vanishing on the Dirichlet part of the boundary Γ_D, and all test functions q_h. Obviously, integrals corresponding to forms defined on the fluid region are extended over $\Omega_f(t)$, whereas integrals corresponding to forms associated to the solid are extended over $\Omega_s(t)$.

Another point we want to stress is that if one wants to solve the Fluid-Structure Interaction problem using a monolithic scheme, but the solid is to be solved using a fractional step method, the strategy to follow is simply to solve the *material* phase of the solid coupled monolithically with the fluid problem. Once this phase is solved variables of interest in $\Omega_s(t)$ can be transported in the *convective* phase (only for the solid mechanics problem).

In the Fluid-Structure Interaction example in Section 5.6 the monolithic approach has been used. However, there is no major drawback in using the FM-ALE approach altogether with partitioned schemes.

Let us close this section summarizing the final algorithm for the FM-ALE method applied to FSI problems, which is:

1. Define $\Gamma_{\mathrm{free}}^{n+1}$ by updating the function that defines it.

2. Deform the mesh M^n to M_{ALE}^{n+1} using the classical ALE concepts and compute the mesh velocity $\boldsymbol{u}_{\mathrm{dom}}^{n+1}$.

3. Write down the ALE solid (M_s^t) and fluid (M_f^t) mechanics equations on M_{ALE}^{n+1}. If a fractional step method is used for the solid equations this corresponds to the material phase

4. Solve the equations on M_{ALE}^{n+1} to compute the unknowns in the deformed mesh.

5. If a fractional step method is used for the solid, solve the convective phase

6. *Split the elements* of M^0 cut by $\Gamma_{\mathrm{free}}^{n+1}$ to define a mesh on $\Omega(t^{n+1})$, M^{n+1}.

7. *Project* the results from M_{ALE}^{n+1} to M^{n+1}.

5.5 The FM-ALE method applied to Fluid-Structure Interaction problems involving a free surface

In the previous sections we have seen how to deal with fluid-structure interaction problems using the FM-ALE method. In this section we introduce the extra ingredient of the free surface of the fluid, which requires some care when the solid body boundary is close to the function representing the free surface. In this case we consider the solid body to be rigid, and thus very few degrees of freedom are needed to describe the solid body movement.

5.5.1 Problem statement

Let us consider a region $\Omega^0 \subset \mathbb{R}^d$ ($d = 2, 3$) where a flow will take place during a time interval $[0, T]$. However, we consider the case in which the fluid at time t occupies only a subdomain $\Omega(t) \subset \Omega^0$ (note in particular that $\Omega(0) \subset \Omega^0$). Suppose also that the boundary of $\Omega(t)$ is defined by part of $\partial\Omega^0$ and a moving boundary that we call $\Gamma_{\mathrm{f}}(t) = \partial\Omega(t) \setminus \partial\Omega^0 \cap \partial\Omega(t)$. This moving part of $\partial\Omega(t)$ may correspond to the boundary of a moving solid immersed in the fluid or can be determined by a level set function. The setting of the problem is described in Fig. 5.3, where also the solid domain $\Omega^{\mathrm{s}}(t)$ and the fluid structure interface Γ_{sf} have been depicted.

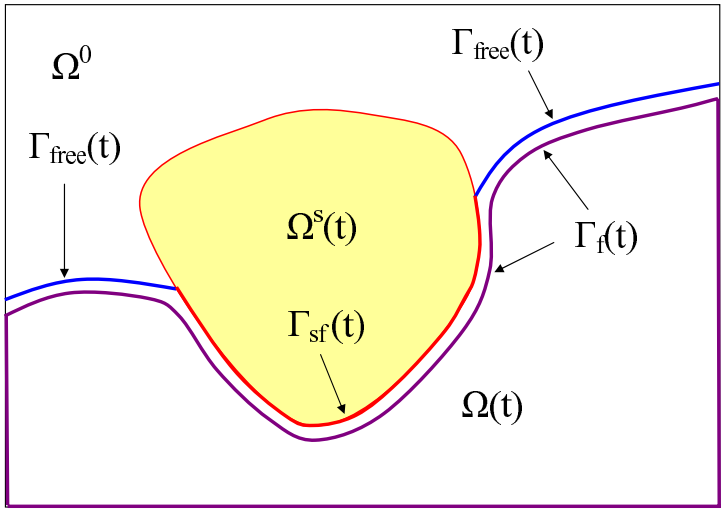


Figure 5.3: Setting

Boundary conditions are of the form

$$\begin{aligned} \boldsymbol{u} &= \bar{\boldsymbol{u}} \quad \text{on } \Gamma_D, \\ \boldsymbol{n} \cdot \boldsymbol{\sigma} &= \bar{\boldsymbol{t}} \quad \text{on } \Gamma_N, \end{aligned}$$

where $\boldsymbol{n}$ is the external normal to the boundary, $\boldsymbol{\sigma} = -p\boldsymbol{I} + 2\mu\nabla^S\boldsymbol{u}$ is the Cauchy stress tensor and $\bar{\boldsymbol{u}}$ and $\bar{\boldsymbol{t}}$ are the given boundary data.

When dealing with the fluid part of the domain, we also have to take into account the movement of the free surface. This movement is dealt with by means of a level set function, as explained for example in [41, 37].

In the level set method we define a smooth function Ψ over Ω^0 which allows us to determine $\Omega(t)$. In our case we define $\Omega(t)$ as the region over which $\Psi(\boldsymbol{x},t)$ is positive. The position of the fluid front will be defined by the iso-value contour $\Psi = 0$. The evolution of this level set function is computed by means of the transport equation: find $\Psi : \Omega^0 \times (0,T) \longrightarrow \mathbb{R}$ such that:

$$\frac{\partial \Psi}{\partial t} + \boldsymbol{u} \cdot \nabla \Psi = 0, \tag{5.28}$$

with the additional requirement that the advection velocity at the free surface coincides with that of the fluid. The procedure used to compute the level set advection velocity is described in subsection 5.5.2.

For the solid part, we consider only the case of rigid bodies. The solid also evolves in time, and we denote its domain by $\Omega_s(t)$. As usual in solid mechanics problems, we face the problem in a purely Lagrangian way. Let us denote by $\boldsymbol{x}_{\mathrm{rb}}$ the position vector of the center of mass of the rigid body, and by $\boldsymbol{\theta}_{\mathrm{rb}}$ the Euler angles. The motion equations for the rigid body are: find $\boldsymbol{x}_{\mathrm{rb}} : (0,T) \longrightarrow \mathbb{R}^d$ and $\boldsymbol{\theta}_{\mathrm{rb}} : (0,T) \longrightarrow \mathbb{R}^d$ such that

$$m \frac{\mathrm{d}^2 \boldsymbol{x}_{\mathrm{rb}}}{\mathrm{d}t^2} = \boldsymbol{F}, \tag{5.29}$$

$$\boldsymbol{I} \frac{\mathrm{d}^2 \boldsymbol{\theta}_{\mathrm{rb}}}{\mathrm{d}t^2} = \boldsymbol{T}, \tag{5.30}$$

where m is the mass of the rigid body, $\boldsymbol{I}$ is the inertia tensor, $\boldsymbol{F}$ is the force vector at the center of mass and $\boldsymbol{T}$ is the torque at the center of mass.

Initial and boundary conditions have to be appended to problem (5.23)-(5.24) and initial conditions to (5.29)-(5.30). In order to impose these conditions we redefine $\Gamma_{\mathrm{f}}(t)$ as $\Gamma_{\mathrm{f}}(t) = \Gamma_{\mathrm{free}}(t) \cup \Gamma_{\mathrm{sf}}(t)$, where $\Gamma_{\mathrm{free}}(t)$ is the part of $\Gamma_{\mathrm{f}}(t)$ corresponding to the free surface and $\Gamma_{\mathrm{sf}}(t)$ is the part of $\Gamma_{\mathrm{f}}(t)$ corresponding to the interaction between the fluid and the structure.

In Γ_{free} boundary conditions are of Neumann type, specifically we prescribe tractions to zero, neglecting surface tension. In Γ_{sf} we must impose the usual conditions in fluid-structure interaction problems, which for rigid bodies are continuity of the velocity field and transmission of the forces and torques exerted on the solid body by the fluid.

On the rest of the boundary of $\Omega(t)$ the usual Dirichlet and Neumann boundary conditions can be considered.

5.5.2 Numerical treatment

The numerical treatment of the incompressible Navier-Stokes equations is done as described in the previous sections, and a similar formulation is used in order to deal with the advection of the level set function.

For the temporal integration of the solid mechanics problem we consider Newmark's method:

$$\begin{aligned} m\ddot{\boldsymbol{x}}_{\mathrm{rb}}^{n+1} &= \boldsymbol{F}^{n+1}, \\ \ddot{\boldsymbol{x}}_{\mathrm{rb}}^{n+1} &= \frac{1}{\beta\delta t^2}[\boldsymbol{x}_{\mathrm{rb}}^{n+1} - \boldsymbol{x}_{\mathrm{rb}}^{n} - \dot{\boldsymbol{x}}_{\mathrm{rb}}^{n}\delta t] - \left(\frac{1}{2\beta} - 1\right)\ddot{\boldsymbol{x}}_{\mathrm{rb}}^{n}, \\ \dot{\boldsymbol{x}}_{\mathrm{rb}}^{n+1} &= \frac{\gamma}{\beta\delta t}[\boldsymbol{x}_{\mathrm{rb}}^{n+1} - \boldsymbol{x}_{\mathrm{rb}}^{n}] + \left(1 - \frac{1}{2\beta}\right)\delta t\ddot{\boldsymbol{x}}_{\mathrm{rb}}^{n}, \end{aligned} \tag{5.31}$$

and

$$\begin{aligned} \boldsymbol{I}\ddot{\boldsymbol{\theta}}_{\mathrm{rb}}^{n+1} &= \boldsymbol{T}^{n+1}, \\ \ddot{\boldsymbol{\theta}}_{\mathrm{rb}}^{n+1} &= \frac{1}{\beta\delta t^2}[\boldsymbol{\theta}_{\mathrm{rb}}^{n+1} - \boldsymbol{\theta}_{\mathrm{rb}}^{n} - \dot{\boldsymbol{\theta}}_{\mathrm{rb}}^{n}\delta t] - \left(\frac{1}{2\beta} - 1\right)\ddot{\boldsymbol{\theta}}_{\mathrm{rb}}^{n}, \\ \dot{\boldsymbol{\theta}}_{\mathrm{rb}}^{n+1} &= \frac{\gamma}{\beta\delta t}[\boldsymbol{\theta}_{\mathrm{rb}}^{n+1} - \boldsymbol{\theta}_{\mathrm{rb}}^{n}] + \left(1 - \frac{1}{2\beta}\right)\delta t\ddot{\boldsymbol{\theta}}_{\mathrm{rb}}^{n}, \end{aligned} \tag{5.32}$$

where β and γ are parameters to be chosen. Most usual values are $\beta = 1/4$ and $\gamma = 1/2$, which provide a second order stable and non dissipative scheme.

Tracking of Γ_{f}

As explained before, the free surface is tracked by means of a level set function. However, there still remain some points to be clarified about how this process is exactly carried out. The main particularity of our problem is that the fluid boundary Γ_{f} is represented not only by Γ_{free} but also by Γ_{sf} (see Fig. 5.3). Theoretically, if the advection velocity of Ψ is that of the rigid body in Γ_{sf}, $\Gamma_{\mathrm{free}} \cap \Gamma_{\mathrm{sf}} = \Gamma_{\mathrm{sf}}$, but in practice both boundaries will rarely exactly coincide. A strategy has to be devised to deal with this lack of coincidence of the functions which define the boundary of the fluid domain, which is due to numerical approximation errors.

The first situation we consider is the one depicted in Fig. 5.4. As we can see, Γ_{sf} does not coincide with the free surface Γ_{free}, understood as the isovalue $\Psi = 0$ of the level set function. This problem can be solved in the following manner: let us define $\Gamma_{\mathrm{free}}^{*}$ as the part of Γ_{free} interior to $\Omega_{\mathrm{s}}(t)$. Now we can define the fluid boundary as:

$$\Gamma_{\mathrm{f}} = (\Gamma_{\mathrm{free}} \setminus \Gamma_{\mathrm{free}}^{*}) \cup \Gamma_{\mathrm{sf}}$$

The second and more delicate problem occurs when Γ_{free} gets *delayed* with respect to Γ_{sf} due to the extra numerical diffusion which appears in the advection of the level set. This situation is outlined in Fig. 5.5. As we are considering continuum mechanics, the only way the water surface can separate from the solid is slipping. In order to avoid an incorrect separation process of the fluid from the solid we rely on the velocity in the non-computed *air domain*.

In the *air domain* we do not solve the Navier-Stokes equations. This is the reason why we have to compute an artificial velocity to advect the level set function. To do this we solve a

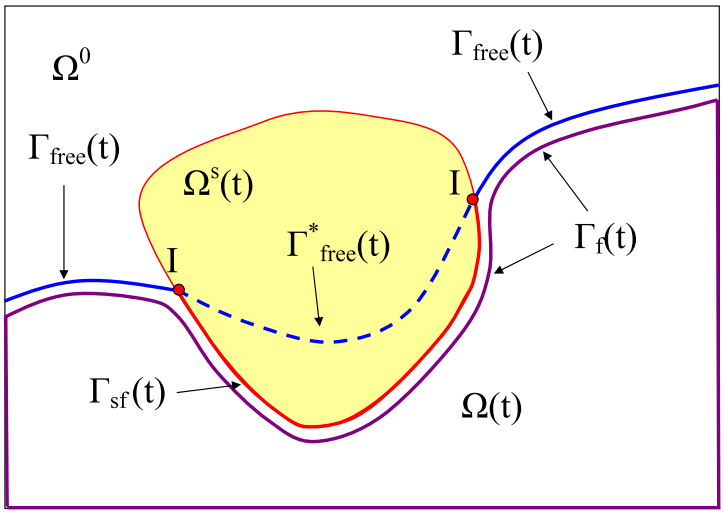


Figure 5.4: Lack of coincidence of the boundary defined by Γ_{free} and Γ_{sf}

modified Stokes problem with the following particularities: find a velocity $\boldsymbol{u} : \Omega^0 \setminus \Omega(t) \longrightarrow \mathbb{R}^d$ and a pressure $p : \Omega^0 \setminus \Omega(t) \longrightarrow \mathbb{R}$ such that

$$-\nabla \cdot (2\nu \nabla^S \boldsymbol{u}) + \nabla p = \boldsymbol{f}, \tag{5.33}$$

$$\nabla \cdot \boldsymbol{u} = -\alpha, \tag{5.34}$$

where α is a constant which is positive in the solid domain, and zero outside of it. Note that using the positive constant α in the interior of the solid body does not introduce any extra numerical error since (5.34) refers only to the computation of the *artificial* velocity. Slip boundary conditions are applied except for Γ_{free} where the fluid velocity is imposed. The positive constant α, which has units of $[T^{-1}]$, makes the solid body act as a sink, which allows us to avoid any numerically induced *delay* in the advection of the interface. Again, the subgrid-scale method is used to stabilize the problem and allow for equal velocity-pressure interpolation. This problem needs to be solved only in a region close to Γ_{f}. In the rest of $\Omega^0 \setminus \Omega(t)$ the advection velocity can be more straightforwardly computed, for example by means of a linear extrapolation.

As usual when using the level set method, we need to reinitialize the level set function every certain number of time steps. This reinitialization may move the position of the interface. In order to avoid this a special procedure is used in the nodes of cut elements. It is a slight variation of the method presented in [66], which consists of the following: Suppose that we are given the free surface configuration in Fig. 5.6. Now we divide the nodes belonging to the elements cut by the free surface in two sets, each set corresponding to one side of the free surface. In the first wet side, we prescribe the nodal values of the level set function to be equal to the (signed) distance between the node and the free surface.

We use the degrees of freedom of the nodes in the second side of the free surface to prescribe the reinitialized level set function to be zero valued on the free surface, by using the approximate imposition of boundary conditions of the previous section. On the rest of the nodes we prescribe the level set function to be the signed distance from the nodes to the free surface, although we could also use the more efficient procedure in [66], or other techniques as

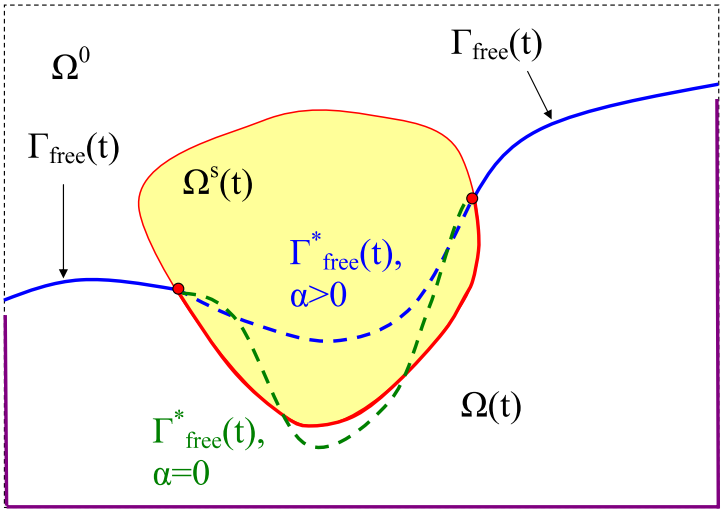


Figure 5.5: Γ^*_{free} for different α parameters

the one described in [37]. This algorithmic procedure allows the reinitialized level set function to very accurately track the free surface, that is, the free surface for the reinitialized level set function minimizes the distance between the interface position before and after the reinitialization. This guarantees that no significative mass loss is introduced during the reinitialization of the level set function, since the error in the reinitialization is of $\mathcal{O}(\delta t^2)$.

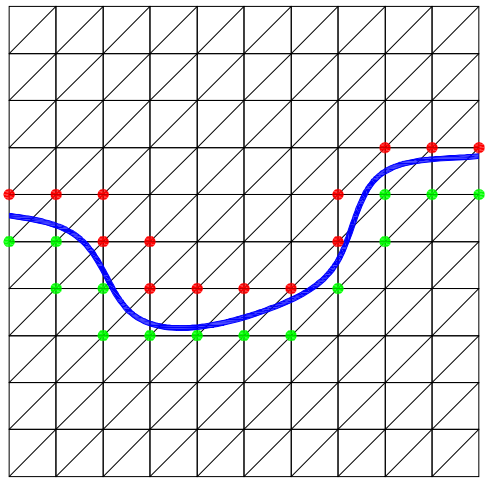


Figure 5.6: Green nodes: nodes in which we prescribe the level set function to be equal to the signed distance from the nodes to the free surface. Red nodes: nodes whose nodal values are such that the level set function is zero on the free surface (Approximate imposition of boundary conditions).

5.6 Numerical examples

In this section we present some numerical examples which illustrate the behavior of the methodology proposed in this work.

5.6.1 An example on FM-ALE applied to solid mechanics

In this example a cantilever subject to gravity forces will be simulated by means of the FM-ALE method. Since a fractional step method is used, the equations to be solved are (5.17)-(5.19) (with the additional terms coming from the stabilization). A Neo-Hookean material has been considered, which takes into account large strains. The constitutive equation of this material is (see [17]):

$$\boldsymbol{\sigma} = \frac{1}{J}[\lambda_0 \ln J \boldsymbol{I} + \mu_0(\boldsymbol{B} - \boldsymbol{I})], \tag{5.35}$$

where $\boldsymbol{B} = \boldsymbol{F} \cdot \boldsymbol{F}^T$, λ_0 and μ_0 are material parameters and $\boldsymbol{I}$ is the identity tensor.

The hold-all domain is the rectangle $B = [-1, 5] \times [0, 11]$. A background mesh of 3200 linear triangles has been used. The considered solid is a rectangular cantilever situated at $[0, 1] \times [0, 10]$. The material parameters are $\lambda_0 = 2000$ and $\mu_0 = 5000$. The solid density is $\rho = 1$, which has been considered to remain constant through the whole process.

In this case the body is only under the effect of (horizontal) body forces, given by $\boldsymbol{b}^T = (1, 0)$. Dirichlet boundary conditions are applied at $y = 0$, where displacements in any direction are prescribed to zero. On the rest of the solid boundary, Neumann boundary conditions are applied:

$$\boldsymbol{n} \cdot \boldsymbol{\sigma} = \boldsymbol{0}.$$

Initial conditions correspond to the undeformed static configuration.

The time step size has been set to $\delta t = 0.2$ and $\theta = 1$ has been taken (first order scheme in time).

Fig. 5.7 shows the mesh used to solve this problem. The boundary of the body does not match the boundary of the mesh. Fig. 5.8 shows horizontal and vertical displacements at time step 90.

In order to validate the FM-ALE method, we have compared the results obtained with our approach with those obtained if a classical Updated - Lagrangian method for boundary fitting meshes is used. To this end we have used a boundary fitting mesh with the same element density to solve the same problem. Fig. 5.9 shows the horizontal displacement of a material point placed at the top of the cantilever, whose coordinates are $\boldsymbol{X}^T = (0.5, 10)$.

The simulation is carried out during 90 time steps. As it can be seen results are very similar to the ones obtained in the classical Updated - Lagrangian approach.

Another issue which we were interested in is the effect of the use of a fractional step method. In Fig. 5.10 we have plotted the results obtained if we take $\boldsymbol{u}_{\text{dom}} = \boldsymbol{u}_h$ (the convective phase is avoided) versus the results obtained if we take $\boldsymbol{u}_{\text{dom}} \neq \boldsymbol{u}_h$ (a convective phase is needed). As we can see no difference can be appreciated between results, and we can conclude that the error introduced by the use of a fractional step method is small.

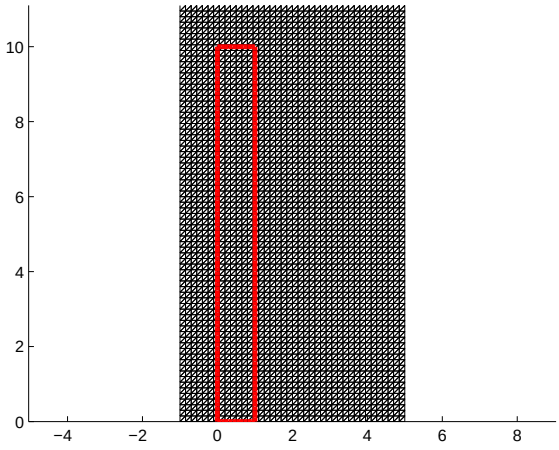


Figure 5.7: Immersed mesh use to solve the solid mechanics example and body surface

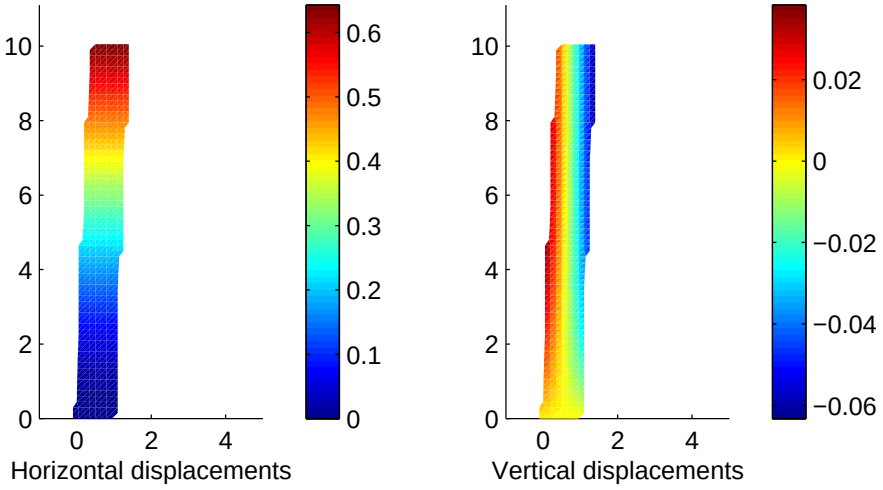


Figure 5.8: Displacements after 90 time steps

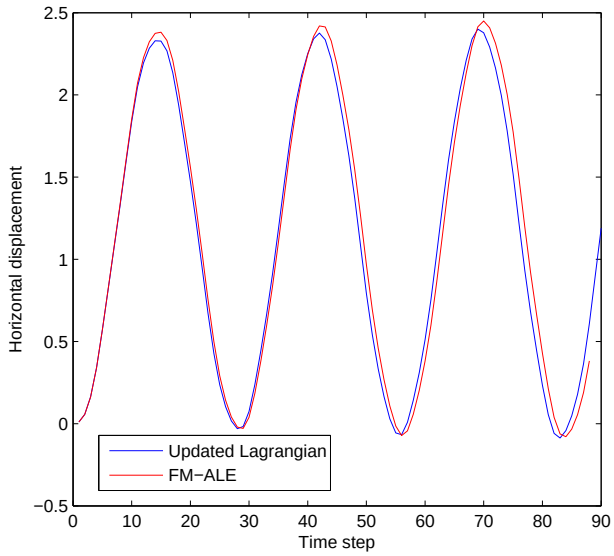


Figure 5.9: Horizontal displacement at a point placed at the top of the cantilever. Comparison between FM-ALE and Updated - Lagrangian formulations

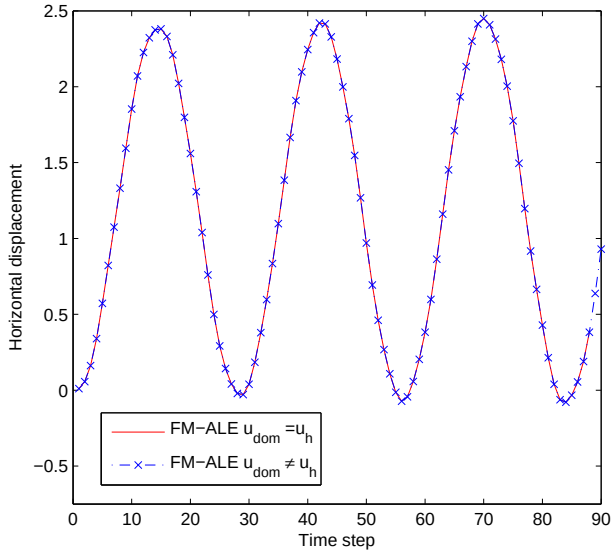


Figure 5.10: Horizontal displacement at a point placed at the top of the cantilever. Comparison between FM-ALE $\boldsymbol{u}_{\mathrm{dom}} = \boldsymbol{u}_h$ and FM-ALE $\boldsymbol{u}_{\mathrm{dom}} \neq \boldsymbol{u}_h$ formulations

5.6.2 Examples of the FM-ALE applied to Fluid-Structure Interaction problems

In the first example the same cantilever as in subsection 5.6.1 is simulated. However, the forces acting on the cantilever are due to the interaction with a fluid in this case. The hold-all domain is the rectangle $B = [-10, 70] \times [0, 20]$. An unstructured background mesh of 4655 linear triangles is used. This mesh is much coarser than the one used in the previous example if we consider the element density in the solid body area.

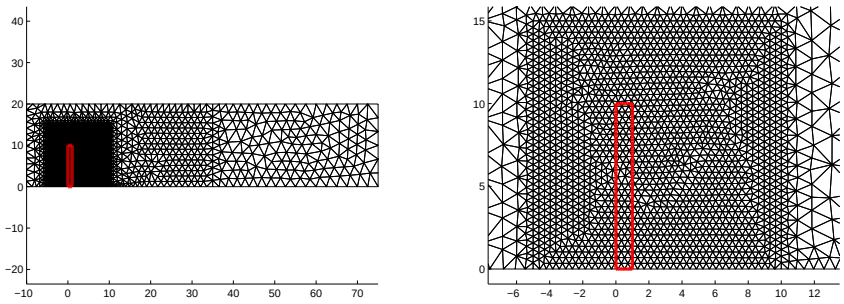


Figure 5.11: Mesh used to solve the first Fluid-Structure Interaction example. Left: full mesh. Right: detail of the area surrounding the solid body

The material parameters for the solid body are the ones used in the previous example. For the fluid we have considered $\rho = 2$ and $\mu = 0.2$. The velocity at $x = 0$ is prescribed to (1,0), whereas at $y = 0$ and $y = 20$ the $y-$velocity component is prescribed to 0 and the x-component is left free. The outflow (where both the x- and y-components are free) is $x = 70$. The Reynolds number is 100, based on the cantilever height and the prescribed inflow velocity. The time step size has been set to $\delta t = 1$ and $\theta = 1$.

A monolithic approach has been used to couple fluid and structure. Both the fluid and the structure have been solved using the FM-ALE method with the *same background mesh* (see Fig. 5.11). The deformed configuration of the beam at time step 100 is shown in Fig. 5.12.

In Fig. 5.13 the horizontal displacement at a point placed at the top of the cantilever is plotted. This figure shows how the movement of the cantilever is dumped by the action of the fluid. After a certain number of time steps the movement becomes stationary. Fig. 5.14 and Fig. 5.15 show the cantilever displacements and the fluid velocities and pressures at time step 100.

In the second example we consider a thin elastic non-linear beam (Neo-Hookean material) attached to a fixed square rigid body, which are submerged in an incompressible fluid flow. Vortices separating from the corners of the rigid body generate oscillating forces on the beam. Geometry is given in Fig. 5.16, while Fig. 5.17 shows the mesh used to solve the problem. Again, a higher element density has been used in the region which will be occupied by the solid. The mesh is as coarse as possible, with the requirement that there are at least three elements to cover the beam width, and it is composed of 9388 triangular elements and 4812 nodes. The setting of the problem is similar to that proposed in [135], although we have considered a thicker beam in order to be able to use the rather coarse mesh described. It is clear that thin structures are not the most favorable situation for fixed mesh methods, and in particular for

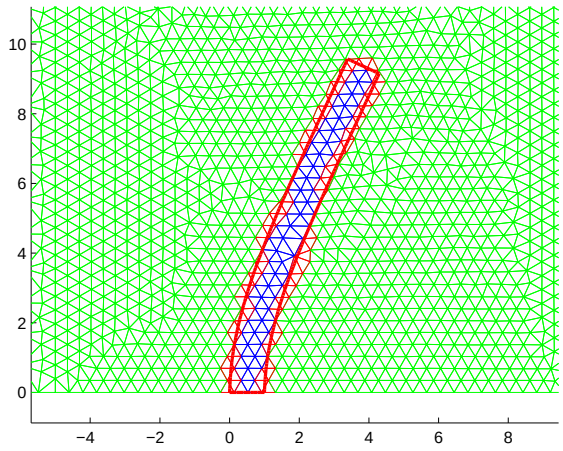


Figure 5.12: The same mesh is used to solve both the fluid and the structure. Green: elements in which only degrees of freedom corresponding to the fluid have to be solved. Blue: elements in which only degrees of freedom corresponding to the structure have to be solved. Red: Elements in which degrees of freedom corresponding to both fluid and structure have to be solved.

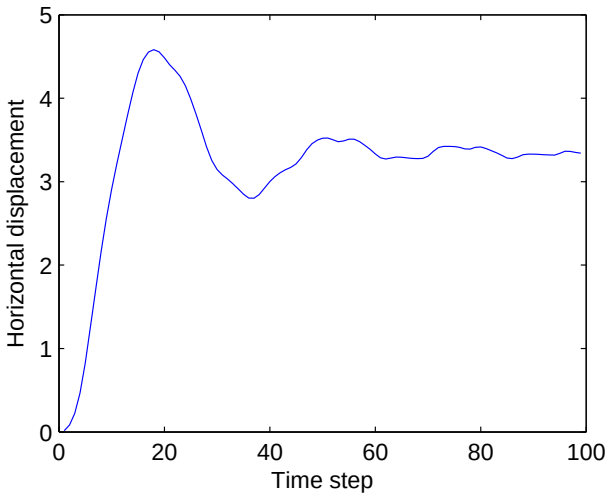


Figure 5.13: Horizontal displacement at a point placed at the top of the cantilever.

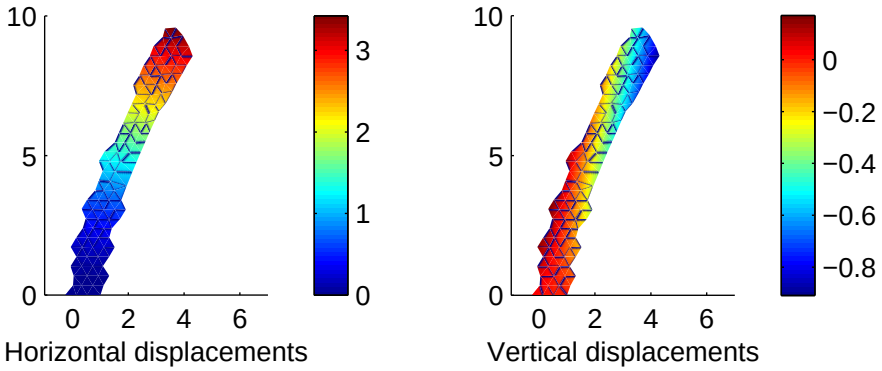


Figure 5.14: Solution for the solid body at t = 100

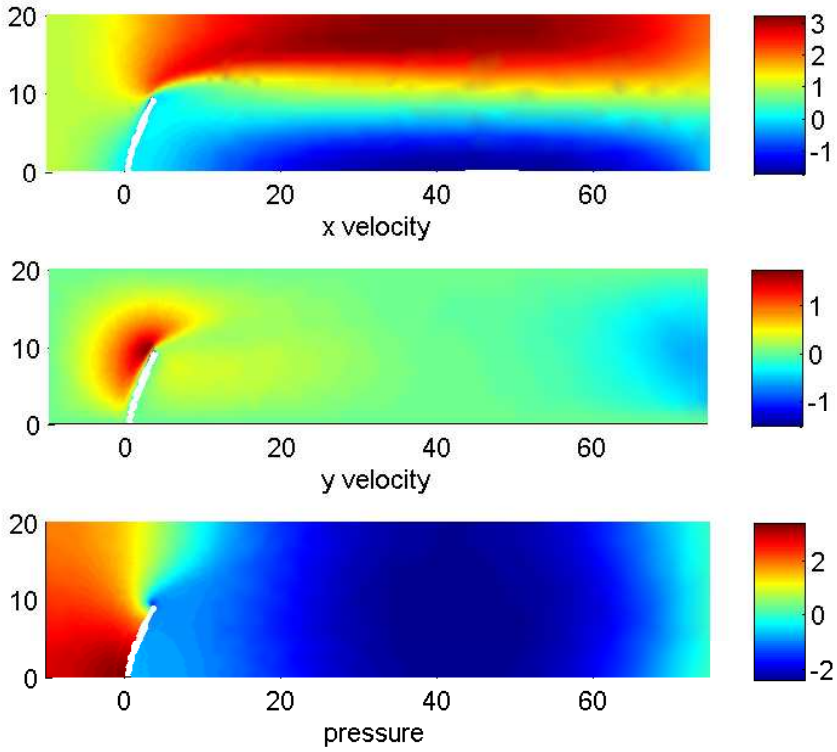


Figure 5.15: Solution for the fluid at t = 100

FM-ALE (an alternative would be to represent these structures by a zero width solid, which is not a situation considered in this work).

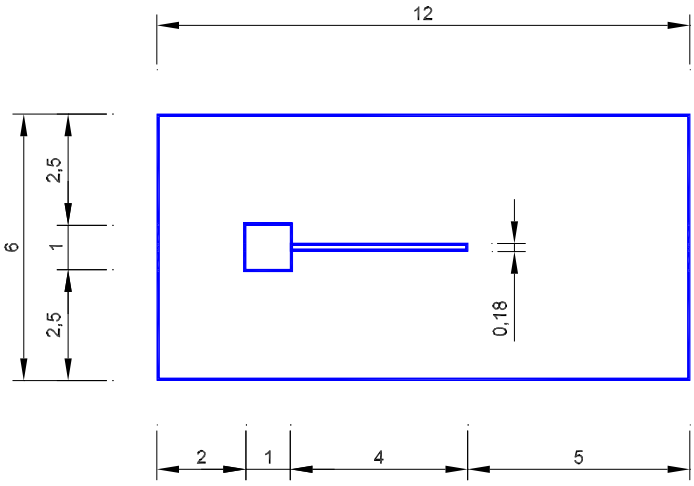


Figure 5.16: Geometry

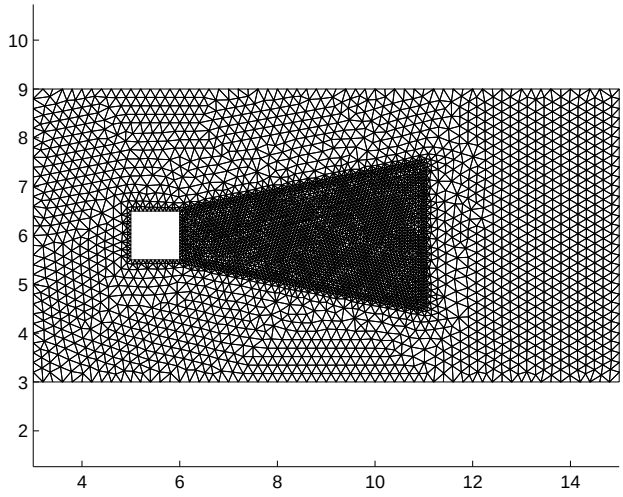


Figure 5.17: Mesh used to solve the second Fluid-Structure Interaction example

The fluid material properties are $\rho = 1 \times 10^{-2}$ and $\mu = 1.7 \times 10^{-3}$. The solid material properties are $\rho = 2$, $\lambda_0 = 1.72 \times 10^6$ and $\mu_0 = 7.4 \times 10^5$. The horizontal inflow velocity at $x = 0$ is set to 40, yielding a Reynolds number of $\mathrm{Re} = 235$ referred to the length of the square rigid body. Slip boundary conditions are set at the walls of the channel. The beam and the square rigid body are assigned non-slip boundary conditions. The time step is set to $\delta t = 0.002$ and $\theta = 1$. A monolithic approach has been used to solve the FSI problem, although the fractional step scheme has been used to deal with the solid. Both the fluid and the structure have been solved using the fixed background mesh.

Contours of pressure and velocity components and $t = 5$ are shown in Fig. 5.18, when the vortex shedding behind the square cylinder has appeared but is not yet fully developed. It can be observed that even in this transient stage results are smooth and boundary conditions on the elastic beam perfectly accounted for. The evolution of the vertical displacement at the edge of the beam is plotted in Fig. 5.19, where it can be observed that the dynamics of the system are fully developed at about $t = 7$. Then, a perfectly harmonic flow pattern sets in, with a single frequency in the time response, as it can be observed from Fig. 5.20.

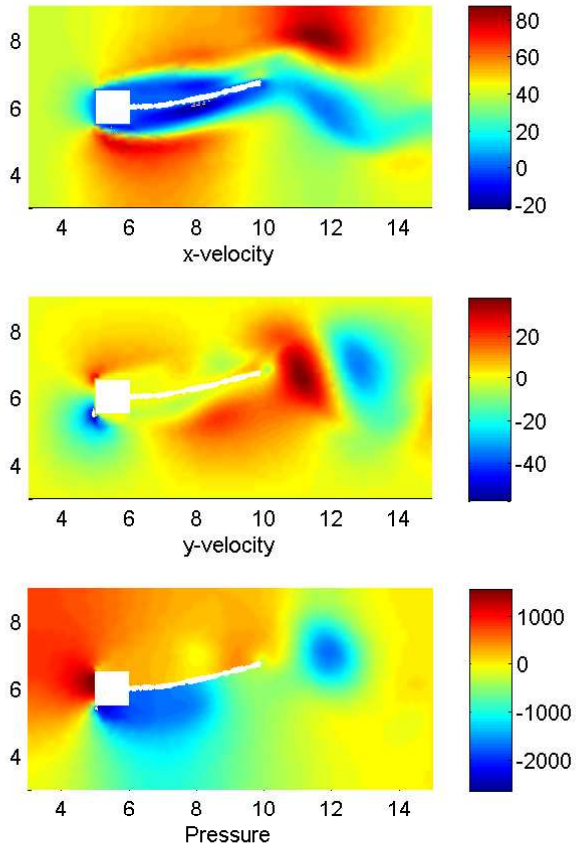


Figure 5.18: Velocity and pressure at time $t = 5.0$

Finally we present a third example in which we compare the results obtained with the FM-ALE method in the benchmark problem proposed in [129]. Again, we consider a thin elastic beam attached to a rigid body. In this case, the material for the elastic beam is a Saint Venant - Kirchhoff material, and the rigid body is a circle. The constitutive equation for a Saint Venant-Kirchhoff material is (see [17]):

$$\boldsymbol{\sigma} = \frac{1}{J}\boldsymbol{F}[\lambda tr(\boldsymbol{E})\boldsymbol{I} + 2\mu\boldsymbol{E}]\boldsymbol{F}^{\boldsymbol{t}}, \tag{5.36}$$

where $\boldsymbol{E} = \frac{1}{2}(\boldsymbol{B} - \boldsymbol{I})$. Both solid bodies are immersed in an incompressible fluid flow. Geometry is given in Fig. 5.21, Fig. 5.22 shows the 10883 triangle mesh used to solve the problem.

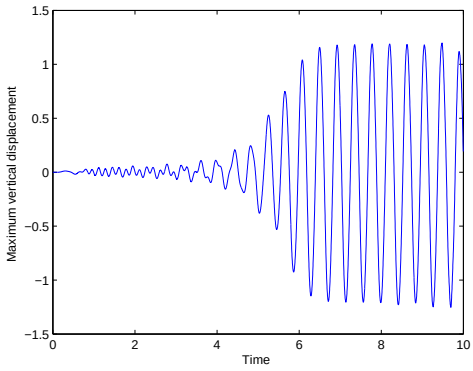


Figure 5.19: Vertical displacement at the edge of the beam

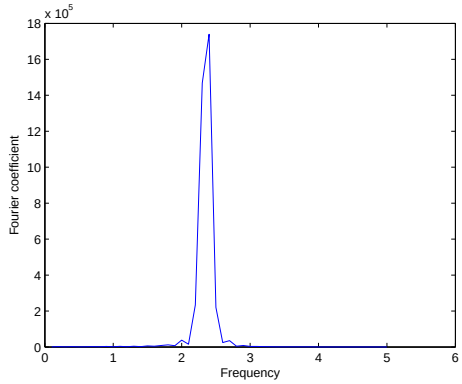


Figure 5.20: Fourier transform of the maximum vertical displacement. The frequency of the main vibration mode is 2.4

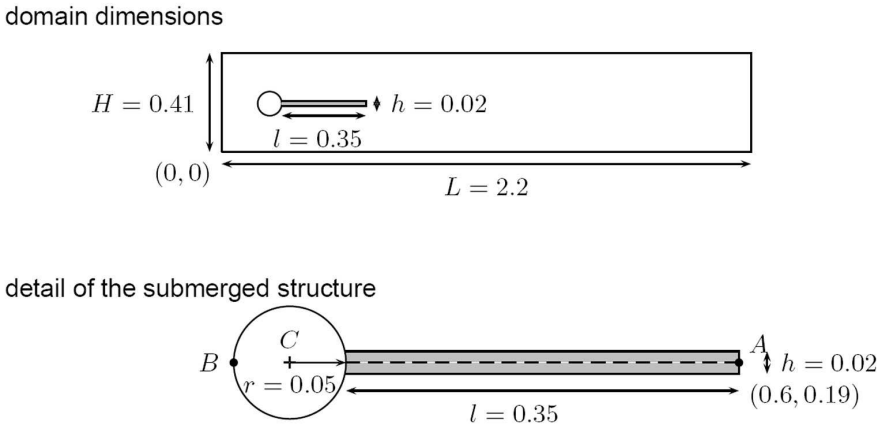


Figure 5.21: Geometry, benchmark [129]

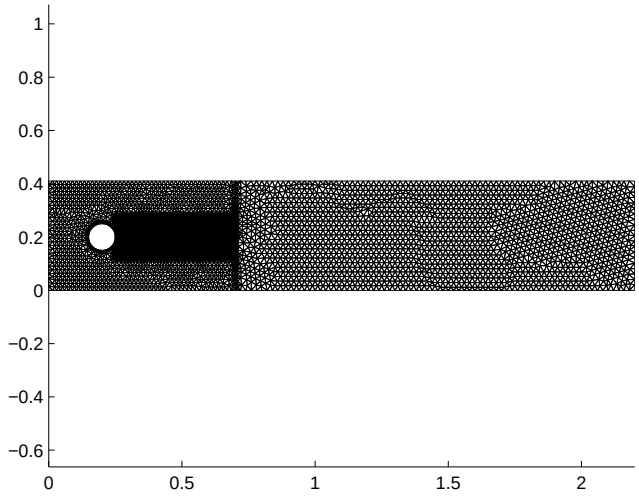


Figure 5.22: Mesh used to solve FSI benchmark [129]

The fluid material properties are $\rho = 1\times10^{3}$ and $\nu = 1\times10^{-3}$. The solid material properties are $\rho = 1 \times 10^{3}$, $\nu = 0.4$ and $\mu = 2.0 \times 10^{6}$. The horizontal inflow velocity at $x = 0$ is set to a parabolic profile with mean value 2, yielding a Reynolds number of $\mathrm{Re} = 200$ referred to the size the circular rigid body. Non-Slip boundary conditions are set at the walls of the channel. The beam and the square rigid body are also assigned non-slip boundary conditions. The time step is set to $\delta t = 0.005$ and a second order backward difference scheme is adopted. A monolithic approach has been used to solve the FSI problem, although the fractional step scheme has been used to deal with the solid. Both the fluid and the structure have been solved using the fixed background mesh.

Contours of pressure and velocity components and $t = 6.8$ are shown in Fig. 5.23, and we can see that again smooth velocity and pressure fields are obtained in the fluid. Finally, we compare the results obtained with the FM-ALE method with the ones presented in [129] in Fig. 5.24. It can be observed that a very good agreement is obtained in both the period (≈ 0.18) and amplitude of the tip displacement ($\approx 0.002 \pm 0.035$) in the oscillations of the elastic beam.

5.6.3 Examples of the FM-ALE method applied to FSI problems involving a free surface

In this section we present two numerical examples which illustrate the behavior of the methodology proposed in this work.

The first example we propose consists of two rigid bodies falling into an incompressible fluid. To run this example we use the FM-ALE method on the fluid part, and we track the free surface by means of a level set function. The initial configuration of the problem can be seen in Fig. 5.25.

The hold-all domain is the rectangle $B = [0, 2.4] \times [0, 1]$. A background mesh of 7968 linear triangles has been used. The fluid density is $\rho = 1$, and the viscosity is set to $\mu = 0.001$. For the solid bodies, density is $\rho = 0.75$.

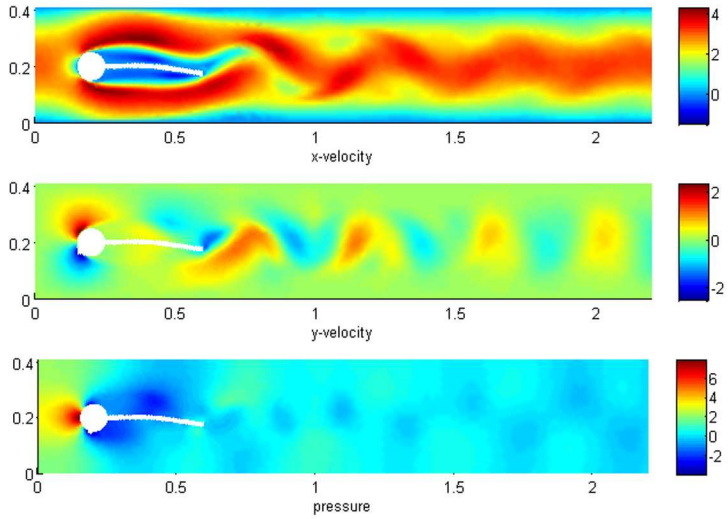


Figure 5.23: Velocity and pressure at time $t = 6.8$

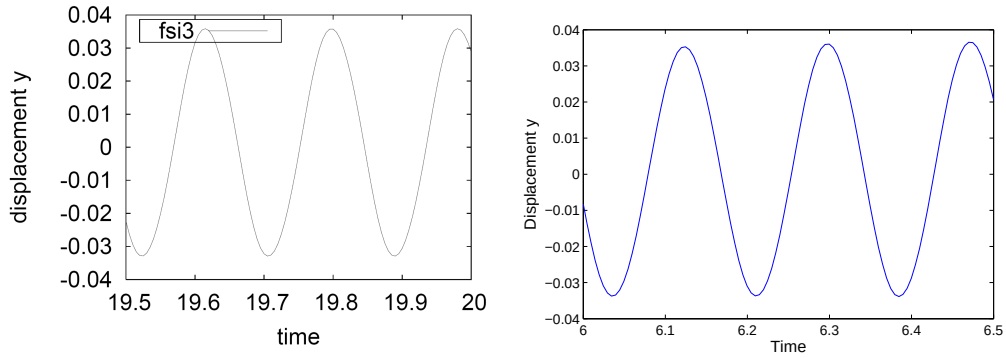


Figure 5.24: Vertical displacement at the edge of the beam. Comparison between benchmark results in [129] (left) and results obtained with the FM-ALE method (right).

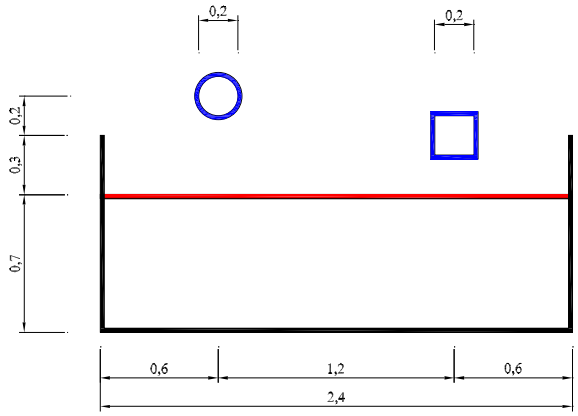


Figure 5.25: Initial configuration, example 1

Both the fluid and the structure are subject to a vertical gravity force of value $g = -10$. We apply slip boundary conditions both at the interface between the fluid and the deposit wall and at the interface between the solid bodies and the fluid. This means that only the velocity in the direction normal to the interface has to coincide between the fluid and the solid bodies. In the free surface Γ_{free} tractions in the normal direction are prescribed to zero.

The time step has been set to $\delta t = 0.02$ and 150 time steps have been carried out. Regarding the advection of the level set function, the α parameter for the artificial mass sink explained in subsection 5.5.2 is taken as $\alpha = 2$.

Figures 5.26 to 5.29 show the results for various time steps. Let us remark that the solution obtained is smooth along all the computation, even for the first critical steps in which the rigid bodies *contact* the free surface. The irregular boundaries are due to the fact that for ease of post processing we have plotted the solution in the elements cut by the boundary without taking into account that the boundary of the domain does not fit the boundary of the elements.

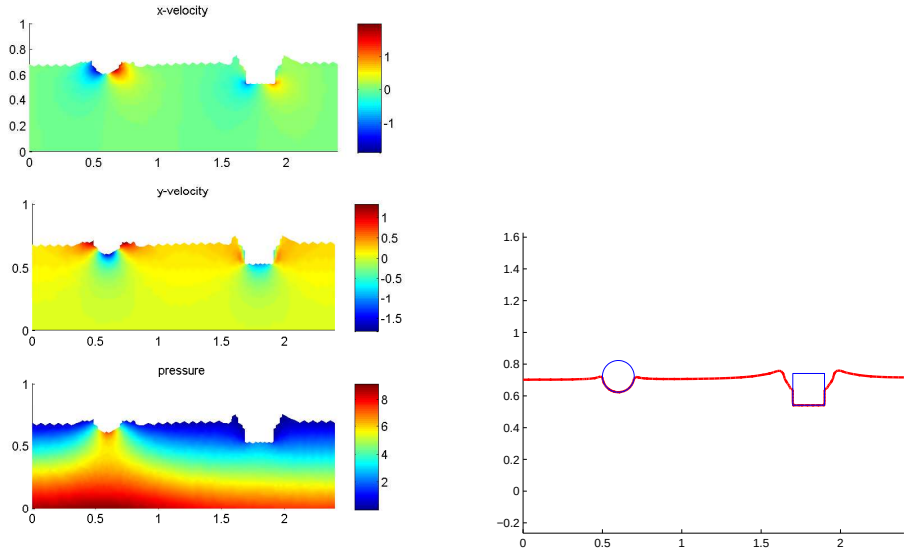


Figure 5.26: Unknown fields and free surface at t = 0.36

The kind of computation involved in this example, in which the fluid region undergoes very large deformations, would have implied the need for continuous remeshing if classical ALE methods were used. Our fixed mesh strategy avoids it by projecting the results to the background mesh at each time step.

The most critical situations in this problem, which are the instant in which *the fluid closes around the rigid body*, and surrounds it completely, and the instant when *the solid body breaks the free surface*, are handled in a very natural way with the level set function strategy.

In the second example, we simulate an oval body falling into an incompressible fluid. Again, we use the FM-ALE method to simulate the fluid part, and we track the free surface with a level set function. The initial configuration for this problem can be seen in Fig. 5.30.

The hold-all domain is the rectangle $B = [0, 1] \times [0, 1]$. The fluid density is $\rho = 1$, and the viscosity is set to $\mu = 0.01$. For the solid body, the density is $\rho = 0.5$. Both the fluid and the structure are subject to a vertical gravity force of value $g = -10$. The time step has been set

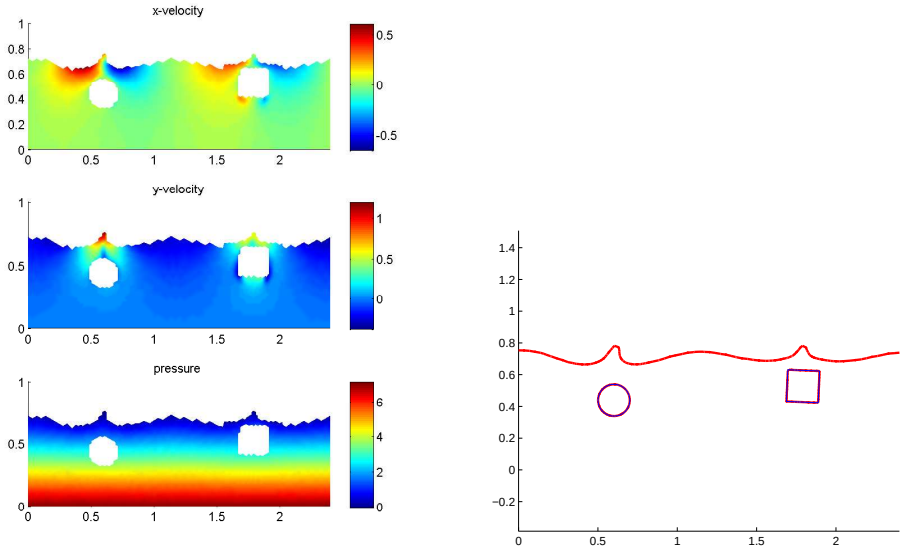


Figure 5.27: Unknown fields and free surface at t = 0.78

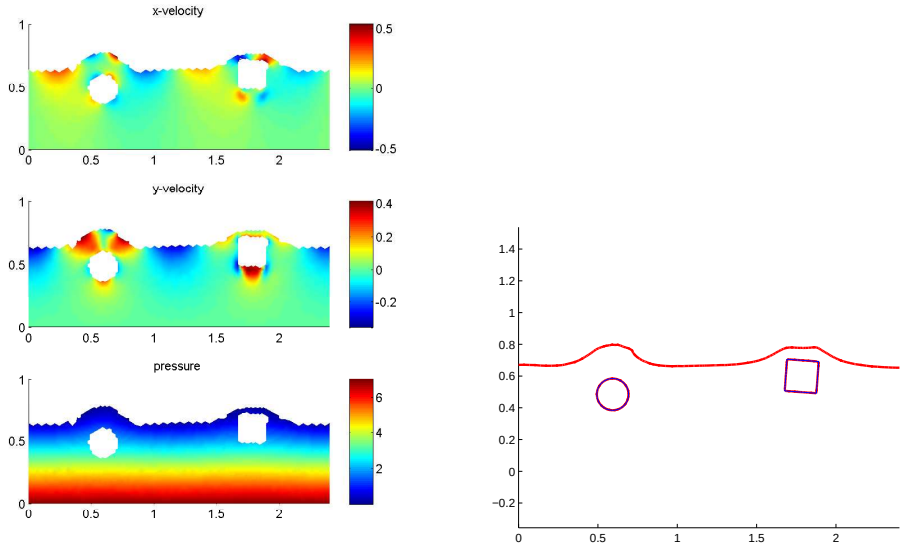


Figure 5.28: Unknown fields and free surface at t = 0.98

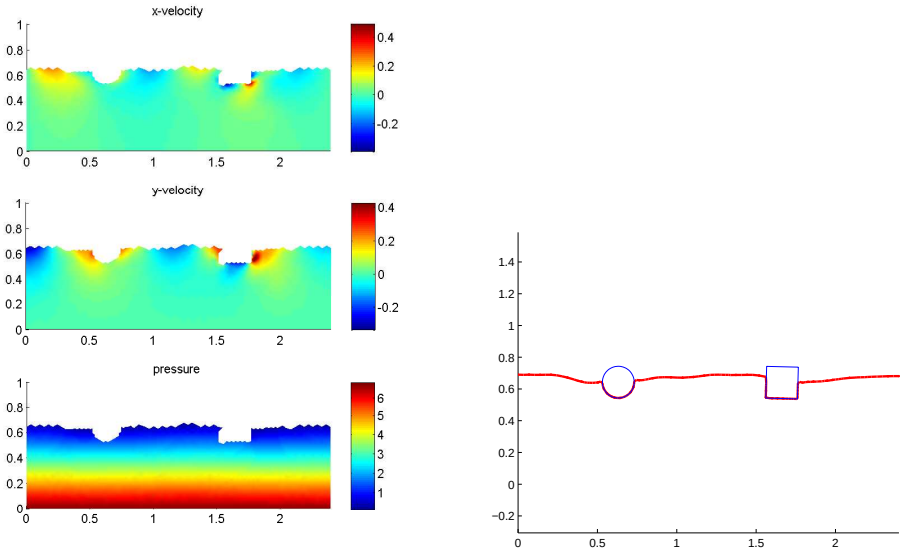


Figure 5.29: Unknown fields and free surface at t = 2.40

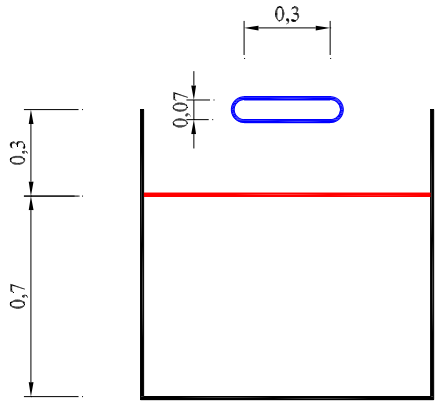


Figure 5.30: Initial configuration, example 2

to $\delta t = 0.02$ and 500 time steps have been carried out. Again, the parameter α has been set to $\alpha = 2$.

In this case we have used three different meshes in order to compare the behavior of the method with different element sizes. In the first case, we have used a relatively coarse triangle mesh with 1890 nodes. In the second case we have used a finer mesh, with 5205 nodes and the last mesh consisted of 11614 nodes. We compare the vertical displacement of the center of mass of the solid body in the three cases in Fig. 5.31. We can see that we obtain a solution close to the converged one for the vertical displacement with the two finer meshes. We have also represented vertical and horizontal velocities for the solid body in Fig. 5.32. Horizontal velocity for the solid body should be zero due to the problem symmetry. The numerical errors introduced in the geometry interpolation of the cut elements (the considered meshes are not symmetric) are the cause for horizontal velocity to appear, although the horizontal velocity is small compared to the vertical velocity and the domain size. In any case, as already explained this local error could be removed by improving the numerical integration, which can be done by introducing appropriate subelements for integration purposes.

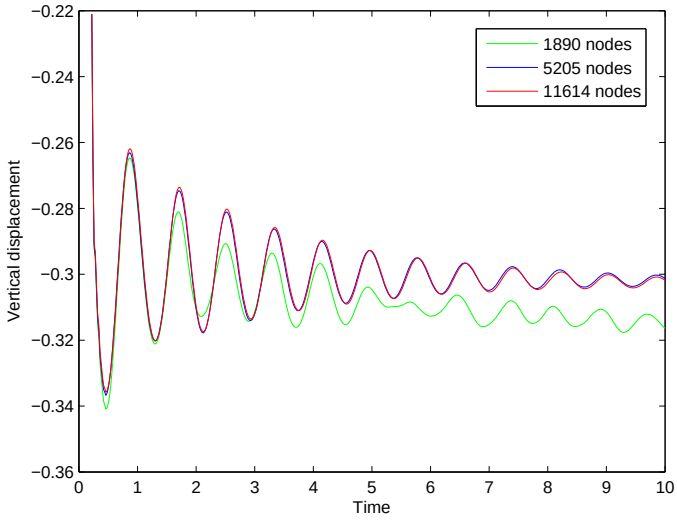


Figure 5.31: Time evolution of the vertical displacement

Fluid velocity, free surface and solid body configurations can be seen in Fig. 5.33 to Fig. 5.35. As expected, in the final configuration, when the body is at rest, half of the body is inside the fluid domain and half of it is in the air domain, since $\rho_s/\rho_f = 0.5$. Again, the lack of symmetry in the velocity fields in Fig. 5.35, when both the fluid and the solid are close to rest, is due to the cumulative numerical errors of the geometry interpolation in the cut elements.

Fig. 5.36 shows the time evolution of the total fluid mass. We can see that mass loss is larger in the coarse mesh case, and much smaller for the finer meshes although no method for correcting mass loss has been used.

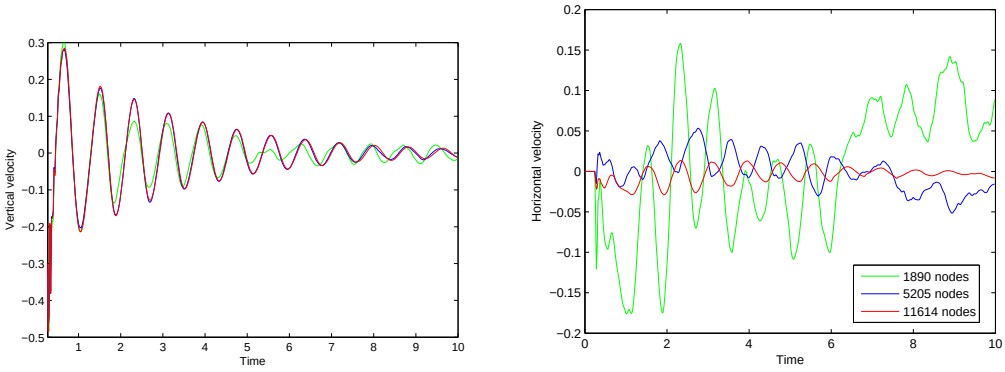


Figure 5.32: Time evolution of the vertical and horizontal velocities

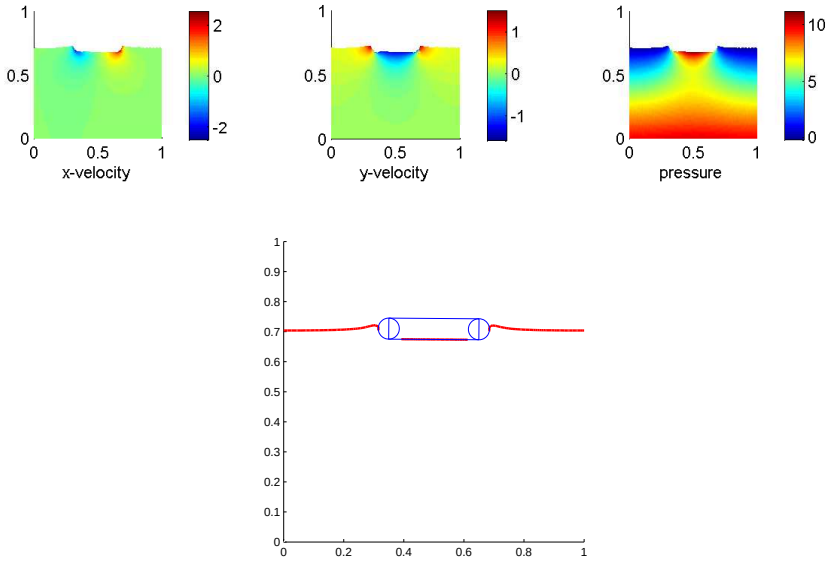


Figure 5.33: Unknown fields and free surface at t = 0.28

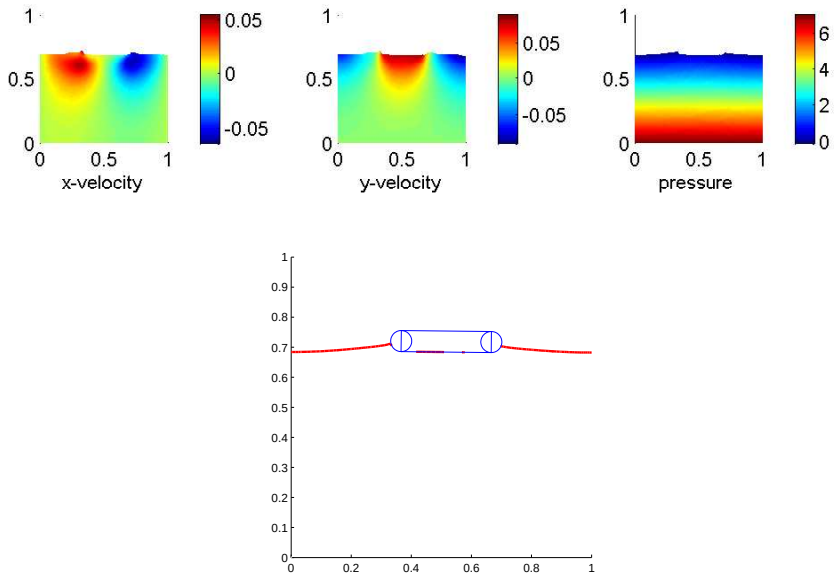


Figure 5.34: Unknown fields and free surface at t = 2.5

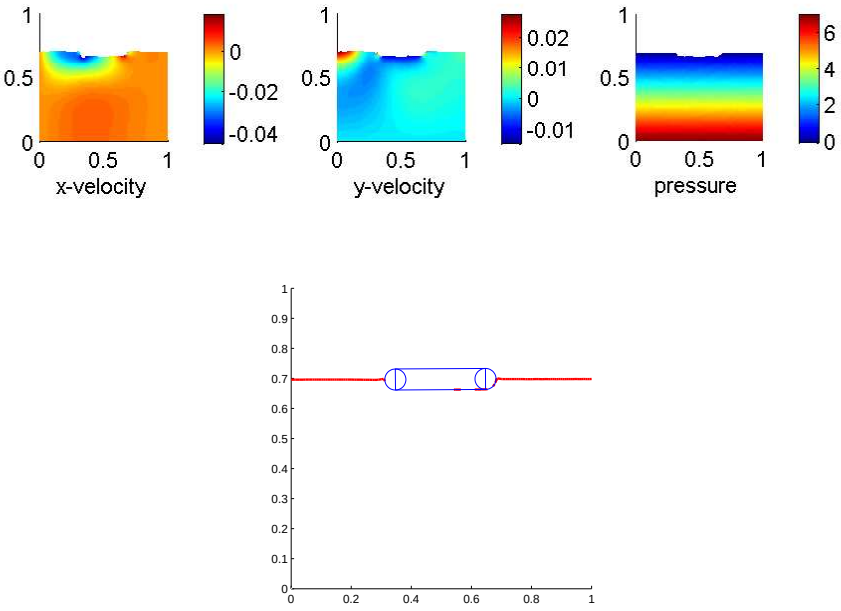


Figure 5.35: Unknown fields and free surface at t = 10.0

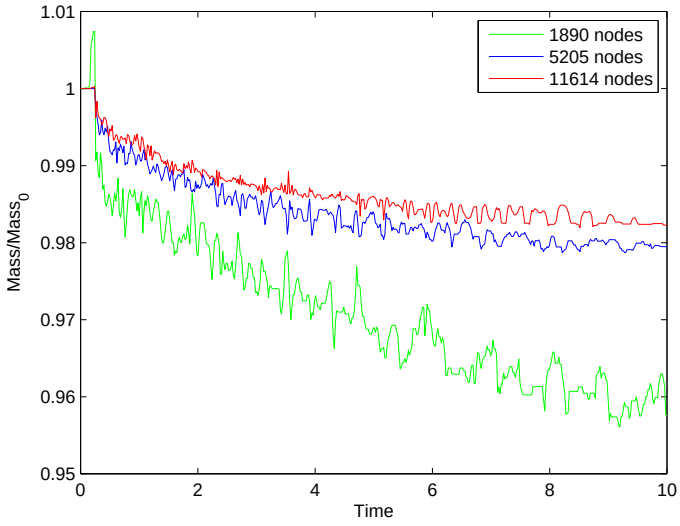


Figure 5.36: Time evolution of mass (with respect to initial mass)

5.7 Conclusions

In this chapter the FM-ALE approach has been applied to solid mechanics and Fluid-Structure Interaction problems. The main feature of the method is its capability of using a fixed background mesh but at the same time correctly taking into account the domain movement in the computation of the time derivatives. Moreover, values of the unknowns for the so-called *newly created nodes* are clearly and uniquely defined with the FM-ALE approach.

For solid mechanics problems the FM-ALE method is of special interest when the solid body is subject to very large strains. In this case Lagrangian formulations cannot be used due to the ill-conditioning caused by the large element stretch. The FM-ALE method, on the other hand, avoids element stretching by using a fixed mesh. A validation test has been carried out comparing results obtained with a classical Updated - Lagrangian formulation and the method proposed in this work. Results show that the method is robust and accurate.

The FM-ALE concept can be applied together with any time integration scheme. In the case of solid mechanics, we have shown how to use it in combination with classical θ schemes and fractional step methods that have a certain popularity in this context.

In the case of Fluid-Structure Interaction problems, the FM-ALE method can be applied to solve the flow and the solid mechanics problems. The main feature of this approach is the possibility of using a *single background mesh to solve both mechanical problems*. We have presented two numerical examples showing this particular capability. Even though monolithic solid-fluid coupling schemes have been employed, the possibility of using iteration-by-subdomain techniques is open.

For free surface problems the FM-ALE method avoids the need for remeshing which appears in classical Lagrangian or ALE methods. Moreover, the free surface is tracked in a very natural way with the level set function strategy, allowing for the solid body *breaking the free surface* without any further algorithmic steps. We have paid special attention to the interaction between the level set function and the solid boundary function which define the fluid domain:

in order to avoid the delay of the level set function with respect to the solid boundary function, we have modified the Stokes problem to be solved in the empty part of the domain, imposing the velocity divergence to be negative inside the solid body. The proposed method has been used to solve the problem of rigid bodies falling into water, and has proved to be robust and provide smooth solution fields, even at the critical instant in which the solid body contacts the free surface.

Chapter 6

Subscales on the element boundaries

In this chapter we introduce a way to approximate the subscales on the boundaries of the elements in a variational two-scale finite element approximation to flow problems. The key idea is that the subscales on the element boundaries must be such that the transmission conditions for the unknown, split as its finite element contribution and the subscale, hold. In particular, we consider the scalar convection-diffusion-reaction equation, the Stokes problem and Darcy's problem. For these problems the transmission conditions are the continuity of the unknown and its fluxes through element boundaries. The former is automatically achieved by introducing a single valued subscale on the boundaries (for the conforming approximations we consider), whereas the latter provides the effective condition for approximating these values. The final result is that the subscale on the interelement boundaries must be proportional to the jump of the flux of the finite element component and the average of the subscale calculated in the element interiors.

6.1 Introduction

The variational multiscale (VMS) framework to approximate boundary value problems starts with the variational formulation of the problem. In particular, in the two-scale version we consider, it consists in splitting the unknown and the test function into a component in a discrete approximating space and another component in its complement, for which an approximation needs to be proposed. This component is called subgrid scale or, simply, *subscale*. This idea was proposed in the finite element context in [76, 77]. The standard Galerkin method accommodates this framework simply by considering the subscales to be negligible.

The main interest of the VMS framework is to develop *stabilized* finite element methods in a broad sense, meaning that it allows to design discrete variational formulations that do not suffer from the stability problems of the standard Galerkin method. In particular, we are interested here in finite element methods for some model problems arising in fluids mechanics (see [29] for a review of different stabilization methods in flow problems).

The VMS concept as described above is quite general. The way to approximate the subscales is left open. Many questions arise, such as the space for these subscales, the problem to be solved to compute them or their behavior in time dependent problems. In principle, the problem for the subscales is *global*, that is to say, defined over all the computational domain.

In order to simplify it, some sort of *localization* is necessary, for example by assuming that the subscales vanish on the interelement boundaries, that is to say, they are bubble functions (see for example [12, 118] for application of this concept to flow problems).

The treatment of the subscales on the interelement boundaries is precisely the subject of this chapter. We propose a way to compute them based on the following ideas:

- We assume the subscales on the element interiors *computed*, and thus the localization process mentioned consists in computing these subscales without accounting for their boundary values.
- The subscales on the element boundaries are *single valued*, even if they are discontinuous in the element interiors. This requires a hybrid-type formalism to write the *exact* variational equations that we develop only in the first problem analyzed.
- The subscales on the element boundaries are computed by imposing that the correct transmission conditions of the problem at hand hold. Obviously, these transmission conditions are problem-dependent.
- The fluxes of the subscales on the interelement boundaries are approximated using a simple finite-difference scheme. This is *the only* approximation we use, apart from those shared with VMS methods that are required to approximate the subscales in the element interiors.

A completely different approach to compute subscales on the interelement boundaries is proposed in [2], where local problems along these boundaries are set.

We will not insist on other aspects of the VMS method, such as the problem for the subscales in the element interiors, the space where they belong or their time dependency. Let us only mention that we approximate them using an approximate Fourier analysis, that very often we compute them as L^2-orthogonal to the finite element space [30] and that we consider them time dependent in transient problems [32, 36]. In order to skip as much as possible this discussion, we will present our formulation without using the explicit expression for the subscales in the element interiors This approach is, as far as we know, original, and we use it mainly to focus the attention in the expression of the subscales on the interelement boundaries.

The particular transmission conditions between interelement boundaries, that serve us to compute the subscales on these boundaries, are problem dependent. This is why we will treat different problems arising in fluid mechanics, all of them linear and stationary. The first is the convection-diffusion-reaction (CDR) equation considered in Section 6.2. We show that the subscale on the element boundaries is proportional to the jump of the flux *with a negative sign*, and also to the average of the subscales computed in the element interiors adjacent to an edge and extended to this edge. In the following, "edge" will refer to the intersection between two element domains, understanding that it is a face in 3D problems. The sign of the subscales on the edges *subtracts* stability to the problem. However, we show that it is possible to control the new terms added. Neither for this problem nor for the other two discussed in the chapter we analyze convergence, since it depends on the particular expression of the subscale on the element interiors. Nevertheless, we provide stability results for all the problems treated.

There is no apparent gain in considering the subscales on the element edges for the CDR equation the way we do. However, the situation is different for the Stokes problem written in

velocity-pressure form analyzed in Section 6.3. We show that the subscales on the edges in this case introduce two terms, one that depends on the velocity gradients and that needs to be controlled with the viscous term and another one *that provides pressure stability*. This term is a least-squares form of the jump of the pressure across the edges, and therefore acts only when discontinuous pressure interpolations are used (note that it is not related to the jump stabilization technique proposed for example in [26, 25]). The term we add is similar to the one already introduced in [78], which has the local variant proposed and analyzed in [125, 85] for the Q_1/P_0 (bilinear-constant) and P_1/P_0 (linear-constant) velocity-pressure pairs.

Section 6.4 describes the application of our ideas to Darcy's problem. We propose a stabilized formulation that includes, with minor modifications, the methods proposed for example in [102] (and extended in [106]) and in [80]. As in the previous cases, we provide a stability result. In this case, the bilinear form associated to the problem is not coercive, but only an inf-sup condition can be proved.

Let us mention that the ideas presented here can be applied to other problems. In particular, in [34] a method to compute the subscales on the element boundaries for the stress-velocity-pressure formulation of the Stokes problem is proposed and fully analyzed. In this case, subscales on the boundaries are essential to deal with discontinuous pressure and stress interpolations.

The main contributions of our approach can be summarized as follows:

- To provide a consistent VMS justification to some stabilizing terms introduced in previous works to deal with discontinuous pressures.

- To propose a *symmetric* stabilized problem for the Stokes and the Darcy equations (if subscales in the element interiors can be considered negligible compared to the jump of the stresses, see Remark 5). The sign of the symmetric operator, which subtracts stability from the Galerkin terms, is crucial to achieve this symmetry. The situation is similar to what happens when minus the adjoint of the differential operator applied to the test functions is used instead of the original differential operator in the stabilizing terms. This suggestion was first introduced in [55] and turns out to be completely natural in the VMS framework. Also in this case, the diffusive term *subtracts* stability in the case of the CDR equation.

- Even though we do not exploit this point here, our approach suggests how to stabilize Neumann boundary conditions, essential for example in some fluid-structure interaction problems (see Remark 6).

Some numerical examples are presented in Section 6.5. Since the stabilizing effect of the boundary terms introduced for the different problems is well known, we simply check what is particular of our approach, namely, the terms that may deteriorate stability. We show that this is not the case in two cases, namely, a convection-diffusion example and two Stokes problems. As the stability analysis dictates, these terms can be controlled by the rest of the terms appearing in the stabilized formulation. Moreover, in the Stokes problem case, some discontinuous pressure interpolations unstable using the Galerkin method, such as the P_1/P_0 pair (see Section 6.5) can be used.

In Section 6.6 we look for an efficient implementation of the $P1/P0$ interpolation. Our approach consists of condensating the pressure unknowns by sending the off-diagonal terms corresponding to the pressure test function equations to the right hand-side. We illustrate the performance of these algorithms with some convergence plots.

Finally, some conclusions close the chapter in Section 6.7.

6.2 Convection-diffusion-reaction equation

6.2.1 Problem statement

Let us consider the boundary value problem:

$$\mathcal{L}u := -k\Delta u + \boldsymbol{a}\cdot\nabla u + su = f \quad \text{in } \Omega, \tag{6.1}$$

$$u = 0 \quad \text{on } \partial\Omega, \tag{6.2}$$

where $\Omega \subset \mathbb{R}^d$ is a bounded domain, with $d = 2, 3$, $u : \Omega \longrightarrow \mathbb{R}$ is the unknown, k is the diffusion coefficient, s the reaction coefficient, $\boldsymbol{a}$ the advection velocity and f the given source term. For simplicity, we assume $k > 0$, $s \geq 0$ and the advection velocity $\boldsymbol{a}$ all constants.

Let $V = H_0^1(\Omega)$ and assume $f \in H^{-1}(\Omega)$. The variational form of the problem consists of finding $u \in V$ such that

$$B(u, v) := k(\nabla u, \nabla v) + (\boldsymbol{a}\cdot\nabla u, v) + s(u, v) = \langle f, v\rangle =: L(v) \quad \forall v \in V. \tag{6.3}$$

Here and below, $(\cdot,\cdot)$ denotes the L^2 product in Ω. In general, the integral of two function g_1 and g_2 over a domain ω will be denoted by $\langle g_1, g_2\rangle_\omega$ and the norm in a function space X by $\|\cdot\|_X$, with the simplifications $\|\cdot\|_{L^2(\Omega)} \equiv \|\cdot\|$ and $\langle\cdot,\cdot\rangle_\Omega \equiv \langle\cdot,\cdot\rangle$. This symbol will also be used for the duality pairing.

6.2.2 Six and four field formulations

As mentioned earlier, we consider that the subscales on the element boundaries are single valued, even if they are discontinuous in the element interiors. To give a variational foundation to the approximation presented in the next subsection, let us consider a hybrid-type approach, starting with a particular six field formulation of the problem. For simplicity, let us assume that $\bar\Omega = \bar\Omega_1 \cup \bar\Omega_2$, with $\Gamma = \partial\Omega_1 \cap \partial\Omega_2$. Consider a decomposition of $V = H_0^1(\Omega)$ of the form $V = \bar V \oplus V'$, and let $u = \bar u + u'$ be the corresponding decomposition of the unknown. Let us state a variational formulation of the problem taking as unknowns $\bar u$, u', their traces on Γ, denoted by $\bar\gamma$ and γ', respectively, and their fluxes, denoted by $\bar\lambda$ and λ', respectively. The space of traces $T = H_{00}^{1/2}(\Gamma)$ and the space of fluxes $F = (H_{00}^{1/2}(\Gamma))'$ (the dual of T) are also assumed to be split as $T = \bar T \oplus T'$ and $F = \bar F \oplus F'$. Note that the prime in V', T' and F' is *not* used to denote the dual of a space.

If we denote with a subscript i the restriction of $\bar u$, u', $\bar\lambda$, λ', B and L to subdomain i

($i = 1, 2$), the problem for the six fields $\bar{u}$, u', $\bar{\gamma}$, γ', $\bar{\lambda}$ and λ' can be written as

$$
\begin{aligned}
&B_1(\bar{u}_1, \bar{v}_1) + B_1(u'_1, \bar{v}_1) - \left\langle \bar{\lambda}_1 + \lambda'_1, \bar{v}_1 \right\rangle_\Gamma = L_1(\bar{v}_1) && \forall\, \bar{v}_1, && (6.4)\\
&B_1(\bar{u}_1, v'_1) + B_1(u'_1, v'_1) - \left\langle \bar{\lambda}_1 + \lambda'_1, v'_1 \right\rangle_\Gamma = L_1(v'_1) && \forall\, v'_1, && (6.5)\\
&B_2(\bar{u}_2, \bar{v}_2) + B_2(u'_2, \bar{v}_2) - \left\langle \bar{\lambda}_2 + \lambda'_2, \bar{v}_2 \right\rangle_\Gamma = L_2(\bar{v}_2) && \forall\, \bar{v}_2, && (6.6)\\
&B_2(\bar{u}_2, v'_2) + B_2(u'_2, v'_2) - \left\langle \bar{\lambda}_2 + \lambda'_2, v'_2 \right\rangle_\Gamma = L_2(v'_2) && \forall\, v'_2, && (6.7)\\
&\langle \bar{\mu}_1, \bar{\gamma} + \gamma' - \bar{u}_1 - u'_1 \rangle_\Gamma = 0 && \forall\, \bar{\mu}_1, && (6.8)\\
&\langle \mu'_1, \bar{\gamma} + \gamma' - \bar{u}_1 - u'_1 \rangle_\Gamma = 0 && \forall\, \mu'_1, && (6.9)\\
&\langle \bar{\mu}_2, \bar{\gamma} + \gamma' - \bar{u}_2 - u'_2 \rangle_\Gamma = 0 && \forall\, \bar{\mu}_2, && (6.10)\\
&\langle \mu'_2, \bar{\gamma} + \gamma' - \bar{u}_2 - u'_2 \rangle_\Gamma = 0 && \forall\, \mu'_2, && (6.11)\\
&\left\langle \bar{\kappa}, \bar{\lambda}_1 + \lambda'_1 + \bar{\lambda}_2 + \lambda'_2 \right\rangle_\Gamma = 0 && \forall\, \bar{\kappa}, && (6.12)\\
&\left\langle \kappa', \bar{\lambda}_1 + \lambda'_1 + \bar{\lambda}_2 + \lambda'_2 \right\rangle_\Gamma = 0 && \forall\, \kappa'. && (6.13)
\end{aligned}
$$

In these equations, $\lambda_i = \bar{\lambda}_i + \lambda'_i \in F$ are the fluxes computed from the side of Ω_i and $\mu_i = \bar{\mu}_i + \mu'_i \in F$ the corresponding test functions ($i = 1, 2$). The test function for the trace of the unknown $\gamma = \bar{\gamma} + \gamma' \in T$ is denoted by $\kappa = \bar{\kappa} + \kappa' \in T$. The boundary terms in (6.4)-(6.7) correspond to the weak imposition of fluxes on Γ, equations (6.8)-(6.11) to the weak continuity of $u_i = \bar{u}_i + u'_i$ on Γ ($i = 1, 2$) and equations (6.12)-(6.13) to the weak continuity of fluxes on Γ.

The previous formulation can be considered a straightforward extension of the classical three field formulation for u, γ and λ, obtained by a splitting of the spaces where these unknowns belong (see [116] for a three field formulation of the convection-diffusion equation). Our particular formulation is obtained by imposing the fluxes of $\bar{u}$ to be $\bar{\lambda}_i = \boldsymbol{n}_i \cdot (k\nabla \bar{u}_i + \boldsymbol{a}\bar{u}_i)|_\Gamma$, where $\boldsymbol{n}_i$ is the normal to Γ from Ω_i, and $\bar{\gamma} = \bar{u}_i|_\Gamma$ ($i = 1, 2$). In other words, *we prescribe the fluxes and the continuity of $\bar{u}$ as in the one field variational formulation (6.3), but treat u', γ' and λ' as in the standard three field formulation*. This approach in particular implies that the test functions $\bar{\mu}_i$ must be of the form $\bar{\mu}_i = \boldsymbol{n}_i \cdot (k\nabla \bar{v}_i + \boldsymbol{a}\bar{v}_i)|_\Gamma$, for $\bar{v}_i \in \bar{V}_i$, and $\bar{\kappa} = \bar{v}|_\Gamma$, with $\bar{v} \in \bar{V}$. Therefore, the previous problem reads

$$
\begin{aligned}
&B_1(\bar{u}_1, \bar{v}_1) + B_1(u'_1, \bar{v}_1) - \langle \boldsymbol{n}_1 \cdot (k\nabla \bar{u}_1 + \boldsymbol{a}\bar{u}_1) + \lambda'_1, \bar{v}_1 \rangle_\Gamma = L_1(\bar{v}_1) && \forall\, \bar{v}_1, && (6.14)\\
&B_1(\bar{u}_1, v'_1) + B_1(u'_1, v'_1) - \langle \boldsymbol{n}_1 \cdot (k\nabla \bar{u}_1 + \boldsymbol{a}\bar{u}_1) + \lambda'_1, v'_1 \rangle_\Gamma = L_1(v'_1) && \forall\, v'_1, && (6.15)\\
&B_2(\bar{u}_2, \bar{v}_2) + B_2(u'_2, \bar{v}_2) - \langle \boldsymbol{n}_2 \cdot (k\nabla \bar{u}_2 + \boldsymbol{a}\bar{u}_2) + \lambda'_2, \bar{v}_2 \rangle_\Gamma = L_2(\bar{v}_2) && \forall\, \bar{v}_2, && (6.16)\\
&B_2(\bar{u}_2, v'_2) + B_2(u'_2, v'_2) - \langle \boldsymbol{n}_2 \cdot (k\nabla \bar{u}_2 + \boldsymbol{a}\bar{u}_2) + \lambda'_2, v'_2 \rangle_\Gamma = L_2(v'_2) && \forall\, v'_2, && (6.17)\\
&\langle \boldsymbol{n}_1 \cdot (k\nabla \bar{v}_1 + \boldsymbol{a}\bar{v}_1), \gamma' - u'_1 \rangle_\Gamma = 0 && \forall\, \bar{v}_1, && (6.18)\\
&\langle \mu'_1, \gamma' - u'_1 \rangle_\Gamma = 0 && \forall\, \mu'_1, && (6.19)\\
&\langle \boldsymbol{n}_2 \cdot (k\nabla \bar{v}_2 + \boldsymbol{a}\bar{v}_2), \gamma' - u'_2 \rangle_\Gamma = 0 && \forall\, \bar{v}_2, && (6.20)\\
&\langle \mu'_2, \gamma' - u'_2 \rangle_\Gamma = 0 && \forall\, \mu'_2, && (6.21)\\
&\langle \bar{\kappa}, \boldsymbol{n}_1 \cdot (k\nabla \bar{u}_1 + \boldsymbol{a}\bar{u}_1) + \boldsymbol{n}_2 \cdot (k\nabla \bar{u}_2 + \boldsymbol{a}\bar{u}_2) + \lambda'_1 + \lambda'_2 \rangle_\Gamma = 0 && \forall\, \bar{\kappa}, && (6.22)\\
&\langle \kappa', \boldsymbol{n}_1 \cdot (k\nabla \bar{u}_1 + \boldsymbol{a}\bar{u}_1) + \boldsymbol{n}_2 \cdot (k\nabla \bar{u}_2 + \boldsymbol{a}\bar{u}_2) + \lambda'_1 + \lambda'_2 \rangle_\Gamma = 0 && \forall\, \kappa'. && (6.23)
\end{aligned}
$$

Adding up (6.14) and (6.16) and using (6.22) yields the original variational equation projected

onto $\bar{V}$, that is to say,

$$B(\bar{u},\bar{v}) + B(u',\bar{v}) = L(\bar{v}) \qquad \forall\, \bar{v}.$$

It is understood that $B(u',\bar{v}) = B_1(u_1',\bar{v}_1) + B_2(u_2',\bar{v}_2)$. Integrating these terms by parts and using (6.18) and (6.20) we get

$$B(\bar{u},\bar{v}) + \sum_{i=1}^{2} \langle u', \mathcal{L}^*\bar{v}\rangle_{\Omega_i} + \sum_{i=1}^{2} \langle \gamma', \boldsymbol{n}_i \cdot (k\nabla\bar{v}_i + \boldsymbol{a}\bar{v}_i)\rangle_\Gamma = L(\bar{v}), \tag{6.24}$$

where

$$\mathcal{L}^*\bar{v} := -k\Delta\bar{v} - \boldsymbol{a}\cdot\nabla\bar{v} + s\bar{v}$$

is the formal adjoint of $\mathcal{L}$. Adding up (6.15) and (6.17) and integrating the first terms by parts we get

$$\begin{aligned}\sum_{i=1}^{2} &(B_i(\bar{u},v') + B_i(u',v') - \langle \boldsymbol{n}_i\cdot(k\nabla\bar{u}_i + \boldsymbol{a}\bar{u}_i) + \lambda_i', v_i'\rangle_\Gamma) \\ &= \sum_{i=1}^{2} \left(\langle \mathcal{L}\bar{u}, v'\rangle_{\Omega_i} + B_i(u',v') - \langle \lambda_i', v_i'\rangle_\Gamma\right) = \sum_{i=1}^{2} L_i(v'). \end{aligned}\tag{6.25}$$

It is understood in this equation that $\bar{(\cdot)}|_{\Omega_i} = \bar{(\cdot)}_i$. The final problem can be written as (6.24), (6.25), (6.23) and the addition of (6.19) and (6.21), that is to say,

$$B(\bar{u},\bar{v}) + \sum_{i=1}^{2} \langle u', \mathcal{L}^*\bar{v}\rangle_{\Omega_i} + \sum_{i=1}^{2} \langle \gamma', \boldsymbol{n}_i \cdot (k\nabla\bar{v}_i + \boldsymbol{a}\bar{v}_i)\rangle_\Gamma = L(\bar{v}) \qquad \forall\, \bar{v}, \tag{6.26}$$

$$\sum_{i=1}^{2} \langle \mathcal{L}\bar{u}, v'\rangle_{\Omega_i} + B(u',v') - \sum_{i=1}^{2} \langle \lambda_i', v_i'\rangle_\Gamma = L(v') \qquad \forall\, v', \tag{6.27}$$

$$\sum_{i=1}^{2} \langle \kappa', \boldsymbol{n}_i\cdot(k\nabla\bar{u}_i + \boldsymbol{a}\bar{u}_i) + \lambda_i'\rangle_\Gamma = 0 \qquad \forall\, \kappa', \tag{6.28}$$

$$\sum_{i=1}^{2} \langle \mu_i', \gamma' - u_i'\rangle_\Gamma = 0 \qquad \forall\, \mu_1', \mu_2'. \tag{6.29}$$

This is the four field formulation we were looking for. Its importance relies on the fact that *it is the theoretical framework to develop approximations in which u is split into a contribution which is continuous on Γ and another one which is discontinuous*. Obviously, this formulation is symmetric for symmetric problems (in our case, if $\boldsymbol{a} = \boldsymbol{0}$).

6.2.3 Finite element approximation

Let $\mathcal{T}_h := \{K\}$ be a finite element partition of the domain Ω of size h, and $V_h \subset V$ a finite element space where an approximate solution $u_h \in V_h$ is sought. We assume that this space

is made of continuous functions. To simplify the analysis, we will assume that the family of finite element partitions $\mathcal{F} = \{\mathcal{T}_h\}_{h>0}$ is quasi-uniform, so that all the element sizes are bounded above and below by constants multiplied by h. We will also use the abbreviations $\|\cdot\|_{L^2(K)} \equiv \|\cdot\|_K$ and $\|\cdot\|_{L^2(\partial K)} \equiv \|\cdot\|_{\partial K}$.

Consider the previous setting with $\bar{V} = V_h$, and therefore $V = V_h \oplus V'$, with V' to be defined, and $u = u_h + u'$, $v = v_h + v'$. In order to focus our attention on the expression for the subscales on the interelement boundaries, we will not specify the choice for V', which depends on the particular VMS approximation used.

As before, let also γ' be the trace of u' on the interelement boundaries and λ' the flux, being the corresponding spaces T' and F', and the corresponding test functions $\kappa' \in T'$ and $\mu' \in F'$. According to the four field formulation presented in the previous subsection, now considering Ω split into the element domains of the finite element partition, the variational problem (6.3) is exactly equivalent to find $u_h \in V_h$, $u' \in V'$, $\gamma' \in T'$ and $\lambda' \in F'$ such that

$$B(u_h, v_h) + \sum_K \langle u', \mathcal{L}^* v_h \rangle_K + \sum_K \langle \gamma', k\partial_n v_h \rangle_{\partial K} = L(v_h) \quad \forall v_h \in V_h, \tag{6.30}$$

$$\sum_K \langle \mathcal{L} u_h, v' \rangle_K + B(u', v') - \sum_K \langle \lambda', v' \rangle_{\partial K} = L(v') \quad \forall v' \in V', \tag{6.31}$$

$$\sum_K \langle \kappa', k\partial_n u_h + \lambda' \rangle_{\partial K} = 0 \quad \forall \kappa' \in T', \tag{6.32}$$

$$\sum_K \langle \mu', \gamma' - u' \rangle_{\partial K} = 0 \quad \forall \mu' \in F'. \tag{6.33}$$

Note that the jumps of the convective fluxes are zero because of the continuity assumed for the finite element functions.

The approximation process consists of different ingredients, *all aiming at giving a closed problem for u_h alone*. For that we will propose heuristic approximations for γ' and λ' and then we will perform a stability analysis to check that the resulting formulation is stable. Let us insist that, up to this point, problem (6.30)-(6.33) is exact. Furthermore, for $u_h = 0$ it could be used as the variational framework to develop discontinuous Galerkin approximations (see Remark 2 below).

6.2.4 Subscales on the element boundaries

Let us consider for simplicity the 2D case and the situation depicted in Fig. 6.1, where two elements K_1 and K_2 share an edge E (recall that E stands for "edge" in 2D or face in 3D). Unless otherwise indicated (see Remark 1 below), all the edges are considered interior, that is to say, the element boundaries on $\partial\Omega$ are excluded.

Let u'_i be the subscale approximated in the interior of element K_i, $i = 1, 2$. *We assume that this approximation is valid up to a distance δ to the element boundary*. This distance will be taken of the form $\delta = \delta_0 h$, with $0 \leq \delta_0 \leq 1/2$.

Approximation of λ'. The values of λ' on ∂K are *weak* approximations to the fluxes of u'. Given the trace γ' of this unknown, we delete (6.33) and propose the following closed form

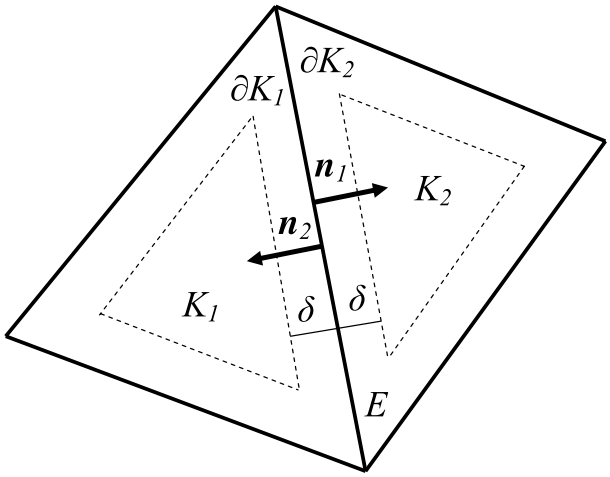


Figure 6.1: Notation for the approximation of the subscales on the element boundaries

expression for λ':

$$\lambda'_{\partial K_i \cap E} \approx k\frac{\gamma'_E - u'_i}{\delta}, \quad i = 1, 2, \tag{6.34}$$

where now u'_i has to be understood as the subscale computed in the element interiors and evaluated at edge E. We want to remark that, apart from the assumptions inherent to the VMS framework and the imposition of the transmission conditions (see below), this is the only approximation we really require to compute the subscales on the interelement boundaries. Obviously, other finite-difference-like approximations to the fluxes of the subscales could be adopted.

Approximation of γ'. Equation (6.32) states the weak continuity of the total fluxes on the element boundaries. The idea now is to replace this equation by an explicit prescription of this continuity. If $[\![\boldsymbol{n} g]\!]_E := \boldsymbol{n}_1 g|_{\partial K_1 \cap E} + \boldsymbol{n}_2 g|_{\partial K_2 \cap E}$ denotes the jump of a scalar function g across edge E and $[\![\partial_n g]\!]_E = \boldsymbol{n}_1 \cdot \nabla g|_{\partial K_1 \cap E} + \boldsymbol{n}_2 \cdot \nabla g|_{\partial K_2 \cap E}$ the jump of the normal derivative, the continuity of the total fluxes can be imposed as follows:

$$\begin{aligned} 0 = [\![k\partial_n u]\!]_E &= [\![k\partial_n u_h]\!]_E + \lambda'_{\partial K_1 \cap E} + \lambda'_{\partial K_2 \cap E} \\ &\approx [\![k\partial_n u_h]\!]_E + k\frac{\gamma'_E - u'_1}{\delta} + k\frac{\gamma'_E - u'_2}{\delta}. \end{aligned} \tag{6.35}$$

From this expression, and for k constant, we obtain the approximation we were looking for:

$$\boxed{\gamma'_E \approx \{u'\}_E - \frac{\delta}{2}[\![\partial_n u_h]\!]_E} \tag{6.36}$$

where $\{u'\}_E := \frac{1}{2}(u'_1 + u'_2)$ is the average of the subscales computed in the element interiors evaluated at edge E. From (6.36) it is observed that δ_0 will play the role of an algorithmic parameter for which, following our approach, we have a geometrical interpretation.

From now onwards we will use the symbol $=$ instead of $\approx$, understanding that in some places we perform approximation (6.34) that has led us to (6.36).

Remark 1 (Neumann boundary conditions). Suppose that $F_K = \partial K \cap \partial\Omega$ and that instead of the Dirichlet condition (6.2) the Neumann condition $-k\partial_n u = q$ is prescribed. In this case, (6.35) should be replaced by

$$q = -k\partial_n u_h|_{F_K} - k\frac{\gamma'_{F_K} - u'_K}{\delta},$$

so that the contribution to L in (6.3) that would appear due to the Neumann condition would be modified by the approximation to the subscale on the boundary, and there would be also a contribution to the bilinear form B. We will come back to this point in the case of the Stokes problem, where this fact has more important consequences. △

Problem for u_h and u'. From (6.36) we obtain the following approximation for the fluxes of the subscales:

$$\begin{aligned}\lambda'_{\partial K_i \cap E} &= \frac{k}{\delta}\left(\{u'\}_E - u'_i\right) - \frac{k}{2}[\![\,\partial_n u_h\,]\!]_E \\ &= -\frac{k}{2\delta}\boldsymbol{n}_i \cdot [\![\,\boldsymbol{n}u'\,]\!]_{\partial K_i \cap E} - \frac{k}{2}[\![\,\partial_n u_h\,]\!]_E, \quad i = 1, 2.\end{aligned} \tag{6.37}$$

Once λ' and γ' are approximated, the problem we are left with reads as follows: find $u_h \in V_h$ and $u' \in V'$ such that

$$B(u_h, v_h) + \sum_K \langle u', \mathcal{L}^* v_h\rangle_K - \frac{k\delta}{2}\sum_E \langle [\![\,\partial_n u_h\,]\!], [\![\,\partial_n v_h\,]\!]\rangle_E + \sum_E \langle \{u'\}, k[\![\,\partial_n v_h\,]\!]\rangle_E = L(v_h), \tag{6.38}$$

$$B(u', v') + \sum_K \langle \mathcal{L}u_h, v'\rangle_K + \frac{k}{2\delta}\sum_E \langle [\![\,\boldsymbol{n}u'\,]\!], [\![\,\boldsymbol{n}v'\,]\!]\rangle_E + \sum_E \langle k[\![\,\partial_n u_h\,]\!], \{v'\}\rangle_E = L(v'), \tag{6.39}$$

for all $v_h \in V_h$ and $v' \in V'$.

Remark 2 Observe that this system of variational equations can be understood as *a general framework to approximate unknowns with a continuous part (u_h) and an approximated discontinuous part (u').* Furthermore, if the continuous part is zero, we are left with (6.39) with $u_h = 0$, which corresponds to *the classical Galerkin method enforcing continuity across interelement boundaries through Nitsche's method*, although with approximation (6.37) for the fluxes, so that the classical terms involving $\partial_n u'$ and $\partial_n v'$ are missing (see [5, 64]). For piecewise constant approximations these terms would not appear, and we would obtain a classical piecewise-constant discontinuous Galerkin approximation. △

6.2.5 Subscales in the element interiors

Up to now we have replaced variational equations for the fluxes of the subscales and their traces by approximated closed form expressions. It can be seen from problem (6.38)-(6.39) that the resulting formulation is symmetric for symmetric problems. However, now we will use and additional approximation that will make the problem loose its symmetry, but that will

greatly simplify the implementation of the formulation. This approximation is inherent to all VMS formulations to yield a closed form expression for the subscales in the element interiors.

If we integrate the second term in the left-hand-side (LHS) of (6.31) by parts we get

$$B(u',v') = \sum_K \langle \mathcal{L}u', v' \rangle_K + \sum_K \langle k\partial_n u', v' \rangle_{\partial K}.$$

If instead of using (6.37) we assume that λ' approximates $k\partial_n u'$, the second term in this last expression cancels with the third one in the LHS of (6.31). Therefore, the final problem is: find $u_h \in V_h$ and $u' \in V'$ such that

$$B(u_h, v_h) + \sum_K \langle u', \mathcal{L}^* v_h \rangle_K + \sum_E \left\langle \{u'\} - \frac{\delta}{2} [\![\partial_n u_h]\!], k [\![\partial_n v_h]\!] \right\rangle_E = L(v_h), \tag{6.40}$$

$$\sum_K \langle \mathcal{L}u_h, v' \rangle_K + \sum_K \langle \mathcal{L}u', v' \rangle_K = L(v'), \tag{6.41}$$

for all $v_h \in V_h$ and $v' \in V'$. The last term in the LHS of (6.40) is the main novelty with respect to classical stabilized finite element methods designed in the variational multiscale framework.

Remark 3 Note that if in (6.39) v' is considered *continuous* we obtain (6.41) *with no additional approximation*. In other words, if the subscale is approximated with a Petrov-Galerkin method (leading to a non-symmetric formulation) in which the space of test functions is continuous, we recover (6.41). This is not however the approach we will adopt. △

It only remains to approximate u' in the element interiors. To this end, in (6.41) the approximation

$$\langle \mathcal{L}u', v' \rangle_K = \tau^{-1} \langle u', v' \rangle_K, \quad \tau = \left(C_1 \frac{k}{h^2} + C_2 \frac{|\boldsymbol{a}|}{h} \right)^{-1} \tag{6.42}$$

may be adopted. This can be motivated by a Fourier analysis of the problem for the subscales [32]. In particular, it implies that *the subscales in the element interiors are not affected by their boundary values*. This simplification makes the formulation we propose feasible from the implementation standpoint. Let us stress once again that this approximation is not original of this work, but common to all VMS methods that compute locally the subscales in the element interiors.

Once all the approximations are made, the final problem is to find $u_h \in V_h$ and $u' \in V'$ such that

$$B(u_h, v_h) + \sum_K \langle u', \mathcal{L}^* v_h \rangle_K + \sum_E \left\langle \{u'\} - \frac{\delta}{2} [\![\partial_n u_h]\!], k [\![\partial_n v_h]\!] \right\rangle_E = L(v_h), \tag{6.43}$$

$$\sum_K \langle \mathcal{L}u_h, v' \rangle_K + \sum_K \tau^{-1} \langle u', v' \rangle_K = L(v'), \tag{6.44}$$

for all $v_h \in V_h$ and $v' \in V'$.

The variational equation (6.44) automatically yields and expression for the subscales in the element interiors in terms of the finite element component, provided V' is approximated by a space of discontinuous functions. It implies that

$$u' = \tau P_{V'}(f - \mathcal{L}u_h),$$

where $P_{V'}$ is the projection onto V'. However, it will be convenient for the following analysis to keep u' as unknown of the problem. Particular cases of projection that fit into the present framework are the orthogonal subscales stabilization (OSS) proposed in [30] and the algebraic version of the subgrid-scale stabilization (ASGS) (see [29, 76]), where $P_{V'}$ is the identity (at least when applied to $f - \mathcal{L}u_h$). The expression of τ (6.42) is in fact not important, except for a condition on constant C_1 indicated later.

6.2.6 Stability analysis

Let us consider the bilinear form of the problem in $(V_h \times V') \times (V_h \times V')$:

$$B_{\text{exp}}(u_h, u'; v_h, v') := B(u_h, v_h) + \sum_K \langle u', \mathcal{L}^* v_h \rangle_K + \sum_E \langle \{u'\}, k[\![\partial_n v_h]\!] \rangle_E$$
$$- \frac{\delta}{2} \sum_E \langle [\![\partial_n u_h]\!], k[\![\partial_n v_h]\!] \rangle_E + \sum_K \langle \mathcal{L}u_h, v' \rangle_K + \sum_K \tau^{-1} \langle u', v' \rangle_K.$$

Let us prove stability of the problem by showing that B_{exp} is coercive in a certain norm. We have that

$$\begin{aligned} B_{\text{exp}}(u_h, u'; u_h, u') &= B(u_h, u_h) + \sum_K \langle u', \mathcal{L}^* u_h + \mathcal{L}u_h \rangle_K + \sum_E \langle \{u'\}, k[\![\partial_n u_h]\!] \rangle_E \\ &\quad - \frac{\delta}{2} \sum_E k \| [\![\partial_n u_h]\!] \|_E^2 + \sum_K \tau^{-1} \|u'\|_K^2 \\ &\geq k\|\nabla u_h\|^2 + s\|u_h\|^2 - \sum_K \|u'\|_K \| -2k\Delta u_h + 2su_h \|_K \\ &\quad - \sum_E \|\{u'\}\|_E k \| [\![\partial_n u_h]\!] \|_E - \frac{\delta}{2} \sum_E k \| [\![\partial_n u_h]\!] \|_E^2 + \sum_K \tau^{-1} \|u'\|_K^2. \end{aligned}$$

We assume now that the classical inverse estimates

$$\|\Delta v_h\|_K^2 \leq \frac{C_{\text{inv}}}{h^2} \|\nabla v_h\|_K^2, \quad \|v_h\|_{L^\infty(K)}^2 \leq \frac{C_{\text{inv}}}{h^d} \|v_h\|_K^2 \quad \forall v_h \in V_h, \tag{6.45}$$

hold true (see [48, 22]). In particular, the second, which also holds for derivatives of finite element functions, implies the trace inequality

$$\|v_h\|_{\partial K}^2 \leq C_{\text{tr}} h^{-1} \|v_h\|_K^2, \tag{6.46}$$

which applied to $\partial_n v_h$ yields

$$\|\partial_n v_h\|_{\partial K}^2 \leq C_{\text{tr}} h^{-1} \|\nabla v_h\|_K^2.$$

Using these inverse estimates, we have (see Fig. 6.1 for the notation):

$$\begin{aligned}
-\frac{\delta}{2}\sum_E k\|[\![\,\partial_n u_h\,]\!]\|_E^2 &= -\frac{\delta}{2}\sum_E k\|\partial_n u_h|_{\partial K_1\cap E} + \partial_n u_h|_{\partial K_2\cap E}\|_E^2 \\
&\geq -\frac{\delta}{2}\sum_K 2k\|\partial_n u_h\|_{\partial K}^2 \\
&\geq -\frac{\delta_0 h}{2}\sum_K 2kC_{\mathrm{tr}}h^{-1}\|\nabla u_h\|_K^2 \\
&= -\delta_0 C_{\mathrm{tr}}k\|\nabla u_h\|^2.
\end{aligned}$$

Let us obtain a working inequality. Let a and b be discontinuous positive functions defined on the finite element partition. Using the notation $a_i := a|_{\partial K_i\cap E}$, for any $\beta > 0$ we have that

$$\begin{aligned}
\sum_E (a_1+a_2)(b_1+b_2) &\leq \sum_E \frac{h}{2\beta}(a_1+a_2)^2 + \sum_E \frac{\beta}{2h}(b_1+b_2)^2 \\
&\leq \sum_E \frac{h}{\beta}(a_1^2+a_2^2) + \sum_E \frac{\beta}{h}(b_1^2+b_2^2) \\
&\leq \sum_K \frac{h}{\beta}a|_{\partial K}^2 + \sum_K \frac{\beta}{h}b|_{\partial K}^2.
\end{aligned}$$

Now we make the assumption that the subscales are such that the inverse estimates also hold for them. Using the previous inequality we obtain, for any $\beta_3 > 0$:

$$\begin{aligned}
-\sum_E \|\{u'\}\|_E k\|[\![\,\partial_n u_h\,]\!]\|_E &= -\sum_E k\frac{1}{2}\|u_1'+u_2'\|_E\|\partial_n u_h|_{\partial K_1\cap E} + \partial_n u_h|_{\partial K_2\cap E}\|_E \\
&\geq -\sum_K \frac{\beta_3}{2}C_{\mathrm{tr}}\frac{k}{h^2}\|u'\|_K^2 - \sum_K \frac{1}{2\beta_3}C_{\mathrm{tr}}k\|\nabla u_h\|_K^2.
\end{aligned}$$

Using the bounds obtained, it follows that

$$
\begin{aligned}
B_{\exp}(u_h,u';u_h,u') &\geq k\|\nabla u_h\|^2+s\|u_h\|^2 \\
&\quad -\sum_K \|u'\|_K 2k\frac{C_{\text{inv}}^{1/2}}{h}\|\nabla u_h\|_K-\sum_K\|u'\|_K 2s\|u_h\|_K \\
&\quad -\delta_0 C_{\text{tr}}k\|\nabla u_h\|^2-\sum_E\|\{u'\}\|_E k\|[\![\,\partial_n u_h\,]\!]\|_E+\sum_K\tau^{-1}\|u'\|_K^2 \\
&\geq k\|\nabla u_h\|^2+s\|u_h\|^2 \\
&\quad -\sum_K\left(\beta_1 k\frac{C_{\text{inv}}}{h^2}\|u'\|_K^2+\frac{1}{\beta_1}k\|\nabla u_h\|_K^2\right)-\sum_K\left(\beta_2 s\|u'\|_K^2+\frac{1}{\beta_2}s\|u_h\|_K^2\right) \\
&\quad -\sum_K\frac{\beta_3}{2}C_{\text{tr}}\frac{k}{h^2}\|u'\|_K^2-\sum_K\frac{1}{2\beta_3}C_{\text{tr}}k\|\nabla u_h\|_K^2 \\
&\quad -\delta_0 C_{\text{tr}}k\|\nabla u_h\|^2+\sum_K\tau^{-1}\|u'\|_K^2 \\
&= \sum_K\left(1-\frac{1}{\beta_1}-\delta_0 C_{\text{tr}}-C_{\text{tr}}\frac{1}{2\beta_3}\right)k\|\nabla u_h\|_K^2+\sum_K\left(1-\frac{1}{\beta_2}\right)s\|u_h\|_K^2 \\
&\quad +\sum_K\left(\tau^{-1}-\beta_1 k\frac{C_{\text{inv}}}{h^2}-s\beta_2-\frac{\beta_3}{2}C_{\text{tr}}\frac{k}{h^2}\right)\|u'\|_K^2,
\end{aligned}
$$

where β_i are constants, $i=1,2,3$. Taking these constants sufficiently large, δ_0 sufficiently small and C_1 in the definition of τ large enough, the following result follows:

Theorem 1 *There are constants C_1 and δ_0 in the definition of the stabilization parameters such that*

$$
B_{\exp}(u_h,u';u_h,u')\geq C\left(k\|\nabla u_h\|^2+s\|u_h\|^2+\sum_K\tau^{-1}\|u'\|_K^2\right). \tag{6.47}
$$

Remark 4 Let us enumerate the essential ideas and highlight the original aspects of the analysis presented in this section:

- The driving idea is that the subscales on the boundary are determined by the transmission condition. In the case of the CDR equation and using continuous interpolations, this is the continuity of the diffusive fluxes.
- The essential approximation to make the problem computationally viable is to compute the subscales in the element interiors without taking into account their values on the boundaries.
- In the stability analysis presented, the subscales have their own "personality". They appear explicitly in the stability estimate. The final stability estimate for the finite element unknown depends on the way the subscales are approximated (that is to say, on how V' is chosen).

- It is observed that the expression of τ in terms of $\boldsymbol{a}$ is not used in the stability analysis. However, it is required in the convergence analysis.
- The only thing we have shown is that the terms introduced by the boundary contribution from the subscales can be controlled, but there seems to be no gain in considering the subscales on the boundaries. The stability estimate (6.47) is the same that would be obtained without the last term in the LHS of (6.40) which is, as it has been said, the main novelty of our proposal.

As stated in the last item, subscales on the boundary do not improve stability for the CDR equation. This is not so for the Stokes problem analyzed next. △

6.3 Stokes problem

6.3.1 Problem statement and finite element approximation

In this section we turn our attention to the Stokes problem, which consists of finding a velocity $\boldsymbol{u} : \Omega \longrightarrow \mathbb{R}^d$ and a pressure $p : \Omega \longrightarrow \mathbb{R}$ such that

$$\begin{aligned} -\nu\Delta\boldsymbol{u} + \nabla p &= \boldsymbol{f} \quad \text{in } \Omega \subset \mathbb{R}^d, \\ \nabla \cdot \boldsymbol{u} &= 0 \quad \text{in } \Omega, \\ \boldsymbol{u} &= \boldsymbol{0} \quad \text{on } \partial\Omega. \end{aligned}$$

The purpose is to extend the ideas of the previous section to this problem.

Let now $V = H_0^1(\Omega)^d$, $Q = L^2(\Omega)/\mathbb{R}$. The variational problem consists of finding $[\boldsymbol{u}, p] \in V \times Q$ such that

$$B([\boldsymbol{u}, p], [\boldsymbol{v}, q]) := \nu(\nabla\boldsymbol{u}, \nabla\boldsymbol{v}) - (p, \nabla \cdot \boldsymbol{v}) + (q, \nabla \cdot \boldsymbol{u}) = \langle \boldsymbol{f}, \boldsymbol{v} \rangle \quad \forall [\boldsymbol{v}, q] \in V \times Q.$$

For the sake of simplicity, we will consider subscales only for the velocity, not for the pressure. Pressure subscales can be easily introduced (see [32]), but they do not contribute to the present discussion. It is also possible to derive a general framework as in the previous section, using the trace of the velocity subscales and their fluxes as additional variables, leading to a five field formulation, the five fields being velocity, velocity subscale, trace of velocity subscale, flux of velocity subscale and pressure. However, we may directly work with velocity, velocity subscale and pressure, understanding that the velocity subscale on the interelement boundaries (and its test function) will be approximated independently, being single valued on these boundaries.

If $V_h \times Q_h \subset V \times Q$ is a conforming finite element approximation and V' is the space for the velocity subscales, the discrete variational problem to be considered is to find $[\boldsymbol{u}_h, p_h] \in V_h \times Q_h$ and $\boldsymbol{u}' \in V'$ such that

$$B([\boldsymbol{u}_h, p_h], [\boldsymbol{v}_h, q_h]) + \sum_K \langle \boldsymbol{u}', -\nu\Delta\boldsymbol{v}_h - \nabla q_h \rangle_K + \sum_K \langle \boldsymbol{u}', \nu\partial_n\boldsymbol{v}_h + q_h\boldsymbol{n} \rangle_{\partial K} = \langle \boldsymbol{f}, \boldsymbol{v}_h \rangle,$$

$$\sum_K \langle \nu(\partial_n\boldsymbol{u}_h + \partial_n\boldsymbol{u}') - p_h\boldsymbol{n}, \boldsymbol{v}' \rangle_{\partial K} + \sum_K \langle -\nu\Delta\boldsymbol{u}_h + \nabla p_h, \boldsymbol{v}' \rangle_K + \sum_K \langle -\nu\Delta\boldsymbol{u}', \boldsymbol{v}' \rangle_K = \langle \boldsymbol{f}, \boldsymbol{v}' \rangle,$$

which must hold for all $[\boldsymbol{v}_h, q_h] \in V_h \times Q_h$ and all $\boldsymbol{v}' \in V'$. The first term in the second discrete variational equation must be zero because of the (weak) continuity of the stress normal to the element boundaries (recall that $\boldsymbol{v}'$ has to be considered single valued when evaluated at the interelement boundaries).

As for the CDR equation, the approximation

$$\langle -\nu\Delta\boldsymbol{u}', \boldsymbol{v}'\rangle_K = \tau^{-1}\langle \boldsymbol{u}', \boldsymbol{v}'\rangle_K, \quad \tau^{-1} = C_1\frac{\nu}{h^2} \tag{6.48}$$

is adopted. Likewise, the subscale on the boundary will be approximated by an expression $\boldsymbol{u}'_E$ to be determined, so that the problem to be solved is to find $[\boldsymbol{u}_h, p_h] \in V_h \times Q_h$ and $\boldsymbol{u}' \in V'$ such that

$$B([\boldsymbol{u}_h, p_h], [\boldsymbol{v}_h, q_h]) + \sum_K \langle \boldsymbol{u}', -\nu\Delta\boldsymbol{v}_h - \nabla q_h\rangle_K + \sum_K \langle \boldsymbol{u}'_E, \nu\partial_n\boldsymbol{v}_h + q_h\boldsymbol{n}\rangle_{\partial K} = \langle \boldsymbol{f}, \boldsymbol{v}_h\rangle, \tag{6.49}$$

$$\sum_K \langle -\nu\Delta\boldsymbol{u}_h + \nabla p_h, \boldsymbol{v}'\rangle_K + \sum_K \tau^{-1}\langle \boldsymbol{u}', \boldsymbol{v}'\rangle_K = \langle \boldsymbol{f}, \boldsymbol{v}'\rangle, \tag{6.50}$$

which must hold for all $[\boldsymbol{v}_h, q_h] \in V_h \times Q_h$ and all $\boldsymbol{v}' \in V'$. The expression of τ is given in (6.48), but $\boldsymbol{u}'_E$ is required to close the problem.

6.3.2 Subscales on the element boundaries

The condition to determine the expression of the subscale velocity on the boundary is that the normal component of the stress be continuous across interelement boundaries. Using the same notation as in the previous section, this can be written as follows:

$$\begin{aligned}
\mathbf{0} &= [\![-p\boldsymbol{n} + \nu\partial_n\boldsymbol{u}]\!]_E \\
&= [\![-p_h\boldsymbol{n} + \nu\partial_n\boldsymbol{u}_h]\!]_E + [\![\nu\partial_n\boldsymbol{u}']\!]_E \\
&= [\![-p_h\boldsymbol{n} + \nu\partial_n\boldsymbol{u}_h]\!]_E + \frac{\nu}{\delta}\left(2\boldsymbol{u}'_E - \boldsymbol{u}'_1 - \boldsymbol{u}'_2\right),
\end{aligned}$$

from where the approximation we propose is

$$\boxed{\boldsymbol{u}'_E = \{\boldsymbol{u}'\}_E - \frac{\delta}{2\nu}[\![\nu\partial_n\boldsymbol{u}_h - p_h\boldsymbol{n}]\!]_E} \tag{6.51}$$

which is the counterpart of (6.36) for the Stokes problem.

Inserting (6.51) into the discrete variational problem (6.49)-(6.50) results in

$$B([\boldsymbol{u}_h, p_h], [\boldsymbol{v}_h, q_h]) + \sum_K \langle \boldsymbol{u}', -\nu\Delta\boldsymbol{v}_h - \nabla q_h\rangle_K + \sum_E \langle \{\boldsymbol{u}'\}, [\![\nu\partial_n\boldsymbol{v}_h + q_h\boldsymbol{n}]\!]\rangle_E$$
$$- \sum_E \frac{\delta}{2\nu}\langle [\![\nu\partial_n\boldsymbol{u}_h - p_h\boldsymbol{n}]\!], [\![\nu\partial_n\boldsymbol{v}_h + q_h\boldsymbol{n}]\!]\rangle_E = \langle \boldsymbol{f}, \boldsymbol{v}_h\rangle, \tag{6.52}$$

$$\sum_K \langle -\nu\Delta\boldsymbol{u}_h + \nabla p_h, \boldsymbol{v}'\rangle_K + \sum_K \tau^{-1}\langle \boldsymbol{u}', \boldsymbol{v}'\rangle_K = \langle \boldsymbol{f}, \boldsymbol{v}'\rangle, \tag{6.53}$$

which must hold for all $[\boldsymbol{v}_h, q_h] \in V_h \times Q_h$ and all $\boldsymbol{v}' \in V'$. This is the numerical approximation of the Stokes problem we propose and whose stability is analyzed next.

Remark 5 Note that if the term $\sum_E \langle \{\boldsymbol{u}'\}, [\![\nu \partial_n \boldsymbol{v}_h + q_h \boldsymbol{n}]\!] \rangle_E$ is neglected, the formulation *is symmetric*. To this end, the sign of q_h in the term $\langle [\![\nu \partial_n \boldsymbol{u}_h - p_h \boldsymbol{n}]\!], [\![\nu \partial_n \boldsymbol{v}_h + q_h \boldsymbol{n}]\!] \rangle_E$ is essential. On the other hand, it seems reasonable to neglect $\{\boldsymbol{u}'\}$ if discontinuous pressures are used because then the source of instability is known to be related to the lack of control on the pressure jumps. In particular, for the P_1/P_0 element used in the examples of Section 5, $\boldsymbol{u}' = \boldsymbol{0}$ in the element interiors. △

Remark 6 (Neumann boundary conditions) Suppose again that $F_K = \partial K \cap \partial\Omega$ and that the Neumann condition $-p\boldsymbol{n} + \nu \partial_n \boldsymbol{u} = \boldsymbol{t}$ is prescribed. The subscale $\boldsymbol{u}'_{F_K}$ should be computed from

$$\boldsymbol{t} = -p_h \boldsymbol{n} + \nu \partial_n \boldsymbol{u}_h + \frac{\nu}{\delta}(\boldsymbol{u}'_{F_K} - \boldsymbol{u}'_K).$$

In this case, the terms

$$\sum_K \left(\langle \boldsymbol{u}'_K, \nu \partial_n \boldsymbol{v}_h + q_h \boldsymbol{n} \rangle_{F_K} - \frac{\delta}{\nu} \langle \nu \partial_n \boldsymbol{u}_h - p_h \boldsymbol{n}, \nu \partial_n \boldsymbol{v}_h + q_h \boldsymbol{n} \rangle_{F_K} \right)$$

$$\text{and} \quad -\frac{\delta}{\nu} \sum_K \langle \boldsymbol{t}, \nu \partial_n \boldsymbol{v}_h + q_h \boldsymbol{n} \rangle_{F_K}$$

should be added to the LHS and right-hand-side (RHS) of (6.52), respectively. Stability on these boundaries will be enhanced by the term $\sum_K \frac{\delta}{\nu} \langle p_h, q_h \rangle_{F_K}$. This approach might be important as well in fluid-structure interaction problems, where one of the problems (the structure for example) is computed using the normal stresses $\boldsymbol{t}$ computed in the other domain. It is known that in some situations staggered coupled algorithms may suffer from the so called artificial mass effect due to the lack of stability in the imposition of the Neumann condition. △

6.3.3 Stability analysis

As for the CDR equation, it is convenient to define the expanded bilinear form of problem (6.52)-(6.53), including the subscales as unknowns, which is

$$\begin{aligned} B_{\text{exp}}([\boldsymbol{u}_h, p_h], \boldsymbol{u}'; [\boldsymbol{v}_h, q_h], \boldsymbol{v}') &= B([\boldsymbol{u}_h, p_h], [\boldsymbol{v}_h, q_h]) \\ &+ \sum_K \langle \boldsymbol{u}', -\nu \Delta \boldsymbol{v}_h - \nabla q_h \rangle_K + \sum_K \langle \boldsymbol{v}', -\nu \Delta \boldsymbol{u}_h + \nabla p_h \rangle_K \\ &+ \sum_E \langle \{\boldsymbol{u}'\}, [\![\nu \partial_n \boldsymbol{v}_h + q_h \boldsymbol{n}]\!] \rangle_E - \sum_E \frac{\delta}{2\nu} \langle [\![\nu \partial_n \boldsymbol{u}_h - p_h \boldsymbol{n}]\!], [\![\nu \partial_n \boldsymbol{v}_h + q_h \boldsymbol{n}]\!] \rangle_E \\ &+ \sum_K \tau^{-1} \langle \boldsymbol{u}', \boldsymbol{v}' \rangle_K. \end{aligned}$$

Taking $[\boldsymbol{v}_h, q_h] = [\boldsymbol{u}_h, p_h]$ and $\boldsymbol{v}' = \boldsymbol{u}'$ it follows that

$$\begin{aligned} B_{\exp}([\boldsymbol{u}_h, p_h], \boldsymbol{u}'; [\boldsymbol{u}_h, p_h], \boldsymbol{u}') &= \nu \|\nabla \boldsymbol{u}_h\|^2 \\ &+ \sum_K \langle \boldsymbol{u}', -2\nu\Delta \boldsymbol{u}_h \rangle_K + \sum_E \langle \{\boldsymbol{u}'\}, [\![\nu \partial_n \boldsymbol{u}_h + p_h \boldsymbol{n}]\!] \rangle_E \\ &- \sum_E \frac{\delta}{2} \nu \| [\![\partial_n \boldsymbol{u}_h]\!] \|_E^2 + \sum_E \frac{\delta}{2\nu} \| [\![\boldsymbol{n} p_h]\!] \|_E^2 + \sum_K \tau^{-1} \|\boldsymbol{u}'\|_K^2. \end{aligned}$$

We may deal with the terms

$$\sum_K \langle \boldsymbol{u}', -2\nu\Delta \boldsymbol{u}_h \rangle_K, \quad -\sum_E \frac{\delta}{2} \nu \| [\![\partial_n \boldsymbol{u}_h]\!] \|_E^2, \quad \sum_E \langle \{\boldsymbol{u}'\}, [\![\nu \partial_n \boldsymbol{u}_h]\!] \rangle_E,$$

exactly as for the CDR equation. It only remains the following bound:

$$\begin{aligned} \sum_E \langle \{\boldsymbol{u}'\}, [\![p_h \boldsymbol{n}]\!] \rangle_E &\geq -\sum_E \|\{\boldsymbol{u}'\}\|_E \| [\![p_h \boldsymbol{n}]\!] \|_E \\ &\geq -\sum_E \left(\frac{\beta\nu}{2\delta} \|\{\boldsymbol{u}'\}\|_E^2 + \frac{\delta}{2\beta\nu} \| [\![\boldsymbol{n} p_h]\!] \|_E^2 \right) \\ &\geq -\sum_K \frac{\beta\nu}{2\delta} \|\boldsymbol{u}'\|_{\partial K}^2 - \sum_E \frac{\delta}{2\beta\nu} \| [\![\boldsymbol{n} p_h]\!] \|_E^2 \\ &\geq -\sum_K \frac{\beta\nu}{2\delta_0 h^2} C_{\mathrm{tr}} \|\boldsymbol{u}'\|_K^2 - \sum_E \frac{\delta}{2\beta\nu} \| [\![\boldsymbol{n} p_h]\!] \|_E^2, \end{aligned}$$

which holds for all $\beta > 0$. Taking it sufficiently large ($\beta > 1$) and proceeding exactly as for the CDR equation we obtain:

Theorem 2 *There are constants C_1 and δ_0 in the definition of the stabilization parameters such that*

$$B_{\exp}([\boldsymbol{u}_h, p_h], \boldsymbol{u}'; [\boldsymbol{u}_h, p_h], \boldsymbol{u}') \geq C \left(\nu \|\nabla \boldsymbol{u}_h\|^2 + \sum_E \frac{\delta}{\nu} \| [\![\boldsymbol{n} p_h]\!] \|_E^2 + \sum_K \tau^{-1} \|\boldsymbol{u}'\|_K^2 \right).$$

Remark 7 In the previous estimate, it is important to note that

- Contrary to the CDR equation, now there is *a clear gain* by accounting for the subscales on the boundary: we have control on the pressure jumps over interelement boundaries. This in particular stabilizes elements with discontinuous pressures.
- Control over $\| [\![\boldsymbol{n} p_h]\!] \|_E^2$ can be transformed into L^2 control over p_h. This can be proved for example using the strategy presented in [34] and in references therein.

The stability estimate obtained is clearly optimal. △

6.4 Darcy flow

6.4.1 Problem statement and finite element approximation

We will consider here the simplest situation in which the permeability is isotropic and uniform. The problem to be solved consists in finding a velocity $\boldsymbol{u}$ and a pressure p such that

$$\begin{aligned} \kappa^{-1}\boldsymbol{u} + \nabla p &= \boldsymbol{0} \qquad \text{in } \Omega, \\ \nabla \cdot \boldsymbol{u} &= f \qquad \text{in } \Omega, \\ \boldsymbol{u} \cdot \boldsymbol{n} &= 0 \qquad \text{on } \partial\Omega, \end{aligned}$$

where κ is the permeability coefficient. The functional spaces where the problem *can* be posed are

$$V = H_0(\mathrm{div}, \Omega),\ Q = L^2(\Omega)/\mathbb{R},$$

for the velocity and the pressure, respectively. In this case, $f \in L^2(\Omega)$. The classical variational formulation of the Darcy problem is well posed in these spaces. However, it is observed from the momentum equation that in fact the pressure will belong to $H^1(\Omega)/\mathbb{R}$.

The weak form of the problem is

$$\begin{aligned} (\kappa^{-1}\boldsymbol{u}, \boldsymbol{v}) - (p, \nabla \cdot \boldsymbol{v}) &= 0, \\ (q, \nabla \cdot \boldsymbol{u}) &= (q, f), \end{aligned}$$

which must hold for all $[\boldsymbol{v}, q] \in V \times Q$.

As in the previous section, the finite element spaces for velocity and pressure will be respectively denoted by $V_h \subset V$, $Q_h \subset Q$ (conforming approximations will be considered). If we consider as before the scale splitting

$$\begin{aligned} \boldsymbol{u} &= \boldsymbol{u}_h + \boldsymbol{u}', \quad \boldsymbol{u}_h \in V_h,\ \boldsymbol{u}' \in V', \\ p &= p_h + p', \quad p_h \in Q_h,\ p' \in Q', \end{aligned}$$

with spaces V' and Q' for the moment undefined, the problem to be solved becomes

$$(\kappa^{-1}\boldsymbol{u}_h, \boldsymbol{v}_h) + (\kappa^{-1}\boldsymbol{u}', \boldsymbol{v}_h) - (p_h, \nabla \cdot \boldsymbol{v}_h) - (p', \nabla \cdot \boldsymbol{v}_h) = 0 \qquad \forall\, \boldsymbol{v}_h \in V_h, \tag{6.54}$$

$$(q_h, \nabla \cdot \boldsymbol{u}_h) - \sum_K (\boldsymbol{u}', \nabla q_h)_K + \sum_K (q_h, \boldsymbol{n} \cdot \boldsymbol{u}')_{\partial K} = (q_h, f) \qquad \forall\, q_h \in Q_h, \tag{6.55}$$

together with the equations obtained by testing the differential equations with the velocity and pressure subscale test functions.

In this case, we need to deal both with a velocity and with a pressure subscale, which makes the derivation of a closed form for them more involved than for the problems of sections 6.2 and 6.3. This can be done in a similar way as for the Stokes problem in [34]. If $P_{V'}$ and $P_{Q'}$ denote the L^2-projection onto V' and Q', respectively, the final result is that $\boldsymbol{u}'$ and p' can be approximated in the element interiors by

$$\begin{aligned} \boldsymbol{u}' &= -P_{V'}\left(\boldsymbol{u}_h + \kappa \nabla p_h\right), \\ p' &= \tau_p P_{Q'}\left(f - \nabla \cdot \boldsymbol{u}_h\right), \end{aligned}$$

where the stabilization parameter τ_p is given by

$$\tau_p = C_p \frac{h^2}{\kappa}, \tag{6.56}$$

C_p being an algorithmic constant. The main idea to obtain this approximation is to approximate the Darcy operator in the equation for the subscales by a matrix $\mathrm{diag}(\tau_u^{-1}\boldsymbol{I}, \tau_p^{-1})$, where $\boldsymbol{I}$ is the $d \times d$ identity. Using an approximate Fourier analysis it can be shown that the norm of this matrix is an approximate upper bound to the norm of the Darcy operator if $\tau_u = 1$ and τ_p is given by (6.56) (see [34] for details about this approach).

It is convenient to write the previous approximation in 'weak' form as follows:

$$\begin{aligned}(\kappa^{-1}\boldsymbol{u}', \boldsymbol{v}') + (\kappa^{-1}\boldsymbol{u}_h, \boldsymbol{v}') + \sum_K (\nabla p_h, \boldsymbol{v}') = 0 \qquad & \forall \boldsymbol{v}' \in V', \\ (q', \nabla \cdot \boldsymbol{u}_h) + \sum_K \tau_p^{-1}(p', q') = (q', f) \quad & \forall q' \in Q'.\end{aligned}$$

6.4.2 Subscales on the element boundaries

The transmission conditions for this problem are different from those of the Stokes problem of the previous section. First of all, observe that

- Only the velocity subscale is needed on the boundary of the elements (see (6.54)-(6.55)).
- For each element, this velocity subscale can be computed from the pressure subscale on the boundary by projecting the momentum equation.
- Since in fact $p \in H^1(\Omega)$, p must be such that

$$[\![\, \boldsymbol{n} p \,]\!]_E = \boldsymbol{0}, \quad [\![\, \partial_n p \,]\!]_E = 0. \tag{6.57}$$

Equations (6.57) are the transmission conditions that have to allow us to compute the subscales on the element boundaries. Since the pressure is allowed to be discontinuous across these interelement boundaries, the pressure subscale must also be allowed to be discontinuous. Let us denote by p_{h,E_i} the pressure finite element function on an edge E from the side of K_i (see again Fig. 6.1) and p'_{E_i} the corresponding subscale. Pressure continuity across E implies

$$[\![\, \boldsymbol{n} p \,]\!]_E = (p_{h,E_1} + p'_{E_1})\boldsymbol{n}_1 + (p_{h,E_2} + p'_{E_2})\boldsymbol{n}_2 = \boldsymbol{0},$$

from where

$$p'_{E_1} - p'_{E_2} = -p_{h,E_1} + p_{h,E_2} = -[\![\, \boldsymbol{n} p_h \,]\!]_E \cdot \boldsymbol{n}_1. \tag{6.58}$$

Using an approximation for the derivatives of the subscales similar to that of the previous sections, continuity of the pressure normal derivative implies:

$$\begin{aligned}0 &= [\![\, \partial_n p_h \,]\!]_E + [\![\, \partial_n p' \,]\!]_E \\ &= [\![\, \partial_n p_h \,]\!]_E + \frac{1}{\delta}\left(p'_{E_1} - p'_{K_1}\right) + \frac{1}{\delta}\left(p'_{E_2} - p'_{K_2}\right),\end{aligned}$$

from where

$$p'_{E_1} + p'_{E_2} = p'_{K_1} + p'_{K_2} - \delta \, [\![\, \partial_n p_h \,]\!]_E. \tag{6.59}$$

The solution of system (6.58)-(6.59) yields

$$\boxed{p'_{\partial K} = \{p'_K\}_{\partial K} - \frac{\delta}{2} [\![\, \partial_n p_h \,]\!]_{\partial K} - \frac{1}{2} [\![\, \boldsymbol{n} p_h \,]\!]_{\partial K} \cdot \boldsymbol{n}} \tag{6.60}$$

Equation (6.60) is the expression of the pressure subscale on the element edges (now discontinuous), obtained from the application of our ideas to the Darcy problem. However, as mentioned earlier, this expression is only required to compute the velocity subscales on the edges, again considering them discontinuous. Projecting the momentum equation on the element boundaries we have:

$$\begin{aligned}
\boldsymbol{n} \cdot \boldsymbol{u}'|_{\partial K} &= -\boldsymbol{n} \cdot \boldsymbol{u}_h|_{\partial K} - \kappa \partial_n p_h|_{\partial K} - \kappa \partial_n p'|_{\partial K} \\
&= -\boldsymbol{n} \cdot \boldsymbol{u}_h|_{\partial K} - \kappa \partial_n p_h|_{\partial K} - \frac{\kappa}{\delta}(p'_{\partial K} - p'_K) \\
&= -\boldsymbol{n} \cdot \boldsymbol{u}_h|_{\partial K} - \kappa \partial_n p_h|_{\partial K} - \frac{\kappa}{\delta} \left[\{p'_K\}_{\partial K} - \frac{\delta}{2} [\![\, \partial_n p_h \,]\!]_{\partial K} - \frac{1}{2} [\![\, \boldsymbol{n} p_h \,]\!]_{\partial K} \cdot \boldsymbol{n} - p'_K \right] \\
&= -\boldsymbol{n} \cdot \boldsymbol{u}_h|_{\partial K} - \kappa \partial_n p_h|_{\partial K} + \frac{\kappa}{2} [\![\, \partial_n p_h \,]\!]_{\partial K} + \frac{\kappa}{2\delta} [\![\, \boldsymbol{n}(p_h + p'_K) \,]\!]_{\partial K} \cdot \boldsymbol{n},
\end{aligned}$$

from where we obtain the expression for the velocity subscale on ∂K:

$$\boxed{\boldsymbol{n} \cdot \boldsymbol{u}'|_{\partial K} = -\boldsymbol{n} \cdot \boldsymbol{u}_h|_{\partial K} - \kappa \{\partial_n p_h\}|_{\partial K} + \frac{\kappa}{2\delta} [\![\, \boldsymbol{n}(p_h + p'_K) \,]\!]_{\partial K} \cdot \boldsymbol{n}} \tag{6.61}$$

Since no velocity derivatives appear in the transmission conditions for this problem, the velocity subscale on ∂K turns out to be independent from the velocity subscale on K.

Note now that all the terms on the RHS of (6.61) are vectors whose normal component is continuous across interelement boundaries (the first because we assume $V_h \subset V$). If $\boldsymbol{w}$ is a vector defined on E, with continuous normal component, it holds that

$$\sum_K \langle q_h, \boldsymbol{n} \cdot \boldsymbol{w} \rangle_{\partial K} = \sum_E \langle [\![\, \boldsymbol{n} q_h \,]\!], \boldsymbol{w} \rangle_E.$$

Using this in the finite element approximation for the continuity equation we obtain the final problem to be solved, which consists of finding $\boldsymbol{u}_h \in V_h$, $p_h \in Q_h$, $\boldsymbol{u}' \in V'$ and $p' \in Q'$ such

that

$$(\kappa^{-1}\boldsymbol{u}_h, \boldsymbol{v}_h) + (\kappa^{-1}\boldsymbol{u}', \boldsymbol{v}_h) - (p_h, \nabla\cdot\boldsymbol{v}_h) - (p', \nabla\cdot\boldsymbol{v}_h) = 0 \quad \forall\, \boldsymbol{v}_h \in V_h, \tag{6.62}$$

$$\begin{aligned}(q_h, \nabla\cdot\boldsymbol{u}_h) &- \sum_K (\boldsymbol{u}', \nabla q_h)_K \\ &- \sum_E \left\langle [\![\, \boldsymbol{n} q_h \,]\!], \boldsymbol{u}_h + \{\kappa\nabla p_h\} - \frac{\kappa}{2\delta}[\![\, \boldsymbol{n} p' \,]\!] \right\rangle_E \\ &+ \sum_E \frac{\kappa}{2\delta}\langle [\![\, \boldsymbol{n} q_h \,]\!], [\![\, \boldsymbol{n} p_h \,]\!]\rangle_E = (q_h, f) \qquad \forall\, q_h \in Q_h,\end{aligned} \tag{6.63}$$

$$(\kappa^{-1}\boldsymbol{u}', \boldsymbol{v}') + (\kappa^{-1}\boldsymbol{u}_h, \boldsymbol{v}') + \sum_K (\nabla p_h, \boldsymbol{v}') = 0 \qquad \forall \boldsymbol{v}' \in V', \tag{6.64}$$

$$(q', \nabla\cdot\boldsymbol{u}_h) + \sum_K \tau_p^{-1}(p', q') = (q', f) \qquad \forall q' \in Q', \tag{6.65}$$

with τ_p given by (6.56).

6.4.3 Stability analysis

The previous problem can be written as

$$B_{\text{exp}}(\boldsymbol{u}_h, p_h, \boldsymbol{u}', p'; \boldsymbol{v}_h, q_h, \boldsymbol{v}', q') = (q_h, f) + (q', f),$$

with the obvious definition for the bilinear form B_{exp}. The stability analysis in this case is a bit more delicate than for the CDR equation and for the Stokes problem. The problem is that B_{exp} is not coercive, but satisfies an inf-sup condition in a norm to be introduced in the following.

We assume that the decomposition $V_h \oplus V'$ is L^2-stable, in the sense that for any functions $\boldsymbol{v}_h \in V_h$ and $\boldsymbol{v}' \in V'$ we have

$$\|\boldsymbol{v}_h + \boldsymbol{v}'\|^2 \geq C_{\text{dec}} \left(\|\boldsymbol{v}_h\|^2 + \|\boldsymbol{v}'\|^2 \right), \tag{6.66}$$

for a constant C_{dec} independent of the equation parameters and of the mesh size. In general, $C_{\text{dec}} \leq 1$ and if V' is taken L^2-orthogonal to V_h, $C_{\text{dec}} = 1$.

Let $\boldsymbol{U}_h = [\boldsymbol{u}_h, p_h, \boldsymbol{u}', p']$ be the unknown of the problem and $\boldsymbol{V}_h = [\boldsymbol{v}_h, q_h, \boldsymbol{v}', q']$ the corresponding vector of test functions. Let also

$$\begin{aligned}|\!|\!|\boldsymbol{U}_h|\!|\!|^2 :=& \kappa^{-1}\|\boldsymbol{u}_h\|^2 + \sum_E \frac{\kappa}{\delta}\|[\![\, \boldsymbol{n} p_h \,]\!]\|_E^2 + \sum_K \kappa^{-1}\|\boldsymbol{u}'\|_K^2 \\ &+ \sum_K \tau_p^{-1}\|p'\|_K^2 + \sum_K \kappa^{-1}\|P_{V_h}(\boldsymbol{u}_h + \kappa\nabla p_h)\|_K^2,\end{aligned}$$

where P_{V_h} is the L^2-projection onto V_h. However, later on we will introduce another norm in which stability holds and that clearly displays the stability enhancement we obtain with respect to the classical Galerkin method.

Let us start writing

$$\begin{aligned}\boldsymbol{u}_h + \kappa\nabla p_h &= P_{V_h}\left(\boldsymbol{u}_h + \kappa\nabla p_h\right) + P_{V'}\left(\boldsymbol{u}_h + \kappa\nabla p_h\right) \\ &:= \boldsymbol{m}_h - \boldsymbol{u}',\end{aligned}$$

which allows us to write

$$B_{\exp}(\boldsymbol{U}_h,\boldsymbol{U}_h) = \kappa^{-1}\|\boldsymbol{u}_h+\boldsymbol{u}'\|^2 + \sum_E \frac{\kappa}{2\delta}\|[\![\,\boldsymbol{n}p_h\,]\!]\|_E^2 + \sum_K \tau_p^{-1}\|p'\|_K^2 \\ - \sum_E \langle [\![\,\boldsymbol{n}p_h\,]\!], \boldsymbol{m}_h\rangle_E + \sum_E \langle [\![\,\boldsymbol{n}p_h\,]\!], \{\boldsymbol{u}'\}\rangle_E + \sum_E \frac{\kappa}{2\delta}\langle [\![\,\boldsymbol{n}p_h\,]\!], [\![\,\boldsymbol{n}p'\,]\!]\rangle_E. \tag{6.67}$$

The objective now is to bound the last three terms in the RHS of this equality. Let us start with the last one. Using (6.46) we have that

$$\begin{aligned}\sum_E \frac{\kappa}{2\delta}\langle [\![\,\boldsymbol{n}p_h\,]\!], [\![\,\boldsymbol{n}p'\,]\!]\rangle_E &\geq -\frac{\kappa}{2\delta}\sum_E\left(\frac{\beta_1}{2}\|[\![\,\boldsymbol{n}p_h\,]\!]\|_E^2 + \frac{1}{2\beta_1}\|[\![\,\boldsymbol{n}p'\,]\!]\|_E^2\right)\\ &\geq -\frac{\kappa}{2\delta}\frac{\beta_1}{2}\sum_E\|[\![\,\boldsymbol{n}p_h\,]\!]\|_E^2 - \frac{\kappa}{2\delta}\frac{1}{2\beta_1}\sum_K\|p'\|_{\partial K}^2\\ &\geq -\frac{\kappa}{2\delta}\frac{\beta_1}{2}\sum_E\|[\![\,\boldsymbol{n}p_h\,]\!]\|_E^2 - \frac{\kappa}{2\delta}\frac{1}{2\beta_1}C_{\mathrm{tr}}h^{-1}\sum_K\|p'\|_K^2\\ &\geq -\frac{\kappa}{2\delta}\frac{\beta_1}{2}\sum_E\|[\![\,\boldsymbol{n}p_h\,]\!]\|_E^2 - \frac{1}{4\delta_0\beta_1}C_{\mathrm{tr}}\frac{\kappa}{h^2}\sum_K\|p'\|_K^2.\end{aligned} \tag{6.68}$$

We also have that

$$\begin{aligned}\sum_E \langle [\![\,\boldsymbol{n}p_h\,]\!], \{\boldsymbol{u}'\}\rangle_E &\geq -\sum_E\left(\frac{\delta}{2\beta_2}\kappa^{-1}\|\{\boldsymbol{u}'\}\|_E^2 + \frac{\beta_2}{2\delta}\kappa\|[\![\,\boldsymbol{n}p_h\,]\!]\|_E^2\right)\\ &\geq -\sum_K\frac{\delta}{2\beta_2}\kappa^{-1}\|\boldsymbol{u}'\|_{\partial K}^2 - \sum_E\frac{\beta_2}{2\delta}\kappa\|[\![\,\boldsymbol{n}p_h\,]\!]\|_E^2\\ &\geq -\sum_K\frac{\delta_0}{2\beta_2}C_{\mathrm{tr}}\kappa^{-1}\|\boldsymbol{u}'\|_K^2 - \sum_E\frac{\beta_2}{2\delta}\kappa\|[\![\,\boldsymbol{n}p_h\,]\!]\|_E^2.\end{aligned} \tag{6.69}$$

Using (6.66), (6.68) and (6.69) in (6.67) we have

$$B_{\exp}(\boldsymbol{U}_h,\boldsymbol{U}_h) \geq \kappa^{-1}C_{\mathrm{dec}}\|\boldsymbol{u}_h\|^2 + \kappa^{-1}\left(C_{\mathrm{dec}} - \frac{\delta_0}{2\beta_2}C_{\mathrm{tr}}\right)\sum_K\|\boldsymbol{u}'\|_K^2 \\ + \sum_E\frac{\kappa}{2\delta}\left(1-\frac{\beta_1}{2}-\beta_2\right)\|[\![\,\boldsymbol{n}p_h\,]\!]\|_E^2 + \sum_K\frac{\kappa}{h^2}\left(\frac{1}{C_p}-\frac{C_{\mathrm{tr}}}{4\delta_0\beta_1}\right)\|p'\|_K^2 \\ - \sum_E\langle [\![\,\boldsymbol{n}p_h\,]\!], \boldsymbol{m}_h\rangle_E. \tag{6.70}$$

It remains to control the last term. It is responsible for the fact that the bilinear form $B_{\exp}$ is not coercive, but it only satisfies an inf-sup condition. By the definition of $\boldsymbol{m}_h$ and using (6.45)

we have that

$$\begin{aligned}
&B_{\exp}(\boldsymbol{U}_h,[\boldsymbol{m}_h,0,\mathbf{0},0])\\
&\quad=(\kappa^{-1}\boldsymbol{u}_h+\nabla p_h,\boldsymbol{m}_h)+(\kappa^{-1}\boldsymbol{u}',\boldsymbol{m}_h)-(p',\nabla\cdot\boldsymbol{m}_h)-\sum_K\langle p_h,\boldsymbol{n}\cdot\boldsymbol{m}_h\rangle_{\partial K}\\
&\quad=\kappa^{-1}(\boldsymbol{m}_h,\boldsymbol{m}_h)-(p',\nabla\cdot\boldsymbol{m}_h)-\sum_E\langle[\![\,\boldsymbol{n}p_h\,]\!],\boldsymbol{m}_h\rangle_E\\
&\quad\geq\kappa^{-1}\|\boldsymbol{m}_h\|^2-\|p'\|\frac{C_{\mathrm{inv}}}{h}\|\boldsymbol{m}_h\|-\sum_E\langle[\![\,\boldsymbol{n}p_h\,]\!],\boldsymbol{m}_h\rangle_E\\
&\quad\geq\kappa^{-1}\|\boldsymbol{m}_h\|^2-\frac{1}{2\beta_3}\frac{\kappa}{h^2}C_{\mathrm{inv}}^2\|p'\|^2-\frac{\beta_3}{2}\kappa^{-1}\|\boldsymbol{m}_h\|^2-\sum_E\langle[\![\,\boldsymbol{n}p_h\,]\!],\boldsymbol{m}_h\rangle_E,
\end{aligned}$$

which combined with (6.70) yields

$$\begin{aligned}
&B_{\exp}(\boldsymbol{U}_h,\boldsymbol{U}_h+[\boldsymbol{m}_h,0,\mathbf{0},0])\\
&\quad\geq\kappa^{-1}C_{\mathrm{dec}}\|\boldsymbol{u}_h\|^2+\kappa^{-1}\left(C_{\mathrm{dec}}-\frac{\delta_0}{2\beta_2}C_{\mathrm{tr}}\right)\|\boldsymbol{u}'\|^2\\
&\quad+\sum_E\frac{\kappa}{2\delta}\left(1-\frac{\beta_1}{2}-\beta_2\right)\|[\![\,\boldsymbol{n}p_h\,]\!]\|_E^2+\sum_K\frac{\kappa}{h^2}\left(\frac{1}{C_p}-\frac{C_{\mathrm{tr}}}{4\delta_0\beta_1}-\frac{C_{\mathrm{inv}}^2}{2\beta_3}\right)\|p'\|_K^2\\
&\quad+\kappa^{-1}\left(1-\frac{\beta_3}{2}\right)\|\boldsymbol{m}_h\|^2-2\sum_E\langle[\![\,\boldsymbol{n}p_h\,]\!],\boldsymbol{m}_h\rangle_E.
\end{aligned}\tag{6.71}$$

On the other hand

$$\begin{aligned}
-2\sum_E\langle[\![\,\boldsymbol{n}p_h\,]\!],\boldsymbol{m}_h\rangle_E&\geq-\sum_E\beta_4\frac{\kappa}{2\delta}\|[\![\,\boldsymbol{n}p_h\,]\!]\|_E^2-\sum_E\frac{1}{\beta_4}\frac{2\delta}{\kappa}\|\boldsymbol{m}_h\|_E^2\\
&\geq-\sum_E\beta_4\frac{\kappa}{2\delta}\|[\![\,\boldsymbol{n}p_h\,]\!]\|_E^2-\sum_K\frac{1}{\beta_4}\kappa^{-1}\delta_0C_{\mathrm{tr}}\|\boldsymbol{m}_h\|_K^2,
\end{aligned}$$

which used in (6.71) gives

$$\begin{aligned}
&B_{\exp}(\boldsymbol{U}_h,\boldsymbol{U}_h+[\boldsymbol{m}_h,0,\mathbf{0},0])\\
&\quad\geq\kappa^{-1}C_{\mathrm{dec}}\|\boldsymbol{u}_h\|^2+\kappa^{-1}\left(C_{\mathrm{dec}}-\frac{\delta_0}{2\beta_2}C_{\mathrm{tr}}\right)\|\boldsymbol{u}'\|^2\\
&\quad+\sum_E\frac{\kappa}{2\delta}\left(1-\frac{\beta_1}{2}-\beta_2-\beta_4\right)\|[\![\,\boldsymbol{n}p_h\,]\!]\|_E^2+\sum_K\frac{\kappa}{h^2}\left(\frac{1}{C_p}-\frac{C_{\mathrm{tr}}}{4\delta_0\beta_1}-\frac{C_{\mathrm{inv}}^2}{2\beta_3}\right)\|p'\|_K^2\\
&\quad+\kappa^{-1}\left(1-\frac{\beta_3}{2}-\frac{\delta_0C_{\mathrm{tr}}}{\beta_4}\right)\|\boldsymbol{m}_h\|^2.
\end{aligned}$$

From this expression we see that if we take β_i, $i=1,2,3,4$, sufficiently small, then there exists a constant C for which

$$B_{\exp}(\boldsymbol{U}_h,\boldsymbol{U}_h+[\boldsymbol{m}_h,0,\mathbf{0},0])\geq C|\!|\!|\boldsymbol{U}_h|\!|\!|^2,\tag{6.72}$$

provided the constants δ_0 and C_p are small enough. On the other hand, $|||[\boldsymbol{m}_h, 0, \mathbf{0}, 0]||| \leq C|||\boldsymbol{U}_h|||$, from where we obtain the result we wished to prove:

Theorem 3 *There are constants C_p and δ_0 in the definition of the stabilization parameters such that for all $\boldsymbol{U}_h$ there exists $\boldsymbol{V}_h$ such that*

$$B_{\exp}(\boldsymbol{U}_h, \boldsymbol{V}_h) \geq C|||\boldsymbol{U}_h|||\,|||\boldsymbol{V}_h|||.$$

Remark 8 Since $\boldsymbol{u}' = P_{V'}(\boldsymbol{u}_h + \kappa\nabla p_h)$ and in view of (6.66) our result also applies with the norm

$$|||\boldsymbol{U}_h|||_*^2 := \kappa^{-1}\|\boldsymbol{u}_h\|^2 + \kappa\sum_K \|\nabla p_h\|_K^2 + \sum_E \frac{\kappa}{\delta}\|[\![\,\boldsymbol{n}p_h\,]\!]\|_E^2 + \sum_K \tau_p^{-1}\|p'\|_K^2,$$

which allows us to see that the stability result of Theorem 3 is optimal. Moreover, from the expression of p' in the element interiors, usually proportional to the velocity divergence, it is possible to control $\|\nabla\cdot\boldsymbol{u}_h\|$ which, together with the stability obtained on $\|\boldsymbol{u}_h\|$, leads to full control of $\boldsymbol{u}_h$ in $H_0(\mathrm{div},\Omega)$. △

6.5 Numerical examples

In this section we present the results of some numerical examples in order to study the performance of the presented method. We compare the results obtained using the approximation of the subscales on the interelement boundaries u'_E given by (6.36) (or (6.51) in the case of the Stokes problem) with those obtained considering $u'_E = 0$. A parameter $\delta_0 = 0.2$ has been adopted for the computation of the terms corresponding to the subscales on the element boundaries, as it has proved to be suitable for these numerical examples, even though for the Stokes problem the effect of the choice of δ_0 has also been analyzed.

No results for the Darcy problem have been included, since in the case of interest, that is to say, for discontinuous pressure interpolations, the accuracy heavily relies on the expression of the subscales in the element interiors.

6.5.1 Convection-diffusion equation

Let us start solving the convection-diffusion equation. We consider a domain Ω enclosed in a circle of radius $R = 1$, which we discretize in a triangular finite element mesh, and we prescribe

$$u = 0 \qquad \text{on } \partial\Omega.$$

We now study two different cases: in the first one diffusion dominates over convection ($k = 0.1$, $\boldsymbol{a} = (1,0)$, $s = 0$, $f = 1$ in (6.1)), while the second one is convection dominated ($k = 10^{-12}$, $\boldsymbol{a} = (1,0)$, $s = 0$, $f = 1$ in (6.1)). In both the diffusion and the convection dominated cases, no difference between the solution obtained considering u'_E and the one obtained without considering it can be appreciated. Fig. 6.2 shows and compares the obtained solution u for the considered methods in the diffusion dominated case, while Fig. 6.3 does so when convection dominates over diffusion. In any case, there is no noticeable influence of the value of δ_0 on the results.

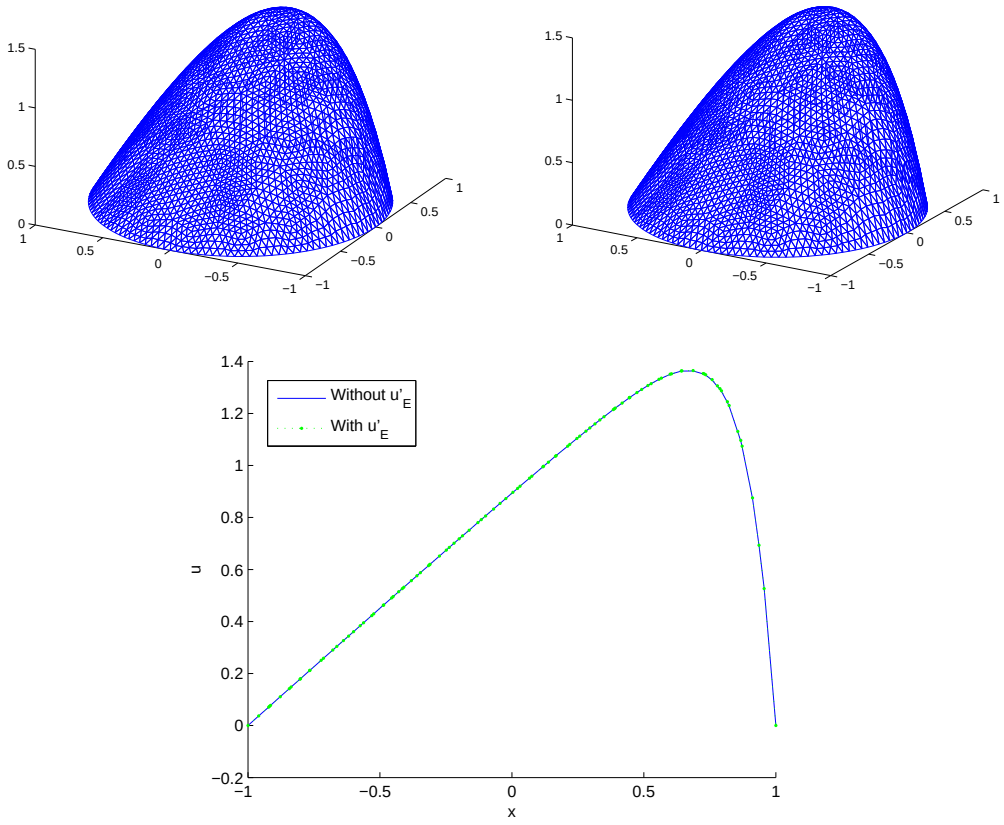


Figure 6.2: Elevations for the diffusion dominated problem without (left) and considering (right) u'_E. Cut along $y = 0$ (bottom).

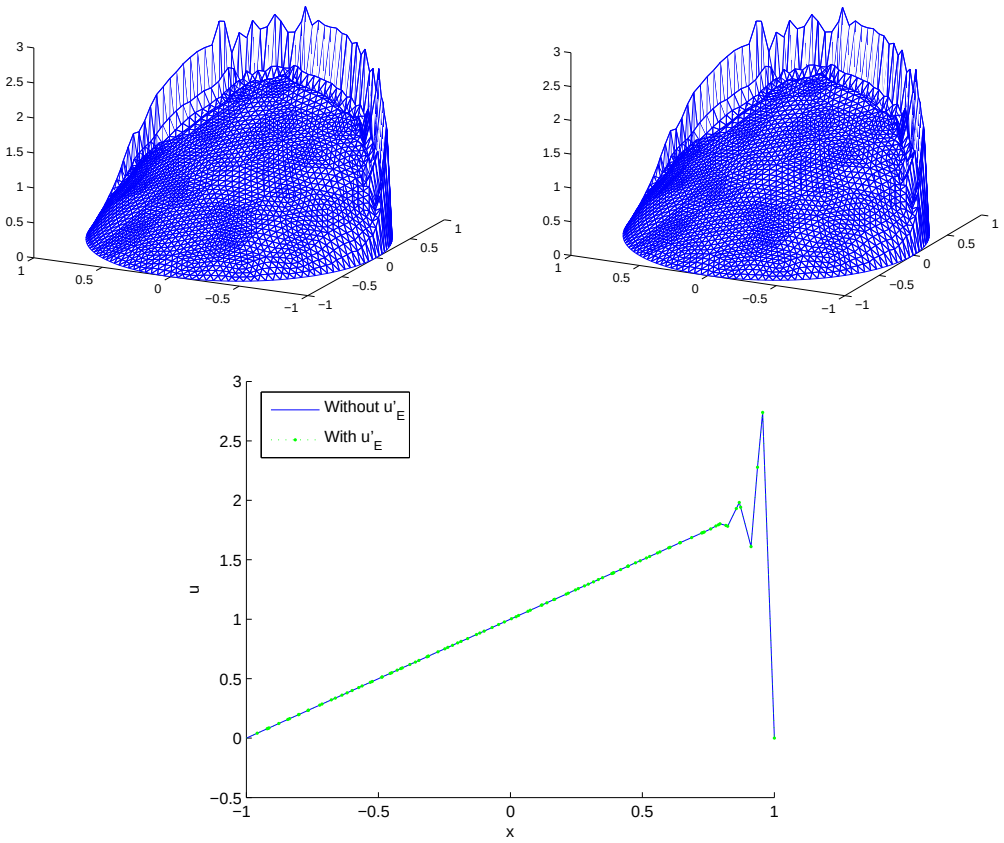


Figure 6.3: Elevations for the convection dominated problem without (left) and considering (right) u'_E. Cut along $y = 0$ (bottom).

6.5.2 Stokes problem

In this section we study the performance of the method proposed for the Stokes problem. As stated in Section 6.3, considering the contribution of the subscales in the element boundaries $\boldsymbol{u}'_E$ stabilizes elements with discontinuous pressures. In particular it allows the use of P_1/P_0 (linear-constant) velocity-pressure pairs. Results using P_1/P_0 interpolation and considering the contribution of the subscales on the boundary will be compared with those obtained using P_1/P_1 (linear-linear) velocity-pressure pairs, in which no subscales on the boundaries are considered.

Flow in a cavity

In this example, the motion of a fluid enclosed in a square cavity $\Omega = [0,1] \times [0,1]$ is analyzed. The velocity is set to $(1,0)$ at the top horizontal wall ($y = 1$), while it is prescribed to $\mathbf{0}$ on the other walls ($y = 0, x = 0$ and $x = 1$). Pressure is fixed to 0 in an arbitrary point of the domain.

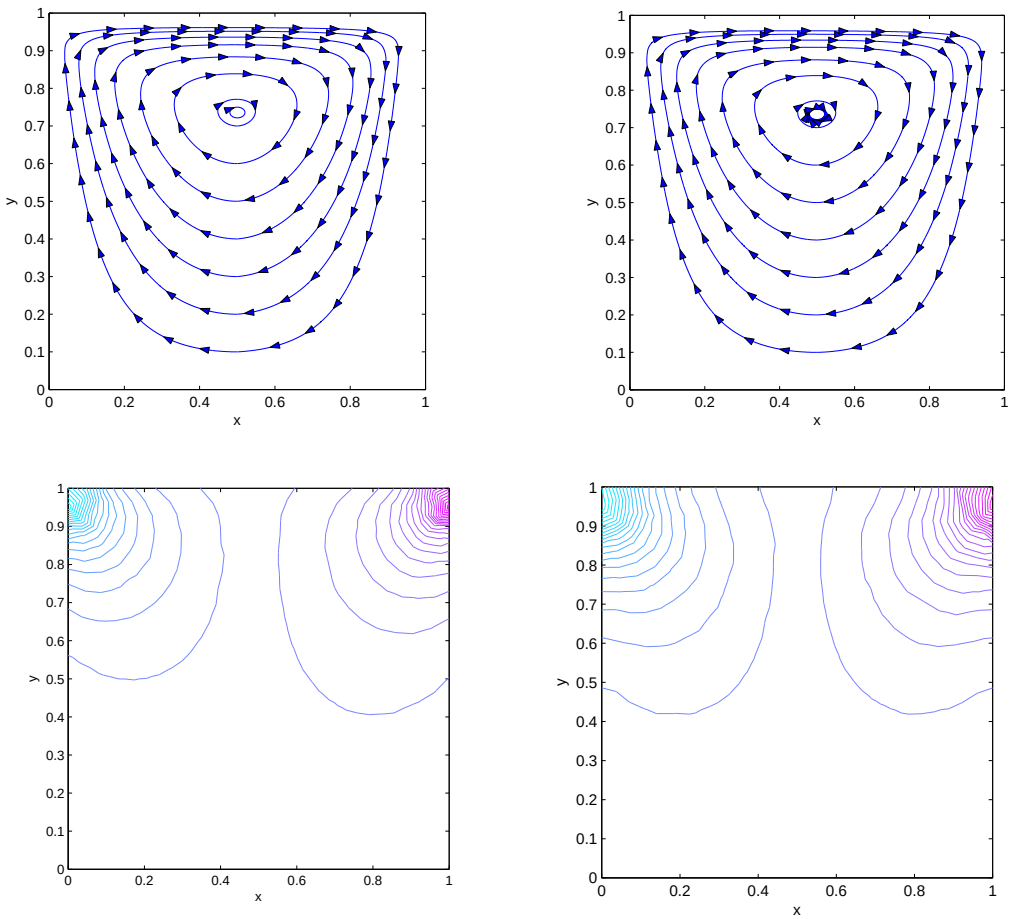


Figure 6.4: Results for the flow in a cavity. Left: P_1/P_1 interpolation (without $\boldsymbol{u}'_E$). Right: P_1/P_0 interpolation (with $\boldsymbol{u}'_E$). From top to bottom: streamlines and pressure contours.

As Fig. 6.4 shows, little difference can be observed between results obtained using P_1/P_1

interpolation and those obtained using P_1/P_0 and taking into account the contribution of the subscales on the element boundaries with $\delta_0 = 0.2$. The slight differences which can be observed between both results are due to the fact that a poorer interpolation space for the pressure is used in the second case.

In order to check the behavior of the solution in terms of δ_0, Fig. 6.5 shows a comparison between the pressure along $y = 1$ for $\delta_0 = 0.05$, 0.2 and 0.5. Note that this last value would be the maximum allowed by our way to motivate the subscales on the element boundaries (see Fig. 6.1). It is observed that $\delta_0 = 0.05$ allows for pressure oscillations, whereas no much difference is observed for $\delta_0 = 0.2$ and $\delta_0 = 0.5$ (in fact, similar results are obtained for any δ_0 greater than 0.1). Of course, results are more diffusive the greater the value of δ_0 is.

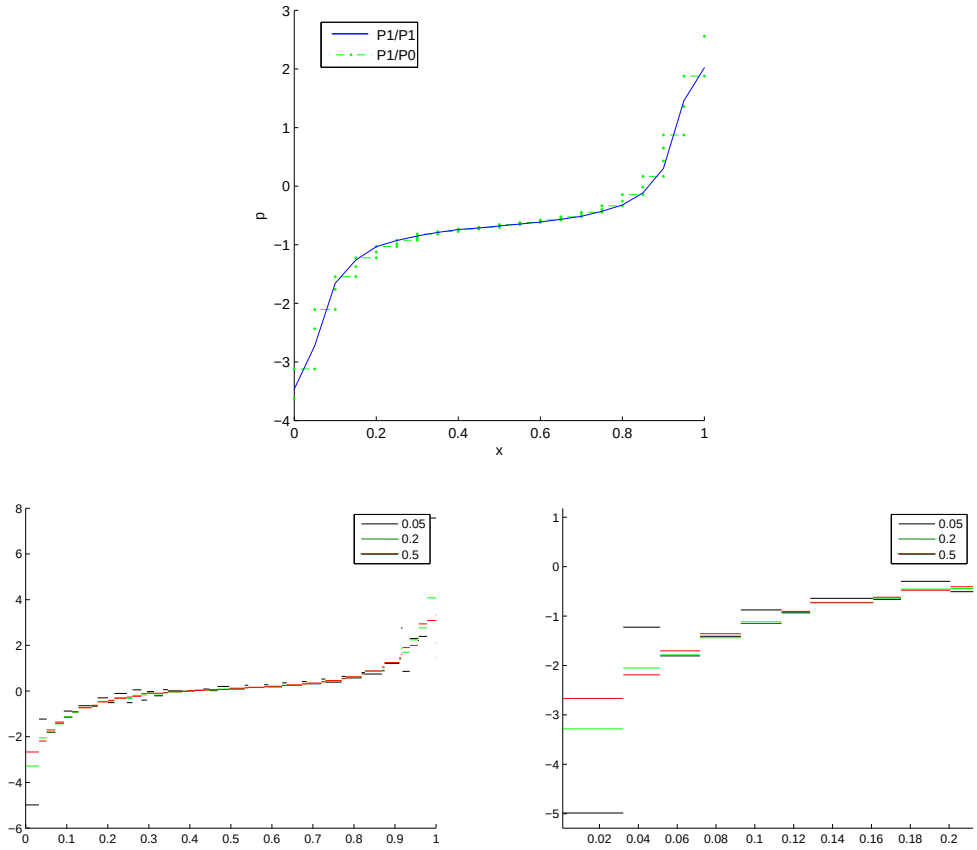


Figure 6.5: Results for the flow in a cavity, pressure on a cut along $y = 1$. Top: P_1/P_0 element with $\delta_0 = 0.2$ compared to the P_1/P_1 element. Bottom: P_1/P_0 element results for different values of δ_0, global cut (left) and detail (right).

Flow over a cylinder

In this example we study the Stokes flow past a cylinder. The computational domain is $\Omega = [0, 16]\times[0, 8]\backslash D$, with the cylinder D of diameter 2 and centered at (4,4). The velocity at $x = 0$ is prescribed at $(1, 0)$, whereas at $y = 0$ and $y = 8$, the y-velocity component is prescribed to

0 and the x-component is left free. The outflow, where both the x- and y- component are free, is $x = 16$. Tractions are set to 0 on the outflow.

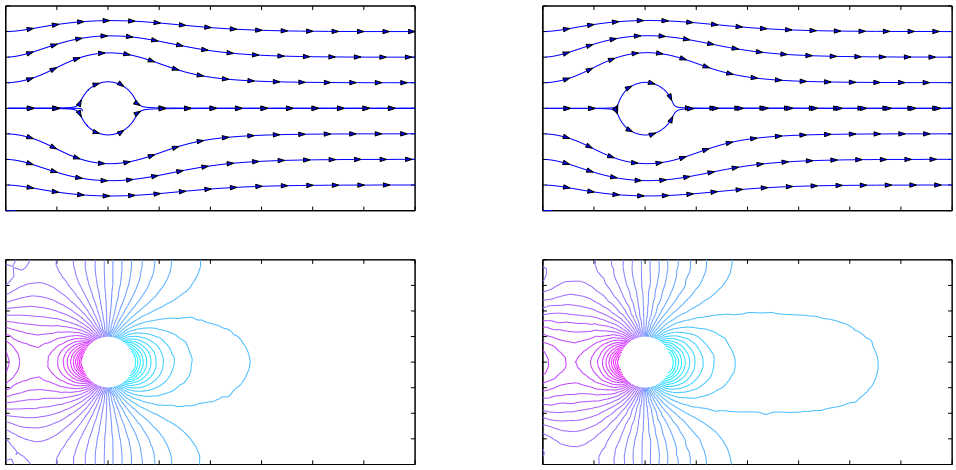


Figure 6.6: Results for the flow over a cilinder. Left: P_1/P_1 (without $\boldsymbol{u}'_E$). Right: P_1/P_0 (with $\boldsymbol{u}'_E$). From top to bottom: streamlines and pressure contours.

As in the previous example, little difference can be appreciated between the solutions obtained with the P_1/P_1 pair with no subscales on the boundaries and the P_1/P_0 element with subscales on the boundaries.

Once again, the behavior of the solution in terms of δ_0 has been checked. A comparison between the pressure in a cut along $y = 4$ is shown in Fig. 6.7 for $\delta_0 = 0.05$, 0.2 and 0.5. The same conclusions as for the cavity flow example can be drawn in this case, namely, $\delta_0 = 0.05$ allows for pressure oscillations which do not appear using $\delta_0 = 0.2$ and $\delta_0 = 0.5$, the latter being more diffusive than the former.

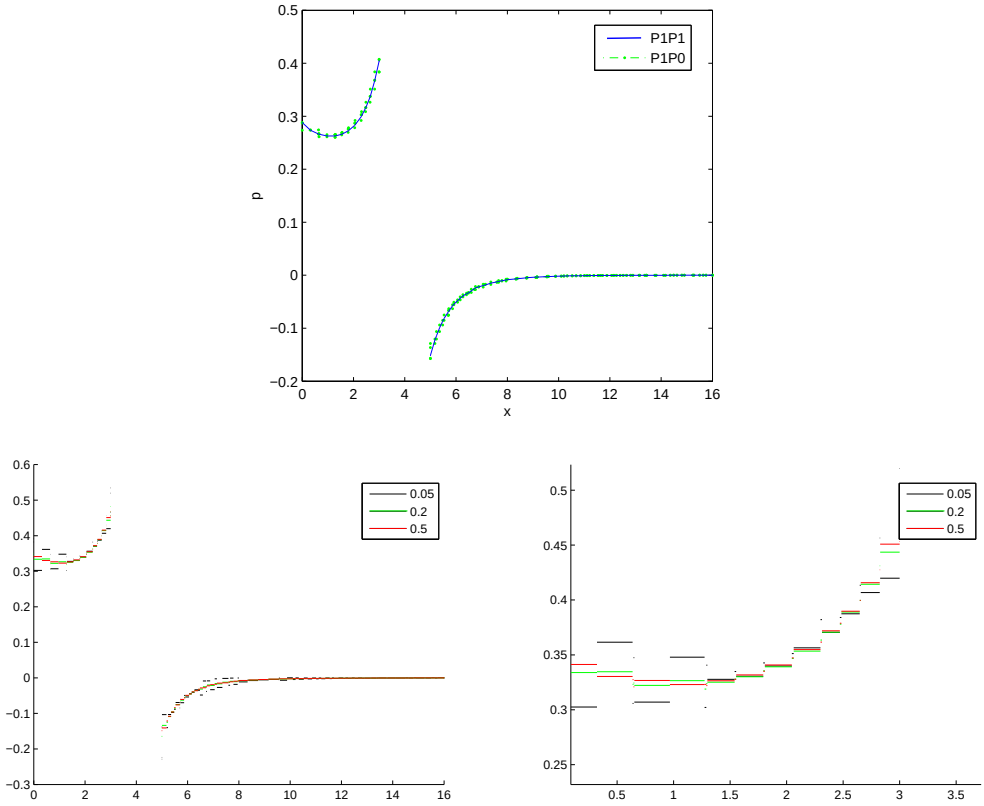


Figure 6.7: Results for the flow over a cilinder, pressure in a cut along $y = 4$. Top: P_1/P_0 element with $\delta_0 = 0.2$ compared to the P_1/P_1 element. Bottom: P_1/P_0 element results for different values of δ_0, global cut (left) and detail (right).

6.6 In search of an efficient implementation for the $P1/P0$ element

We have seen how considering subscales to be different from zero in the element edges can be used to obtain stabilized formulations for discontinuous pressure interpolations, in particular for the $P1/P0$ element. However, this element is not very useful from the computational point of view: compared to the stabilized $P1/P1$ element, the number of pressure unknowns and the number of connectivities for velocity unknowns are greater. In this section we try to find a more efficient formulation in which pressure unknowns can be condensed by sending all the off-diagonal terms of the LHS corresponding to pressure test functions equations to the RHS. This is not effective for the Stokes problem, since an iterative process would be needed for a linear problem, but can be worth in the Navier-Stokes equations where the iterations due to the pressure condensation can be coupled with the iterations due to the non-linearity of the convective term.

Let us consider the subscales on the boundary strategy for the Stokes problem defined in (6.52)-(6.53) and apply it to the $P1/P0$ interpolation. We take into account that, for this particular interpolation:

$$\Delta \boldsymbol{v}_h = 0, \quad \nabla q_h = 0,$$

As we will see, subscales on the element interiors can be neglected in the Stokes problem if the $P1/P0$ interpolation is used. However, this is not the case if we deal with the Navier-Stokes equations. Let us start by defining an explicit expression for the subscales in the element interiors, which can be obtained from (6.53):

$$\boldsymbol{u}' = \tau P_{V'}(\boldsymbol{f} + \nu \Delta \boldsymbol{u}_h - \nabla p_h),$$

where $P_{V'}$ is the projection onto the space for the subscales. This term can be neglected if we consider V' to be orthogonal to the finite element space and $\boldsymbol{f}$ to belong to the finite element space. In this case, the final variational formulation is:

$$B([\boldsymbol{u}_h, p_h], [\boldsymbol{v}_h, q_h]) - \sum_E \frac{\delta}{2\nu} \langle [\![\nu \partial_n \boldsymbol{u}_h - p_h \boldsymbol{n}]\!], [\![\nu \partial_n \boldsymbol{v}_h + q_h \boldsymbol{n}]\!] \rangle_E = \langle \boldsymbol{f}, \boldsymbol{v}_h \rangle, \qquad (6.73)$$

which must hold for all $[\boldsymbol{v}_h, q_h] \in V_h \times Q_h$. The matrix structure of the previous problem is the following:

$$\begin{bmatrix} \boldsymbol{K}_0 + \boldsymbol{K}_\delta & \boldsymbol{G}_0 + \boldsymbol{G}_\delta \\ \boldsymbol{G}_0^T + \boldsymbol{G}_\delta^T & -\boldsymbol{J}_\delta \end{bmatrix} \begin{bmatrix} \boldsymbol{U} \\ \boldsymbol{P} \end{bmatrix} = \begin{bmatrix} \boldsymbol{F} \\ \boldsymbol{0} \end{bmatrix}, \qquad (6.74)$$

where $\boldsymbol{K}_\delta$ corresponds to $-\sum_E \frac{\delta}{2\nu} \langle [\![\nu \partial_n \boldsymbol{u}_h]\!], [\![\nu \partial_n \boldsymbol{v}_h]\!] \rangle_E$, $\boldsymbol{G}_\delta$ and $\boldsymbol{G}_\delta^T$ correspond to $-\sum_E \frac{\delta}{2\nu} \langle [\![p_h \boldsymbol{n}]\!], [\![\nu \partial_n \boldsymbol{v}_h]\!] \rangle_E$ and $-\sum_E \frac{\delta}{2\nu} \langle [\![\nu \partial_n \boldsymbol{u}_h]\!], [\![q_h \boldsymbol{n}]\!] \rangle_E$ respectively and $-\boldsymbol{J}_\delta$ corresponds to $\sum_E \frac{\delta}{2\nu} \langle [\![p_h \boldsymbol{n}]\!], [\![q_h \boldsymbol{n}]\!] \rangle_E$.

In order to be able to condensate pressure unknowns, $\boldsymbol{J}_\delta$ should be diagonal. Since $\boldsymbol{J}_\delta$ involves the pressure jumps across interelement boundaries, it is not diagonal. However, we can split it into a diagonal and a non-diagonal part:

$$\boldsymbol{J}_\delta = \boldsymbol{J}_\delta^0 - \boldsymbol{J}_\delta',$$
$$\boldsymbol{J}_\delta^0 = \mathrm{diag}(\boldsymbol{J}_\delta),$$

We can now write the problem in the following form:

$$\begin{bmatrix} \boldsymbol{K}_0 + \boldsymbol{K}_\delta & \boldsymbol{G}_0 + \boldsymbol{G}_\delta \\ \boldsymbol{G}_0^T + \boldsymbol{G}_\delta^T & -\boldsymbol{J}_\delta^0 \end{bmatrix} \begin{bmatrix} \boldsymbol{U} \\ \boldsymbol{P} \end{bmatrix} = \begin{bmatrix} \boldsymbol{F} \\ \boldsymbol{J}'_\delta \boldsymbol{P} \end{bmatrix}, \tag{6.75}$$

which can be solved by means of a fixed point iteration scheme:

$$\begin{bmatrix} \boldsymbol{K}_0 + \boldsymbol{K}_\delta & \boldsymbol{G}_0 + \boldsymbol{G}_\delta \\ \boldsymbol{G}_0^T + \boldsymbol{G}_\delta^T & -\boldsymbol{J}_\delta^0 \end{bmatrix} \begin{bmatrix} \boldsymbol{U}^{(i)} \\ \boldsymbol{P}^{(i)} \end{bmatrix} = \begin{bmatrix} \boldsymbol{F} \\ \boldsymbol{J}'_\delta \boldsymbol{P}^{(i-1)} \end{bmatrix}, \tag{6.76}$$

where i is the iteration counter. Since $\boldsymbol{J}_\delta^0$ can be trivially inverted, we can eliminate the pressure unknowns from the matrix form of the problem and end up with:

$$\left[\boldsymbol{K}_0 + \boldsymbol{K}_\delta + (\boldsymbol{G}_0 + \boldsymbol{G}_\delta)\left(\boldsymbol{J}_\delta^0\right)^{-1}\left(\boldsymbol{G}_0^T + \boldsymbol{G}_\delta^T\right)\right]\boldsymbol{U}^{(i)} = \boldsymbol{F} + (\boldsymbol{G}_0 + \boldsymbol{G}_\delta)\left(\boldsymbol{J}_\delta^0\right)^{-1}\boldsymbol{J}'_\delta \boldsymbol{P}^{(i-1)}, \tag{6.77}$$

$\boldsymbol{J}_\delta$ is diagonally dominant, and as a consequence the convergence of the scheme can be proved as for the classical Jacobi method.

In the case of transient problems the same strategy can be used. However, we also have the possibility of treating pressure in an explicit way. The matrix form of the transient problem is:

$$\begin{bmatrix} \frac{1}{\delta t}\boldsymbol{M} + \boldsymbol{K}_0 + \boldsymbol{K}_\delta & \boldsymbol{G}_0 + \boldsymbol{G}_\delta \\ \boldsymbol{G}_0^T + \boldsymbol{G}_\delta^T & -\boldsymbol{J}_\delta^0 \end{bmatrix} \begin{bmatrix} \boldsymbol{U}^{(n+1)} \\ \boldsymbol{P}^{(n+1)} \end{bmatrix} = \begin{bmatrix} \boldsymbol{F} + \frac{1}{\delta t}\boldsymbol{M}\boldsymbol{U}^n \\ \boldsymbol{J}'_\delta \tilde{\boldsymbol{P}}^{n+1} \end{bmatrix}, \tag{6.78}$$

where we have added a mass matrix which takes into account the time derivatives and $\tilde{\boldsymbol{P}}^{n+1}$ is an approximation to the pressure nodal unknowns at time step $n+1$. A first order approximation to the pressure would be to consider the pressure at the previous time step:

$$\tilde{\boldsymbol{P}}^{n+1} = \boldsymbol{P}^n = \boldsymbol{P}^{n+1} + \mathcal{O}(\delta t),$$

Similarly we could consider the second order approximation:

$$\tilde{\boldsymbol{P}}^{n+1} = 2\boldsymbol{P}^n - \boldsymbol{P}^{n-1} = \boldsymbol{P}^{n+1} + \mathcal{O}(\delta t^2).$$

However, although some of these methods work, none of them shows a fast enough convergence to be competitive with the $P1/P1$ interpolation. We solve the example of the flow over a cylinder described in the previous section. Fig. 6.8 shows the convergence of the iterative scheme proposed for the stationary Stokes problem. We can appreciate that convergence is very slow, even if a relaxation strategy is used. If we try using over-relaxation the iterative scheme diverges. In Fig. 6.9 we compare the convergence of the non-linearity of the Navier-Stokes equations ($Re = 100$) solved with the original formulation with the convergence if the condensed pressure strategy is used. We can observe that the convergence of the convective non-linearity is much faster than the convergence of the iterative algorithm due to the pressure condensation.

We finally try the explicit treatment for the pressures described for the transient Stokes problem. After 50 time steps the first order scheme has not been able to converge to an incompressible velocity field, as we can appreciate in Fig. 6.10, where the horizontal velocity has been depicted. The second order scheme diverges after a few time steps.

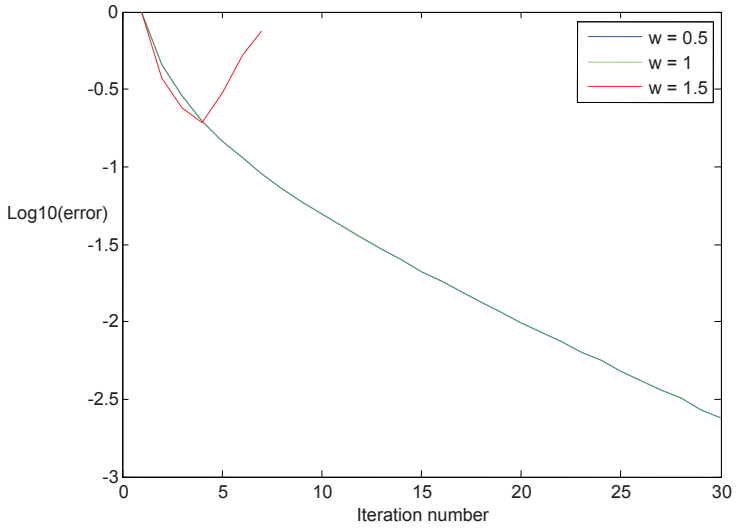


Figure 6.8: Convergence for the iterative scheme for the stationary Stokes problem. Comparison between several relaxation parameters w.

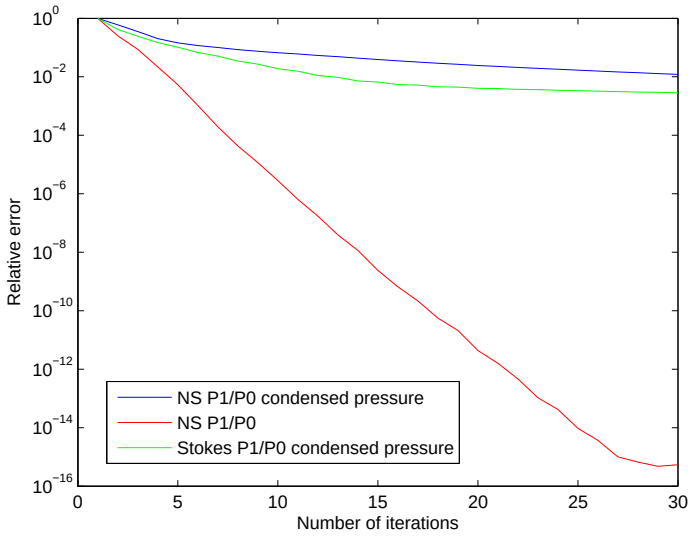


Figure 6.9: Convergence for the iterative scheme for the stationary Navier-Stokes problem. Comparison between condensed, non-condensed pressures and the iterative solution of the Stokes problem.

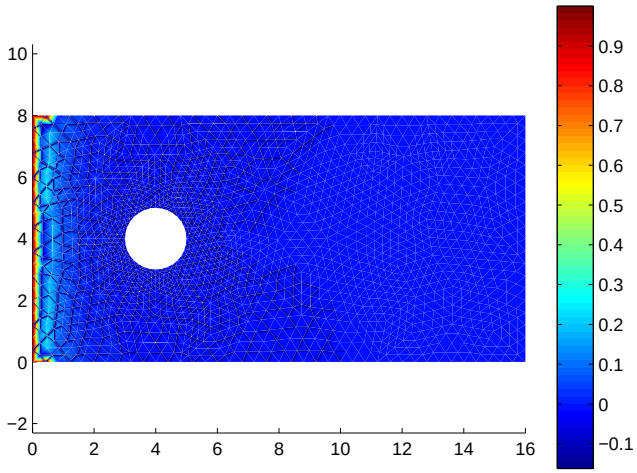


Figure 6.10: Horizontal velocity for the transient problem if an explicit scheme for the pressure is used. Results for the first order scheme.

In conclusion, we have not been able to find a competitive algorithm for the implementation of the $P1/P0$ element. Further research will be carried out in order to consider more complex iterative schemes (starting with, for example, Gauss-Seidel iterations) which might allow to condense the pressure unknowns and at the same time to obtain convergence in very few iterations.

6.7 Conclusions

In this chapter we have extended the two-scale approximation of variational problems with an additional ingredient in the approximation of the subscales, which is an approximation for their values on the interelement boundaries.

The key idea is to assume that the subscales are already computed in the element interiors and to compute the boundary values by imposing the correct transmission conditions of the problem under consideration. Three examples of how to undertake this process have been presented, namely, the CDR equation, the Stokes problem and Darcy's equations.

In order to be as general as possible, examples of how to compute the subscale on the element interiors have been proposed, *but not used*, in the sense that our developments are applicable to any approximation of these unknowns (provided they satisfy some conditions on the algorithmic constants on which they depend). In fact, we have proved stability estimates for the three problems considered which are valid for any choice of subscales in the interior of the elements. However, convergence analyses, not presented here, require the expressions of these subscales.

For the case of the CDR equation, the new terms introduced by accounting for the subscales on the interelement boundaries do not contribute to stability. However, our analysis and the numerical example presented show that they do not spoil it, and also that accuracy seems also

to be unaffected. However, for the Stokes problem and for Darcy flow the terms introduced by the subscales on the boundaries are crucial to provide stability when discontinuous pressure interpolations are used. The *stabilizing* terms introduced are shared with other formulations that can be found in the literature. However, some non-standard terms also appear. Again, our analysis, and the numerical examples in the case of the Stokes problems, show that these terms do not harm stability.

We have looked for an efficient implementation of the $P1/P0$ interpolation, which consists of condensing the pressure unknowns by sending the off-diagonal terms corresponding to the pressure test function equations to the right hand-side. Several iterative and explicit methods have been presented which are suitable for stationary and transient problems respectively. However, although some of these methods work, none of them shows a fast enough convergence to be competitive with the $P1/P1$ interpolation. Further research will be carried out in order to consider more complex iterative schemes (starting with, for example, Gauss-Seidel iterations) which might allow to condense the pressure unknowns and at the same time obtain convergence in very few iterations.

Chapter 7

Finite element approximation of transmission conditions in fluids and solids introducing boundary subgrid scales

Terms involving jumps of stresses on boundaries are proposed for the finite element approximation of the Stokes problem and the linear elasticity equations. These terms are designed to improve the transmission conditions between subdomains at three different levels, namely, between the element domains, between the interfaces in homogeneous domain interaction problems and at the interface between the fluid and the solid in fluid-structure interaction problems. The benefits in each case are respectively the possibility of using discontinuous pressure interpolations in a stabilized finite element approximation of the Stokes problem, a stronger enforcement of the stress continuity in homogeneous domain decomposition problems and a considerable improvement of the behavior of the iterative algorithm to couple the fluid and the solid in fluid-structure interaction problems. The motivation to introduce these terms stems from a decomposition of the unknown into a conforming and a non-conforming part, a hybrid formulation for the latter and a simple approximation for the unknowns involved in the hybrid problem.

7.1 Introduction

Transmission conditions in the numerical approximation of fluid and solid mechanics problems play a key role at different levels. When the discretization involves a partition of the computational domain, as in finite volume or finite element methods, the first level is the interaction between the subdomains of the partition. Appropriate interaction conditions, associated to the problem being solved, are satisfied in an approximate way, and this may have important consequences in the stability of the numerical method. A second level of analysis of transmission conditions could be the interaction of subdomains in a homogeneous domain decomposition method. This problem may be addressed using a purely algebraic point of view, but it is also possible to analyze the interaction from the standpoint of the approximate boundary condi-

tions applied to each subdomain. Both strategies are well known in the domain decomposition community (see [115, 128], for example). A a third level of analysis could be the interaction between *heterogeneous* subdomains, in which different problems associated to different physics are solved within each of the subdomains. This last category could be included in the second, but the heterogeneity of the transmission conditions introduces additional difficulties that deserve to be studied independently. The paradigmatic example of this class of problems are those involving fluid and structure interactions.

In this chapter we analyze the issue of dealing with transmission conditions in fluid and solid mechanics problems approximated using finite elements. The model problems we will consider are the Stokes problem and the Navier equations for a linear elastic solid. Our proposal is to modify the classical approximation of the interaction stresses computed from the finite element solution by introducing terms that depend on the jumps of these stresses when computed from the two sides of the interaction boundary. The way to motivate the introduction of these terms is as follows. First, we consider a splitting of the unknown into a conforming and a discontinuous part. A three-field hybrid formulation is used for the latter, involving the primal variable, its traces and its fluxes on the element boundaries as unknowns. We assume that these terms are *small*, and therefore we consider them as *subgrid scales* (or subscales) of the conforming part of the solution. In this sense, our approach falls within the variational multiscale framework proposed in [77]. Rather than solving for the subscales, we propose simple expressions to *model* them, the main idea being the correct continuity of stresses across the interelement boundaries.

When solving for the Stokes problem in a single domain, the introduction of the element boundary terms involving jumps of stresses has as a consequence a stabilizing effect on the pressure. In particular, in combination with a more standard stabilized finite element method, these new terms open the possibility to use arbitrary discontinuous pressure interpolations, avoiding the need to satisfy the classical velocity-pressure compatibility conditions [23]. Their stabilizing effect is similar to that already found in [78], although their expression is different and motivated in a completely different way.

Pressure stabilization due to the new interelement boundary terms was already proposed and analyzed in Chapter 6. In the present chapter we derive in detail the formulation for the Stokes problem (which in the previous chapter was directly stated from the derivation obtained for the convection-diffusion equation), with emphasis on the treatment of Neumann boundary conditions. This serves us to extend it to two cases, namely, the interaction between two subdomains, in each of which the Stokes problem is solved, and the classical fluid-structure interaction (FSI) problem. In the first case, the new terms we propose help to enforce the continuity of stresses between subdomains. The domain interaction is however more complex than in classical formulations. To introduce an iteration-by-subdomain scheme, we first analyze the matrix structure of the problem and discuss how this iterative scheme can be designed. In the FSI case, we apply the previous ideas to a time-marching block-iterative scheme in which Dirichlet boundary conditions are prescribed to the fluid and Neumann boundary conditions are applied to the solid. The latter correspond to the normal stress exerted by the fluid on the solid. The introduction of the subscales on the element boundaries for the solid enhances notably the stability of the scheme. We illustrate this enhancement with a numerical example. In particular, the example we have chosen displays the so called added-mass effect, which is one of the most important issues to be considered when solving fluid structure interaction problems

by means of a domain decomposition technique. This phenomenon takes place when fluid and solid densities are similar, and consists in a failing of simple coupling strategies. Either if the coupling is implicit (iterating the solution at each time step to converge to the monolithic problem) or explicit (no coupling iterations are done within the time step), the scheme becomes unstable. Several strategies to deal with this problem have been recently proposed. For example, a semi-implicit scheme for pressure-segregated methods is presented in [49], and is further developed in [6]. A different approach for pressure-segregated schemes is proposed in [81]. In [24] a strategy based on Nitsche's method is proposed and a method based on Robin boundary conditions can be found in [8]. Using a more algebraic point of view, several strategies using preconditioned Krylov methods are presented in [9, 10, 11]. Conditions under which the problem becomes unstable are studied in [54]. We present here our own approach, which is directly derived from the use of boundary subgrid scales.

The chapter is organized as follows. In Section 7.2 we state the Stokes problem in strong form, in the classical velocity-pressure variational form and in a non-standard hybrid variational form that we use to motivate our numerical formulation. This formulation is presented in detail in Section 7.3. The final result is a problem posed only for a conforming approximation to the velocity and the pressure that involves jumps of stresses at the interelement boundaries. The application of the same ideas to a homogeneous domain interaction problem is presented in Section 7.4, whereas the application to the FSI problem is the subject of Section 7.5. Numerical examples are presented in Section 7.6 and finally conclusions close the chapter in Section 7.7.

7.2 Problem statement

7.2.1 Stokes problem in $\boldsymbol{u}$-p form

Let us start considering the Stokes problem written in the classical velocity-pressure approach or displacement-pressure, in the case of an elastic solid. To fix terminology, we will consider that it corresponds to a fluid, leaving for Section 7.5 the statement of the elastic problem. Thus, the problem we consider here consists in finding a velocity $\boldsymbol{u} : \Omega \longrightarrow \mathbb{R}^d$ and a pressure $p : \Omega \longrightarrow \mathbb{R}$ such that

$$-\mu\Delta\boldsymbol{u} + \nabla p = \rho\boldsymbol{f} \quad \text{in } \Omega, \tag{7.1}$$

$$\nabla \cdot \boldsymbol{u} = 0 \quad \text{in } \Omega, \tag{7.2}$$

$$\boldsymbol{u} = \boldsymbol{0} \quad \text{on } \Gamma_D, \tag{7.3}$$

$$-p\,\boldsymbol{n} + \mu\,\boldsymbol{n} \cdot \nabla\boldsymbol{u} = \boldsymbol{t} \quad \text{on } \Gamma_N. \tag{7.4}$$

In these equations, $\Omega \subset \mathbb{R}^d$ ($d = 2, 3$) is a bounded domain with boundary $\partial\Omega$ and external normal $\boldsymbol{n}$, $\boldsymbol{f}$ is the vector of body forces and $\boldsymbol{t}$ is the (pseudo-)traction prescribed on Γ_N, with $\partial\Omega = \overline{\Gamma_N \cup \Gamma_D}$, $\Gamma_N \cap \Gamma_D = \emptyset$, $\Gamma_D \neq \emptyset$. The physical parameters μ and ρ are the viscosity and the density, respectively. Note that the Neumann-type conditions do not correspond to the physically meaningful tractions, for which the viscous term should be written using the symmetrical gradient of the velocity. This, however, is irrelevant for our discussion.

Let now $V = H^1_\Gamma(\Omega)^d := \{\boldsymbol{v} \in H^1(\Omega)^d \mid \boldsymbol{v} = \boldsymbol{0} \text{ on } \Gamma_D\}$, $Q = L^2(\Omega)$, and assume that $\boldsymbol{f} \in (H^1_\Gamma(\Omega)^d)'$ (the dual space of $H^1_\Gamma(\Omega)^d$) and $\boldsymbol{t} \in H^{-1/2}(\Gamma_N)^d$. We will use the symbol

$(\cdot,\cdot)_\omega$ to denote the L^2 product in a domain ω. In general, the integral of two functions g_1 and g_2 over a domain ω will be denoted by $\langle g_1, g_2\rangle_\omega$. This symbol will also be used for the duality pairing. The simplifications $(\cdot,\cdot)_\Omega \equiv (\cdot,\cdot)$ and $\langle\cdot,\cdot\rangle_\Omega \equiv \langle\cdot,\cdot\rangle$ will be used.

The variational problem consists of finding $[\boldsymbol{u}, p] \in V \times Q$ such that

$$B([\boldsymbol{u}, p], [\boldsymbol{v}, q]) = L([\boldsymbol{v}, q]) + \langle \boldsymbol{t}, \boldsymbol{v}\rangle_{\Gamma_N} \quad \forall [\boldsymbol{v}, q] \in V \times Q,$$

where

$$B([\boldsymbol{u}, p], [\boldsymbol{v}, q]) := \mu(\nabla \boldsymbol{u}, \nabla \boldsymbol{v}) - (p, \nabla \cdot \boldsymbol{v}) + (q, \nabla \cdot \boldsymbol{u}),$$
$$L([\boldsymbol{v}, q]) := \rho\langle \boldsymbol{f}, \boldsymbol{v}\rangle.$$

7.2.2 Hybrid formulation of an abstract variational problem

The numerical approximation we propose can be motivated from a hybrid formulation of the problem. To introduce it, let us assume that $\bar{\Omega} = \bar{\Omega}_1 \cup \bar{\Omega}_2$, with $\Gamma = \partial\Omega_1 \cap \partial\Omega_2$ (see Fig. 7.1).

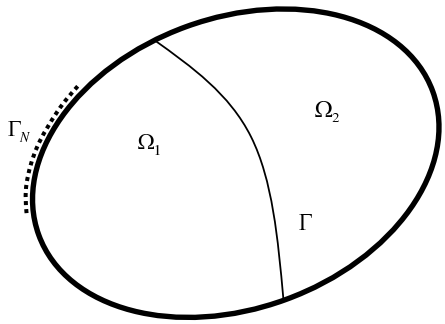


Figure 7.1: Splitting of the domain

Consider an abstract variational problem consisting in finding an unknown u in a functional space X such that

$$a(u, v) = l(v) \quad \forall v \in X, \tag{7.5}$$

where $a(u, v)$ is a bilinear form on $X \times X$ and l a linear form defined on X. Let u_i, v_i be the restrictions of $u, v \in X$ to subdomain Ω_i, and X_i the spaces where they belong, $i = 1, 2$. Suppose that $u \in X$ has a well defined trace on Γ belonging to a space T, and a flux corresponding to the differential operator associated to (7.5) belonging to a space F_i when computed from subdomain Ω_i, $i = 1, 2$. Then, the hybrid formulation of (7.5) that we consider is the following: find $u_i \in X_i$, $\lambda_i \in F_i$, $i = 1, 2$, and $\gamma \in T$ such that

$$\begin{aligned}
&a_1(u_1, v_1) - \langle \lambda_1, v_1\rangle_\Gamma = l_1(v_1) && \forall v_1 \in X_1,\\
&a_2(u_2, v_2) - \langle \lambda_2, v_2\rangle_\Gamma = l_2(v_2) && \forall v_2 \in X_2,\\
&\langle \mu_1, u_1 - \gamma\rangle_\Gamma = 0 && \forall \mu_1 \in F_1,\\
&\langle \mu_2, u_2 - \gamma\rangle_\Gamma = 0 && \forall \mu_2 \in F_2,\\
&\langle \kappa, \lambda_1 + \lambda_2\rangle_\Gamma = 0 && \forall \kappa \in T,
\end{aligned}$$

where a_i and l_i are the restrictions of a and l to $X_i \times X_i$ and X_i, respectively.

If the problem includes imposition of fluxes of u in a part Γ_N of $\partial\Omega$, which for the sake of simplicity we may consider contained in $\partial\Omega_1$ (see Fig. 7.1), this imposition may be also "hybridized", yielding the problem

$$a_1(u_1, v_1) - \langle \lambda_1, v_1 \rangle_\Gamma - \langle \lambda_N, v_1 \rangle_{\Gamma_N} = l_1(v_1) \quad \forall v_1 \in X_1, \tag{7.6}$$

$$a_2(u_2, v_2) - \langle \lambda_2, v_2 \rangle_\Gamma = l_2(v_2) \quad \forall v_2 \in X_2, \tag{7.7}$$

$$\langle \mu_1, u_1 - \gamma \rangle_\Gamma + \langle \mu_1, u_1 - \gamma \rangle_{\Gamma_N} = 0 \quad \forall \mu_1 \in F_1, \tag{7.8}$$

$$\langle \mu_2, u_2 - \gamma \rangle_\Gamma = 0 \quad \forall \mu_2 \in F_2, \tag{7.9}$$

$$\langle \kappa, \lambda_1 + \lambda_2 \rangle_\Gamma + \langle \kappa, \lambda_N \rangle_{\Gamma_N} = \langle \kappa, q \rangle_{\Gamma_N} \quad \forall \kappa \in T, \tag{7.10}$$

where q is the flux to be prescribed. In this case, the linear form l (and the forms l_1 and l_2 resulting from the splitting of the domain) does *not* include the prescription of the fluxes.

Several problems admit this hybrid formulation, including first and second order linear partial differential equations (the fluxes λ_i are zero in the first case). In the following subsection we shall see its application to the Stokes problem. For the diffusion equation $-\Delta u = f$ with $u = 0$ on $\partial\Omega$, we would have that X_i is the subspace of $H^1(\Omega_i)$ of functions vanishing on $\partial\Omega \cap \partial\Omega_i$, $T = H_{00}^{1/2}(\Gamma)$ and $F_i = (H_{00}^{1/2}(\Gamma_i))'$ (the prime denoting dual space), with $\Gamma_i = \partial\Omega_i \cap \Omega$. The solution of the hybrid problem is $\gamma = u_1|_{\Gamma_1} = u_2|_{\Gamma_2}$, $\lambda_1 = -\lambda_2 = \boldsymbol{n}_1 \cdot \nabla u_1|_{\Gamma_1} = -\boldsymbol{n}_2 \cdot \nabla u_2|_{\Gamma_2}$. If flux conditions need to be prescribed, in this case they are of the form $\boldsymbol{n} \cdot \nabla u = q$, and Γ_1 will contain Γ_N, the part of the boundary where these conditions are enforced.

7.2.3 Hybrid formulation of the Stokes problem

The Stokes problem (7.1)-(7.2) admits also the hybrid formulation described above by defining

$$\begin{aligned}
&u = [\boldsymbol{u}, p], \quad v = [\boldsymbol{v}, q], \quad X = V \times Q, \\
&a(u, v) = B([\boldsymbol{u}, p], [\boldsymbol{v}, q]), \quad l(v) = L([\boldsymbol{v}, q]), \\
&\gamma = \boldsymbol{u}_1|_{\Gamma_1} = \boldsymbol{u}_2|_{\Gamma_2} \in T = H_{00}^{1/2}(\Gamma)^d, \\
&\lambda_i = (-p\boldsymbol{n}_i + \mu \boldsymbol{n}_i \cdot \nabla \boldsymbol{u})|_{\Gamma_i} \in F_i = (H_{00}^{1/2}(\Gamma_i)^d)', \quad i = 1, 2,
\end{aligned}$$

and $q = \boldsymbol{t}$ (see (7.4)) being the boundary condition in terms of fluxes.

To present the formulation we are interested in, let us consider a splitting of space V of the form $V = \bar{V} \oplus \tilde{V}$. In principle, there is no restriction in the expression of spaces $\bar{V}$ and $\tilde{V}$. In the finite element approximation, the former will be approximated by *continuous* finite elements (and therefore conforming). The component of $\boldsymbol{u}$ in this space can be considered as *resolvable*, whereas a closed form expression will be given for the component in $\tilde{V}$, which will be called *subgrid scale* or, simply, *subscale*. A similar splitting could be performed for the pressure, although it is not necessary for our purposes.

Let V_i be also split as $V_i = \bar{V}_i \oplus \tilde{V}_i$, $i = 1, 2$. If any $\boldsymbol{u}_i \in V_i$ is written as $\boldsymbol{u}_i = \bar{\boldsymbol{u}}_i + \tilde{\boldsymbol{u}}_i$, with $\bar{\boldsymbol{u}}_i \in \bar{V}_i$ and $\tilde{\boldsymbol{u}}_i \in \tilde{V}_i$, we assume that $\bar{\boldsymbol{u}}_1|_{\Gamma_1} = \bar{\boldsymbol{u}}_2|_{\Gamma_2}$. Only the continuity for the component in $\tilde{V}_i$ needs to be enforced (weakly) through a variational equation.

Let us introduce the boundary operators

$$\mathcal{T}_i([\bar{\boldsymbol{v}}_i, q_i]) := (-q_i \boldsymbol{n}_i + \mu_i \boldsymbol{n}_i \cdot \nabla \bar{\boldsymbol{v}}_i)|_{\Gamma_i}, \quad \mathcal{T}_i^*([\bar{\boldsymbol{v}}_i, q_i]) := (q_i \boldsymbol{n}_i + \mu_i \boldsymbol{n}_i \cdot \nabla \bar{\boldsymbol{v}}_i)|_{\Gamma_i},$$

where $[\boldsymbol{v}_i, q_i] \in \bar{V}_i \times Q_i$, $i = 1, 2$. We may constrain the fluxes to be of the form $\boldsymbol{\lambda}_i = \mathcal{T}_i([\bar{\boldsymbol{u}}_i, p_i]) + \tilde{\boldsymbol{\lambda}}_i$, for appropriate $\tilde{\boldsymbol{\lambda}}_i \in \tilde{F}_i$, and likewise for $\boldsymbol{\lambda}_N$. Test functions in F_i can be similarly split as $\boldsymbol{\mu}_i = \mathcal{T}_i^*([\bar{\boldsymbol{v}}_i, q_i]) + \tilde{\boldsymbol{\mu}}_i$, with $\bar{\boldsymbol{v}}_i \in \bar{V}_i$, $q_i \in Q_i$, $\tilde{\boldsymbol{\mu}}_i \in \tilde{F}_i$. Finally, traces on boundaries can be split as $\boldsymbol{\gamma} = \bar{\boldsymbol{u}} + \tilde{\boldsymbol{\gamma}}$, both on Γ and on Γ_N. On the intersecting boundary Γ the restriction $\bar{\boldsymbol{u}}$ is well defined because of the assumption $\bar{\boldsymbol{u}}_1|_{\Gamma_1} = \bar{\boldsymbol{u}}_2|_{\Gamma_2}$. Note that, in fact, $\tilde{F}_i = F_i$ and $\tilde{T} = T$. The tilde has been introduced to stress that we seek the subscale of fluxes and traces in these spaces.

Having introduced these decompositions of the functional spaces, we may write the hybrid formulation of the Stokes problem as follows: find $\bar{\boldsymbol{u}}_i \in \bar{V}_i$, $\tilde{\boldsymbol{u}}_i \in \tilde{V}_i$, $p_i \in Q_i$, $\tilde{\boldsymbol{\gamma}} \in \tilde{T}$, $\tilde{\boldsymbol{\lambda}}_i \in \tilde{F}_i$ ($i = 1, 2$) such that

$$\begin{aligned}
&B_1([\bar{\boldsymbol{u}}_1 + \tilde{\boldsymbol{u}}_1, p_1], [\bar{\boldsymbol{v}}_1 + \tilde{\boldsymbol{v}}_1, q_1]) - \left\langle \mathcal{T}_1([\bar{\boldsymbol{u}}_1, p_1]) + \tilde{\boldsymbol{\lambda}}_1, \bar{\boldsymbol{v}}_1 + \tilde{\boldsymbol{v}}_1 \right\rangle_{\Gamma} \\
&\qquad - \left\langle \mathcal{T}_1([\bar{\boldsymbol{u}}_1, p_1]) + \tilde{\boldsymbol{\lambda}}_N, \bar{\boldsymbol{v}}_1 + \tilde{\boldsymbol{v}}_1 \right\rangle_{\Gamma_N} = L_1([\bar{\boldsymbol{v}}_1 + \tilde{\boldsymbol{v}}_1, q_1]), \qquad (7.11) \\
&B_2([\bar{\boldsymbol{u}}_2 + \tilde{\boldsymbol{u}}_2, p_2], [\bar{\boldsymbol{v}}_2 + \tilde{\boldsymbol{v}}_2, q_2]) - \left\langle \mathcal{T}_2([\bar{\boldsymbol{u}}_2, p_2]) + \tilde{\boldsymbol{\lambda}}_2, \bar{\boldsymbol{v}}_2 + \tilde{\boldsymbol{v}}_2 \right\rangle_{\Gamma} = L_2([\bar{\boldsymbol{v}}_2 + \tilde{\boldsymbol{v}}_2, q_2]), \qquad (7.12) \\
&\langle \mathcal{T}_1^*([\bar{\boldsymbol{v}}_1, q_1]) + \tilde{\boldsymbol{\mu}}_1, \tilde{\boldsymbol{\gamma}} - \tilde{\boldsymbol{u}}_1 \rangle_{\Gamma} + \langle \mathcal{T}_1^*([\bar{\boldsymbol{v}}_1, q_1]) + \tilde{\boldsymbol{\mu}}_1, \tilde{\boldsymbol{\gamma}} - \tilde{\boldsymbol{u}}_1 \rangle_{\Gamma_N} = 0, \qquad (7.13) \\
&\langle \mathcal{T}_2^*([\bar{\boldsymbol{v}}_2, q_2]) + \tilde{\boldsymbol{\mu}}_2, \tilde{\boldsymbol{\gamma}} - \tilde{\boldsymbol{u}}_2 \rangle_{\Gamma} = 0, \qquad (7.14) \\
&\left\langle \bar{\boldsymbol{v}}_1 + \bar{\boldsymbol{v}}_2 + \tilde{\boldsymbol{\kappa}}, \mathcal{T}_1([\bar{\boldsymbol{u}}_1, p_1]) + \mathcal{T}_2([\bar{\boldsymbol{u}}_2, p_2]) + \tilde{\boldsymbol{\lambda}}_1 + \tilde{\boldsymbol{\lambda}}_2 \right\rangle_{\Gamma} \\
&\qquad + \left\langle \bar{\boldsymbol{v}}_1 + \tilde{\boldsymbol{\kappa}}, \mathcal{T}_1([\bar{\boldsymbol{u}}_1, p_1]) + \tilde{\boldsymbol{\lambda}}_N \right\rangle_{\Gamma_N} = \langle \bar{\boldsymbol{v}}_1 + \tilde{\boldsymbol{\kappa}}, \boldsymbol{t} \rangle_{\Gamma_N}, \qquad (7.15)
\end{aligned}$$

which must hold for all $\bar{\boldsymbol{v}}_i \in V_i$, $\tilde{\boldsymbol{v}}_i \in \tilde{V}_i$, $q_i \in Q_i$, $\tilde{\boldsymbol{\kappa}} \in \tilde{T}$, $\tilde{\boldsymbol{\mu}}_i \in \tilde{F}_i$ ($i = 1, 2$). Recall that L_1 does not contain the contribution from the Neumann-type boundary condition (7.4).

Adding up (7.11) and (7.12) with $\tilde{\boldsymbol{v}}_1 = \tilde{\boldsymbol{v}}_2 = \boldsymbol{0}$ and using (7.15) with $\tilde{\boldsymbol{\kappa}} = \boldsymbol{0}$ yields the original variational equation projected onto $\bar{V} \times Q$, that is to say,

$$B([\bar{\boldsymbol{u}}, p], [\bar{\boldsymbol{v}}, q]) + B([\tilde{\boldsymbol{u}}, 0], [\bar{\boldsymbol{v}}, q]) = L([\bar{\boldsymbol{v}}, q]) + \langle \bar{\boldsymbol{v}}, \boldsymbol{t} \rangle_{\Gamma_N}, \qquad (7.16)$$

which holds for all $[\bar{\boldsymbol{v}}, q] \in \bar{V} \times Q$. If we define the operators

$$\mathcal{L}_i([\bar{\boldsymbol{v}}_i, q_i]) := -\mu \Delta \bar{\boldsymbol{v}}_i + \nabla q_i, \quad \mathcal{L}_i^*([\bar{\boldsymbol{v}}_i, q_i]) := -\mu \Delta \bar{\boldsymbol{v}}_i - \nabla q_i,$$

we may write, making use of (7.13) and (7.14) with $\tilde{\boldsymbol{\mu}}_1 = \tilde{\boldsymbol{\mu}}_2 = \boldsymbol{0}$,

$$\begin{aligned}
B([\tilde{\boldsymbol{u}}, 0], [\bar{\boldsymbol{v}}, q]) &= \sum_{i=1}^{2} B_i([\tilde{\boldsymbol{u}}_i, 0], [\bar{\boldsymbol{v}}_i, q_i]) \\
&= \sum_{i=1}^{2} \langle \tilde{\boldsymbol{u}}_i, \mathcal{L}_i^*([\bar{\boldsymbol{v}}_i, q_i]) \rangle_{\Omega_i} + \sum_{i=1}^{2} \langle \tilde{\boldsymbol{\gamma}}, \mathcal{T}_i^*([\bar{\boldsymbol{v}}_i, q_i]) \rangle_{\Gamma} + \langle \tilde{\boldsymbol{\gamma}}, \mathcal{T}_1^*([\bar{\boldsymbol{v}}_1, q_1]) \rangle_{\Gamma_N}.
\end{aligned} \qquad (7.17)$$

Adding up (7.11) and (7.12) with $\bar{\boldsymbol{v}}_1 = \bar{\boldsymbol{v}}_2 = \mathbf{0}$, $q_1 = q_2 = 0$ yields, after integration by parts of some terms,

$$\begin{aligned}
&\sum_{i=1}^{2} B_i([\bar{\boldsymbol{u}}_i + \tilde{\boldsymbol{u}}_i, p_i], [\tilde{\boldsymbol{v}}_i, 0]) - \sum_{i=1}^{2} \left\langle \mathcal{T}_i([\bar{\boldsymbol{u}}_i, p_i]) + \tilde{\boldsymbol{\lambda}}_i, \tilde{\boldsymbol{v}}_i \right\rangle_{\Gamma} - \left\langle \mathcal{T}_1([\bar{\boldsymbol{u}}_1, p_1]) + \tilde{\boldsymbol{\lambda}}_N, \tilde{\boldsymbol{v}}_1 \right\rangle_{\Gamma_N} \\
&\quad = \sum_{i=1}^{2} \langle \mathcal{L}_i([\bar{\boldsymbol{u}}_i, p_i]), \tilde{\boldsymbol{v}}_i \rangle_{\Omega_i} + \sum_{i=1}^{2} B_i([\tilde{\boldsymbol{u}}_i, 0], [\tilde{\boldsymbol{v}}_i, 0]) - \sum_{i=1}^{2} \left\langle \tilde{\boldsymbol{\lambda}}_i, \tilde{\boldsymbol{v}}_i \right\rangle_{\Gamma} - \left\langle \tilde{\boldsymbol{\lambda}}_N, \tilde{\boldsymbol{v}}_1 \right\rangle_{\Gamma_N} \\
&\quad = \sum_{i=1}^{2} L_2([\tilde{\boldsymbol{v}}_i, 0]). \qquad (7.18)
\end{aligned}$$

As an alternative to (7.11)-(7.15), the final problem can be obtained from (7.16), (7.17), (7.18), (7.13) with $[\bar{\boldsymbol{v}}_1, q_1] = [\mathbf{0}, 0]$, (7.14) with $[\bar{\boldsymbol{v}}_2, q_2] = [\mathbf{0}, 0]$ and (7.15) with $\bar{\boldsymbol{v}}_1 = \bar{\boldsymbol{v}}_2 = \mathbf{0}$. It reads: find $\bar{\boldsymbol{u}}_i \in \bar{V}_i$, $\tilde{\boldsymbol{u}}_i \in \tilde{V}_i$, $p_i \in Q_i$, $\tilde{\boldsymbol{\gamma}} \in \tilde{T}$, $\tilde{\boldsymbol{\lambda}}_i \in \tilde{F}_i$ $(i = 1, 2)$ such that

$$\begin{aligned}
&B([\bar{\boldsymbol{u}}, p], [\bar{\boldsymbol{v}}, q]) + \sum_{i=1}^{2} \langle \tilde{\boldsymbol{u}}_i, \mathcal{L}_i^*([\bar{\boldsymbol{v}}_i, q_i]) \rangle_{\Omega_i} + \sum_{i=1}^{2} \langle \tilde{\boldsymbol{\gamma}}, \mathcal{T}_i^*([\bar{\boldsymbol{v}}_i, q_i]) \rangle_{\Gamma} \\
&\qquad + \langle \tilde{\boldsymbol{\gamma}}, \mathcal{T}_1^*([\bar{\boldsymbol{v}}_1, q_1]) \rangle_{\Gamma_N} = L([\bar{\boldsymbol{v}}, q]) + \langle \bar{\boldsymbol{v}}, \boldsymbol{t} \rangle_{\Gamma_N}, \qquad (7.19) \\
&\sum_{i=1}^{2} \langle \mathcal{L}_i([\bar{\boldsymbol{u}}_i, p_i]), \tilde{\boldsymbol{v}}_i \rangle_{\Omega_i} + \sum_{i=1}^{2} B_i([\tilde{\boldsymbol{u}}_i, 0], [\tilde{\boldsymbol{v}}_i, 0]) - \sum_{i=1}^{2} \left\langle \tilde{\boldsymbol{\lambda}}_i, \tilde{\boldsymbol{v}}_i \right\rangle_{\Gamma} \\
&\qquad - \left\langle \tilde{\boldsymbol{\lambda}}_N, \tilde{\boldsymbol{v}}_1 \right\rangle_{\Gamma_N} = \sum_{i=1}^{2} L_i([\tilde{\boldsymbol{v}}_i, 0]), \qquad (7.20) \\
&\sum_{i=1}^{2} \left\langle \tilde{\boldsymbol{\kappa}}, \mathcal{T}_i([\bar{\boldsymbol{u}}_i, p_i]) + \tilde{\boldsymbol{\lambda}}_i \right\rangle_{\Gamma} + \left\langle \tilde{\boldsymbol{\kappa}}, \mathcal{T}_1([\bar{\boldsymbol{u}}_1, p_1]) + \tilde{\boldsymbol{\lambda}}_N \right\rangle_{\Gamma_N} = \langle \tilde{\boldsymbol{\kappa}}, \boldsymbol{t} \rangle_{\Gamma_N}, \qquad (7.21) \\
&\sum_{i=1}^{2} \langle \tilde{\boldsymbol{\mu}}_i, \tilde{\boldsymbol{\gamma}} - \tilde{\boldsymbol{u}}_i \rangle_{\Gamma} + \langle \tilde{\boldsymbol{\mu}}_1, \tilde{\boldsymbol{\gamma}} - \tilde{\boldsymbol{u}}_1 \rangle_{\Gamma_N} = 0, \qquad (7.22)
\end{aligned}$$

for all $\bar{\boldsymbol{v}}_i \in V_i$, $\tilde{\boldsymbol{v}}_i \in \tilde{V}_i$, $q_i \in Q_i$, $\tilde{\boldsymbol{\kappa}} \in \tilde{T}$, $\tilde{\boldsymbol{\mu}}_i \in \tilde{F}_i$ $(i = 1, 2)$.

This is the hybrid formulation on which we will base the finite element approximation described in the following. Its importance relies on the fact that *it is the theoretical framework to develop approximations in which $\boldsymbol{u}$ is split into a contribution which is continuous on Γ and another one which is discontinuous*. In Chapter 6 we presented a similar development for the convection-diffusion equation. Now we have detailed this development for the Stokes problem, considering also the presence of Neumann type boundary conditions.

7.3 Finite element approximation

7.3.1 Scale splitting

Let $\mathcal{P}_h := \{K\}$ be a finite element partition of the domain Ω of size h, and $V_h \times Q_h$ a finite element space where an approximate solution $[\boldsymbol{u}_h, p_h] \in V_h \times Q_h$ is sought. We assume that V_h is made *of continuous functions*, that is to say, V_h is conforming in V.

Consider the setting of the previous subsection with $\bar{V} = V_h$, and therefore $V = V_h \oplus \tilde{V}$, with $\tilde{V}$ to be defined, and $\boldsymbol{u} = \boldsymbol{u}_h + \tilde{\boldsymbol{u}}$, $\boldsymbol{v} = \boldsymbol{v}_h + \tilde{\boldsymbol{v}}$. The extension of (7.19)-(7.22) to multiple subdomains is straightforward. In particular, we will apply it considering each element a subdomain. No subscript will be used for the functions and operators in play to characterize the element domain over which they are defined, being this clear simply by the domain of integration.

The discrete variational problem to be considered is to find $[\boldsymbol{u}_h, p_h] \in V_h \times Q_h$, $\tilde{\boldsymbol{u}} \in \tilde{V}$, $\tilde{\boldsymbol{\gamma}} \in \tilde{T}$ and $\tilde{\boldsymbol{\lambda}} \in \tilde{F}$ such that

$$B([\boldsymbol{u}_h, p_h], [\boldsymbol{v}_h, q_h]) + \sum_K \langle \tilde{\boldsymbol{u}}, \mathcal{L}^*([\boldsymbol{v}_h, q_h]) \rangle_K + \sum_K \langle \tilde{\boldsymbol{\gamma}}, \mathcal{T}^*([\boldsymbol{v}_h, q_h]) \rangle_{\partial K} = L([\boldsymbol{v}_h, q]) + \langle \boldsymbol{v}_h, \boldsymbol{t} \rangle_{\Gamma_N}, \tag{7.23}$$

$$\sum_K \langle \mathcal{L}([\boldsymbol{u}_h, p_h]), \tilde{\boldsymbol{v}} \rangle_K + \sum_K B_K([\tilde{\boldsymbol{u}}, 0], [\tilde{\boldsymbol{v}}, 0]) - \sum_K \left\langle \tilde{\boldsymbol{\lambda}}, \tilde{\boldsymbol{v}} \right\rangle_{\partial K} = \sum_K L_K([\tilde{\boldsymbol{v}}, 0]), \tag{7.24}$$

$$\sum_K \left\langle \tilde{\boldsymbol{\kappa}}, \mathcal{T}([\boldsymbol{u}_h, p_h]) + \tilde{\boldsymbol{\lambda}} \right\rangle_{\partial K} = \langle \tilde{\boldsymbol{\kappa}}, \boldsymbol{t} \rangle_{\Gamma_N}, \tag{7.25}$$

$$\sum_K \langle \tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\gamma}} - \tilde{\boldsymbol{u}} \rangle_{\partial K} = 0, \tag{7.26}$$

for all $\boldsymbol{v}_h \in V_h$, $\tilde{\boldsymbol{v}} \in \tilde{V}$, $q_h \in Q_h$, $\tilde{\boldsymbol{\kappa}} \in \tilde{T}$, $\tilde{\boldsymbol{\mu}} \in \tilde{F}$, where $\tilde{T}$ is now the space of traces (of subscales) on the element boundaries (satisfying $\tilde{\boldsymbol{\gamma}} = \boldsymbol{0}$ on Γ_D) and $\tilde{F}$ the space of fluxes on these boundaries.

Apart from the imposition of the condition that $p_h \in Q_h$, problem (7.23)-(7.26) is exact. The final approximation is obtained by choosing a way to approximate the velocity subscales $\tilde{\boldsymbol{u}}$, their traces on the element boundaries $\tilde{\boldsymbol{\gamma}}$ and their fluxes $\tilde{\boldsymbol{\lambda}}$. This leaves many possibilities open. In particular, if $\tilde{\boldsymbol{u}}$ is chosen to be a piecewise polynomial, the previous equations constitute a very general framework to develop finite element approximations with a continuous component ($\boldsymbol{u}_h$) and a discontinuous one ($\tilde{\boldsymbol{u}}$). Traces ($\tilde{\boldsymbol{\gamma}}$) and fluxes ($\tilde{\boldsymbol{\lambda}}$) may be approximated independently or linked to $\tilde{\boldsymbol{u}}$ and/or $\boldsymbol{u}_h$ if an irreducible formulation is to be used. Note that if the first option is used there might be compatibility conditions between the approximating finite element spaces to render the final discrete problem numerically stable.

Our purpose here is *not* to exploit the possibilities of (7.23)-(7.26), but *to propose a closed form expression for $\tilde{\boldsymbol{u}}$, $\tilde{\boldsymbol{\gamma}}$ and $\tilde{\boldsymbol{\lambda}}$*. Only (7.23) will remain unaltered, but with a certain approximation for $\tilde{\boldsymbol{u}}$ in the interior of the element domains required to evaluate the second term on the left-hand-side of this equation, and an approximation for $\tilde{\boldsymbol{\gamma}}$ on the element boundaries to evaluate the third term. We proceed to explain how we do this in the following subsection, understanding that other possibilities are open within the present framework.

7.3.2 Subscales on the element boundaries

The way we propose to approximate the subscales was already presented in Chapter 6. Let us just recall the resulting expressions for $\tilde{\boldsymbol{\lambda}}$ and $\tilde{\boldsymbol{\gamma}}$.

Approximation of $\tilde{\boldsymbol{\lambda}}$. The values of $\tilde{\boldsymbol{\lambda}}$ on ∂K are *weak* approximations to the fluxes of $\tilde{\boldsymbol{u}}$. Given the trace $\tilde{\boldsymbol{\gamma}}$ of this unknown, and taking into account that no pressure subscales have been introduced, we propose the following closed form expression for $\tilde{\boldsymbol{\lambda}}$:

$$\tilde{\boldsymbol{\lambda}}_{\partial K_i \cap E} \approx \frac{\mu}{\delta}(\tilde{\boldsymbol{\gamma}} - \tilde{\boldsymbol{u}}_i), \quad i = 1, 2, \tag{7.27}$$

where now $\tilde{\boldsymbol{u}}_i$ has to be understood as the subscale computed in the element interiors and evaluated at edge E.

Approximation of $\tilde{\boldsymbol{\gamma}}$. Equation (7.25) states the weak continuity of the total fluxes on the element boundaries. The idea now is to replace this equation by an explicit prescription of this continuity. We need to distinguish the case in which an edge is interior to the domain Ω and the case in which it belongs to Γ_N. Edges on Γ_D will not contribute because of the zero velocity prescription there.

Let $[\![\boldsymbol{n} g]\!]_E := \boldsymbol{n}_1 g|_{\partial K_1 \cap E} + \boldsymbol{n}_2 g|_{\partial K_2 \cap E}$ denote the jump of a scalar function g across edge E and $[\![\partial_n g]\!]_E = \boldsymbol{n}_1 \cdot \nabla g|_{\partial K_1 \cap E} + \boldsymbol{n}_2 \cdot \nabla g|_{\partial K_2 \cap E}$ the jump of the normal derivative. For a vector field $\boldsymbol{v}$, we also define $[\![\boldsymbol{n} \otimes \boldsymbol{v}]\!]_E = \boldsymbol{n}_1 \otimes \boldsymbol{v}|_{\partial K_1 \cap E} + \boldsymbol{n}_2 \otimes \boldsymbol{v}|_{\partial K_2 \cap E}$.

Consider first the case in which E_0 is an interior edge. The condition to determine the expression of the subscale velocity on the boundary is that the normal component of the stress be continuous across interelement boundaries. This can be written as follows:

$$\begin{aligned} \boldsymbol{0} &= [\![-p\boldsymbol{n} + \mu \partial_n \boldsymbol{u}]\!]_{E_0} \\ &\approx [\![-p_h \boldsymbol{n} + \mu \partial_n \boldsymbol{u}_h]\!]_{E_0} + \tilde{\boldsymbol{\lambda}}_{\partial K_1 \cap E_0} + \tilde{\boldsymbol{\lambda}}_{\partial K_2 \cap E_0} \\ &\approx [\![-p_h \boldsymbol{n} + \mu \partial_n \boldsymbol{u}_h]\!]_{E_0} + \frac{\mu}{\delta} \left(2\tilde{\boldsymbol{\gamma}}_{E_0} - \tilde{\boldsymbol{u}}_1 - \tilde{\boldsymbol{u}}_2 \right), \end{aligned} \tag{7.28}$$

from where the approximation we propose is

$$\boxed{\tilde{\boldsymbol{\gamma}}_{E_0} \approx \{\tilde{\boldsymbol{u}}\}_{E_0} - \frac{\delta}{2\mu} [\![\mu \partial_n \boldsymbol{u}_h - p_h \boldsymbol{n}]\!]_{E_0},} \tag{7.29}$$

where $\{\tilde{\boldsymbol{u}}\}_{E_0} := \frac{1}{2}(\tilde{\boldsymbol{u}}_1 + \tilde{\boldsymbol{u}}_2)$ is the average of the subscales computed in the element interiors evaluated at edge E_0. From (7.29) it is observed that δ_0 will play the role of an algorithmic parameter for which, following our approach, we have a geometrical interpretation.

From now onwards we will use the symbol $=$ instead of $\approx$, understanding that in some places we perform approximation (7.27) that has led us to (7.29).

Let us consider now a boundary edge of the form $E_N = \partial K \cap \Gamma_N$ for a certain element K, where the Neumann condition (7.4) is prescribed. In this case, (7.28) has to be replaced by

$$\begin{aligned} \boldsymbol{t}|_{E_N} &= (-p\boldsymbol{n} + \mu \partial_n \boldsymbol{u})|_{E_N} \\ &= (-p_h \boldsymbol{n} + \mu \partial_n \boldsymbol{u}_h)|_{E_N} + \mu \partial_n \tilde{\boldsymbol{u}}|_{E_N} \\ &= (-p_h \boldsymbol{n} + \mu \partial_n \boldsymbol{u}_h)|_{E_N} + \frac{\mu}{\delta} \left(\tilde{\boldsymbol{\gamma}}_{E_N} - \tilde{\boldsymbol{u}}_{E_N} \right), \end{aligned}$$

from where

$$\boxed{\tilde{\boldsymbol{\gamma}}_{E_N} = \tilde{\boldsymbol{u}}_{E_N} - \frac{\delta}{\mu}(-p_h \boldsymbol{n} + \mu \partial_n \boldsymbol{u}_h - \boldsymbol{t})|_{E_N}.} \tag{7.30}$$

Problem for $\boldsymbol{u}_h$ and $\tilde{\boldsymbol{u}}$. From the approximation of the fluxes (7.27) and the expressions obtained for the traces, (7.29) and (7.30), one can obtain a problem for $\boldsymbol{u}_h$ and $\tilde{\boldsymbol{u}}$ alone from (7.23) and (7.24). After some algebraic manipulations, the problem obtained is: find $[\boldsymbol{u}_h, p_h] \in V_h \times Q_h$ and $\tilde{\boldsymbol{u}} \in \tilde{V}$ such that

$$
\begin{aligned}
&B([\boldsymbol{u}_h, p_h], [\boldsymbol{v}_h, q_h]) + \sum_K \langle \tilde{\boldsymbol{u}}, \mathcal{L}^*([\boldsymbol{v}_h, q_h]) \rangle_K \\
&\quad + \sum_{E_0} \langle \{\tilde{\boldsymbol{u}}\}, [\![\mathcal{T}^*([\boldsymbol{v}_h, q_h])]\!] \rangle_{E_0} - \frac{\delta}{2\mu} \sum_{E_0} \langle [\![\mathcal{T}([\boldsymbol{u}_h, p_h])]\!], [\![\mathcal{T}^*([\boldsymbol{v}_h, q_h])]\!] \rangle_{E_0} \\
&\quad + \sum_{E_N} \langle \tilde{\boldsymbol{u}}, \mathcal{T}^*([\boldsymbol{v}_h, q_h]) \rangle_{E_N} - \frac{\delta}{\mu} \sum_{E_N} \langle \mathcal{T}([\boldsymbol{u}_h, p_h]), \mathcal{T}^*([\boldsymbol{v}_h, q_h]) \rangle_{E_N} \\
&\quad = L([\boldsymbol{v}_h, q_h]) + \langle \boldsymbol{v}_h, \boldsymbol{t} \rangle_{\Gamma_N} - \frac{\delta}{\mu} \sum_{E_N} \langle \boldsymbol{t}, \mathcal{T}^*([\boldsymbol{v}_h, q_h]) \rangle_{E_N}, \qquad (7.31) \\
&\sum_K B_K([\tilde{\boldsymbol{u}}, 0], [\tilde{\boldsymbol{v}}, 0]) + \sum_K \langle \mathcal{L}([\boldsymbol{u}_h, p_h]), \tilde{\boldsymbol{v}} \rangle_K \\
&\quad + \sum_{E_0} \langle [\![\mathcal{T}([\boldsymbol{u}_h, p_h])]\!], \{\tilde{\boldsymbol{v}}\} \rangle_{E_0} + \frac{\delta}{2\mu} \sum_{E_0} \langle [\![\boldsymbol{n} \otimes \tilde{\boldsymbol{u}}]\!], [\![\boldsymbol{n} \otimes \tilde{\boldsymbol{v}}]\!] \rangle_{E_0} \\
&\quad + \sum_{E_N} \langle \mathcal{T}([\boldsymbol{u}_h, p_h]), \tilde{\boldsymbol{v}} \rangle_{E_N} = \sum_K L_K([\tilde{\boldsymbol{v}}, 0]) + \sum_{E_N} \langle \boldsymbol{t}, \tilde{\boldsymbol{v}} \rangle_{E_N}, \qquad (7.32)
\end{aligned}
$$

for all $[\boldsymbol{v}_h, q_h] \in V_h \times Q_h$, $\tilde{\boldsymbol{v}} \in \tilde{V}$.

Problem (7.31)-(7.32) is very general and could be used as such after choosing an approximation for $\tilde{V}$. However, we will further simplify the problem by approximating directly $\tilde{\boldsymbol{u}}$.

7.3.3 Subscales in the element interiors

To approximate $\tilde{\boldsymbol{u}}$, we in fact approximate (7.24) by integrating by parts,

$$
\sum_K B_K([\tilde{\boldsymbol{u}}, 0], [\tilde{\boldsymbol{v}}, 0]) = -\mu \sum_K \langle \Delta \tilde{\boldsymbol{u}}, \tilde{\boldsymbol{v}} \rangle_K + \mu \sum_K \langle \boldsymbol{n} \cdot \nabla \tilde{\boldsymbol{u}}, \tilde{\boldsymbol{v}} \rangle_{\partial K},
$$

assuming that $\mu \boldsymbol{n} \cdot \nabla \tilde{\boldsymbol{u}}$ cancels with the fluxes $\tilde{\boldsymbol{\lambda}}$ and using the crucial approximation

$$
\langle -\mu \Delta \tilde{\boldsymbol{u}}, \tilde{\boldsymbol{v}} \rangle_K \approx \tau^{-1} \langle \tilde{\boldsymbol{u}}, \tilde{\boldsymbol{v}} \rangle_K, \quad \tau^{-1} = C_1 \frac{\mu}{h^2}, \qquad (7.33)
$$

where C_1 is an algorithmic constant. We will not justify this last step, which is the keystone of stabilized finite element methods. It can be motivated for example by using an approximate Fourier analysis [32].

Summarizing, the subscales in the element interiors can be expressed in terms of $[\boldsymbol{u}_h, p_h]$ from the equation

$$
\sum_K \langle \mathcal{L}([\boldsymbol{u}_h, p_h]), \tilde{\boldsymbol{v}} \rangle_K + \tau^{-1} \sum_K \langle \tilde{\boldsymbol{u}}, \tilde{\boldsymbol{v}} \rangle_K = \rho \sum_K \langle \boldsymbol{f}, \tilde{\boldsymbol{v}} \rangle_K, \qquad (7.34)
$$

which can be described by saying the $\tilde{\boldsymbol{u}}$ is the projection of the residual $\rho\boldsymbol{f} - \mathcal{L}([\boldsymbol{u}_h, p_h])$ within each element multiplied by τ onto the space of subscales $\tilde{V}$. The most usual option is to take this projection as the identity (assuming this is feasible), although we favor the choice of taking it as the projection L^2-orthogonal to the finite element space V_h. This leads to the so called orthogonal subscale stabilization (OSS) method [32, 33]. However, the final method is independent of the choice of the space of subscales.

7.3.4 Stabilized finite element problem

The subscales in the element interiors can be approximated as stated in Chapter 6. With all the approximations introduced, the problem to be solved consists of (7.31) and (7.34). However, the approximations used to arrive to (7.34) have as a consequence the loss of symmetry of the problem (which is in fact observed using $-q_h$ as test function). This symmetry can be recovered neglecting the third and fifth terms in (7.31). Note that this maintains the consistency of the method, in the sense that if the approximate solution $[\boldsymbol{u}_h, p_h]$ is replaced by the exact solution $[\boldsymbol{u}, p]$, the discrete variational problem holds exactly.

The variation of the method just explained consists in finding $[\boldsymbol{u}_h, p_h] \in V_h \times Q_h$ and $\tilde{\boldsymbol{u}} \in \tilde{V}$ such that

$$
\begin{aligned}
&B([\boldsymbol{u}_h, p_h], [\boldsymbol{v}_h, q_h]) + \sum_K \langle \tilde{\boldsymbol{u}}, \mathcal{L}^*([\boldsymbol{v}_h, q_h])\rangle_K \\
&\quad - \frac{\delta}{2\mu} \sum_{E_0} \langle [\![\mathcal{T}([\boldsymbol{u}_h, p_h])]\!], [\![\mathcal{T}^*([\boldsymbol{v}_h, q_h])]\!] \rangle_{E_0} - \frac{\delta}{\mu} \sum_{E_N} \langle \mathcal{T}([\boldsymbol{u}_h, p_h]), \mathcal{T}^*([\boldsymbol{v}_h, q_h])\rangle_{E_N} \\
&\quad = L([\boldsymbol{v}_h, q_h]) + \langle \boldsymbol{v}_h, \boldsymbol{t}\rangle_{\Gamma_N} - \frac{\delta}{\mu} \sum_{E_N} \langle \boldsymbol{t}, \mathcal{T}^*([\boldsymbol{v}_h, q_h])\rangle_{E_N}, \qquad (7.35)
\end{aligned}
$$

$$
\sum_K \langle \mathcal{L}([\boldsymbol{u}_h, p_h]), \tilde{\boldsymbol{v}}\rangle_K + \tau^{-1} \sum_K \langle \tilde{\boldsymbol{u}}, \tilde{\boldsymbol{v}}\rangle_K = \rho \sum_K \langle \boldsymbol{f}, \tilde{\boldsymbol{v}}\rangle_K, \qquad (7.36)
$$

for all $[\boldsymbol{v}_h, q_h] \in V_h \times Q_h$, $\tilde{\boldsymbol{v}} \in \tilde{V}$.

We may write the solution of (7.36) as

$$
\tilde{\boldsymbol{u}} = \tau \tilde{P}(\rho\boldsymbol{f} - \mathcal{L}([\boldsymbol{u}_h, p_h])),
$$

where $\tilde{P}$ denotes the L^2 projection onto the space of subscales, which will be left undefined (except in the numerical examples, of course). The problem can now be written in a compact form, only involving the finite element component of the unknown $[\boldsymbol{u}_h, p_h]$, as follows: find $[\boldsymbol{u}_h, p_h] \in V_h \times Q_h$ such that

$$
B_{\text{stab}}([\boldsymbol{u}_h, p_h], [\boldsymbol{v}_h, q_h]) = L_{\text{stab}}([\boldsymbol{v}_h, q_h]) \quad \forall [\boldsymbol{v}_h, q_h] \in V_h \times Q_h,
$$

where

$$B_{\text{stab}}([\boldsymbol{u}_h, p_h], [\boldsymbol{v}_h, q_h]) = B([\boldsymbol{u}_h, p_h], [\boldsymbol{v}_h, q_h]) - \sum_K \tau \langle \tilde{P}(\mathcal{L}([\boldsymbol{u}_h, p_h])), \mathcal{L}^*([\boldsymbol{v}_h, q_h]) \rangle_K$$
$$- \frac{\delta}{2\mu} \sum_{E_0} \langle [\![\mathcal{T}([\boldsymbol{u}_h, p_h])]\!], [\![\mathcal{T}^*([\boldsymbol{v}_h, q_h])]\!] \rangle_{E_0} - \frac{\delta}{\mu} \sum_{E_N} \langle \mathcal{T}([\boldsymbol{u}_h, p_h]), \mathcal{T}^*([\boldsymbol{v}_h, q_h]) \rangle_{E_N}, \tag{7.37}$$

$$L_{\text{stab}}([\boldsymbol{v}_h, q_h]) = L([\boldsymbol{v}_h, q_h]) + \langle \boldsymbol{v}_h, \boldsymbol{t} \rangle_{\Gamma_N}$$
$$- \sum_K \tau \langle \tilde{P}(\rho \boldsymbol{f}), \mathcal{L}^*([\boldsymbol{v}_h, q_h]) \rangle_K - \frac{\delta}{\mu} \sum_{E_N} \langle \boldsymbol{t}, \mathcal{T}^*([\boldsymbol{v}_h, q_h]) \rangle_{E_N}. \tag{7.38}$$

7.4 Interaction between subdomains

7.4.1 Motivation

The stabilized finite element formulation presented in the previous section has been designed to allow arbitrary velocity-pressure interpolations, in particular discontinuous pressures. However, the concepts used to obtain it can be applied to other situations. In particular, we consider in this section the application to the interaction between two subdomains, in both of which the Stokes problem is solved.

The motivation to use the stabilization strategy in interaction problems arises from the fact that if the subdomains are discretized independently, the pressure degrees of freedom at the interface will be doubled and, therefore, pressure will be discontinuous at this interface. If a method that is stable for continuous pressures is applied (either coming from a stabilized formulation or from the use of inf-sup stable velocity-pressure pairs) there is no guarantee that this stability will be preserved at the interface. The use of the approach described in the previous section, known to be stable for arbitrary pressure interpolations, may thus be beneficial.

7.4.2 Continuous problem

The final discrete problem to be proposed can be derived directly using the ideas presented in the previous section and extended to the case in which the physical properties, and in particular the viscosity μ, are discontinuous. However, additional insight on the method is gained if a more "physical" approach is used when two subdomains interact.

Let us consider again the situation of Fig. 7.1, now for simplicity with $\Gamma_N = \emptyset$. For our purposes, instead of using a three-field hybrid formulation, using the primal unknown, its traces and its fluxes as variables, it is enough to consider the more common approach of using only the fluxes on Γ as unknowns, and enforcing continuity weakly. The boundary value problem

to be solved consists in finding $[\boldsymbol{u}_1, p_1]$, $[\boldsymbol{u}_2, p_2]$ and $\boldsymbol{\lambda}$ such that:

$$\begin{aligned}
-\mu_1 \Delta \boldsymbol{u}_1 + \nabla p_1 &= \rho \boldsymbol{f} && \text{in } \Omega_1, \\
\nabla \cdot \boldsymbol{u}_1 &= 0 && \text{in } \Omega_1, \\
\boldsymbol{u}_1 &= \boldsymbol{0} && \text{on } \Gamma_{D,1} = \partial\Omega_1 \cap \partial\Omega, \\
\boldsymbol{u}_1 &= \boldsymbol{u}_2 && \text{on } \Gamma, \\
\boldsymbol{\lambda} &= -p_1 \boldsymbol{n}_1 + \mu_1 \boldsymbol{n}_1 \cdot \nabla \boldsymbol{u}_1 && \text{on } \Gamma, \\
-\mu_2 \Delta \boldsymbol{u}_2 + \nabla p_2 &= \rho \boldsymbol{f} && \text{in } \Omega_2, \\
\nabla \cdot \boldsymbol{u}_2 &= 0 && \text{in } \Omega_2, \\
\boldsymbol{u}_2 &= \boldsymbol{0} && \text{on } \Gamma_{D,2} = \partial\Omega_2 \cap \partial\Omega, \\
-p_2 \boldsymbol{n}_2 + \mu_2 \boldsymbol{n}_2 \cdot \nabla \boldsymbol{u}_2 &= -\boldsymbol{\lambda} && \text{on } \Gamma.
\end{aligned}$$

These equations have been written in the order they can be solved in an iteration-by-subdomain strategy. The first four equations can be solved for $[\boldsymbol{u}_1, p_1]$ if $\boldsymbol{u}_2$ is assumed to be known on Γ, the flux on this surface can be then computed and used to solve the problem on Ω_2 with Neumann conditions on Γ.

The variational form of the continuous problem consists in finding $[\boldsymbol{u}_1, p_1]$, $[\boldsymbol{u}_2, p_2]$ and $\boldsymbol{\lambda}$ such that

$$\begin{aligned}
B_1([\boldsymbol{u}_1, p_1], [\boldsymbol{v}_1, q_1]) - \langle \boldsymbol{\lambda}, \boldsymbol{v}_1 \rangle_\Gamma &= L_1([\boldsymbol{v}_1, q_1]) \quad \forall [\boldsymbol{v}_1, q_1], \\
B_2([\boldsymbol{u}_2, p_2], [\boldsymbol{v}_2, q_2]) + \langle \boldsymbol{\lambda}, \boldsymbol{v}_2 \rangle_\Gamma &= L_2([\boldsymbol{v}_2, q_2]) \quad \forall [\boldsymbol{v}_2, q_2], \\
\langle \boldsymbol{\mu}, \boldsymbol{u}_1 - \boldsymbol{u}_2 \rangle_\Gamma &= 0 \quad \forall \boldsymbol{\mu},
\end{aligned}$$

where the bilinear and linear forms involved are the same as in the previous section. The spaces of unknowns and test functions are also the same as those introduced previously.

When applying the Galerkin method to discretize this problem there are at least two issues that have to be taken into account:

- The space for $\boldsymbol{\lambda}$ has to be properly chosen in order to obtain a numerically stable problem. There are compatibility conditions between the interpolation of this unknown and the interpolation of $\boldsymbol{u}$ and p that have to be met to satisfy the inf-sup conditions associated to the problem.

- It is preferable to compute $\boldsymbol{\lambda}_h$ *weakly* rather than from $\boldsymbol{\lambda} = -p_1 \boldsymbol{n}_1 + \mu_1 \boldsymbol{n}_1 \cdot \nabla \boldsymbol{u}_1$.

However, it is not our purpose to use the classical Galerkin method, but to extend the formulation of Section 7.3.

7.4.3 Finite element approximation

Let $B_{1,\mathrm{stab}}$, $L_{1,\mathrm{stab}}$, $B_{2,\mathrm{stab}}$ and $L_{2,\mathrm{stab}}$ be the stabilized bilinear and linear forms corresponding to each subdomain *without* considering the boundary conditions on Γ, which act as Neumann conditions on each subdomain. These forms are given by (7.37) and (7.38).

If $\boldsymbol{t}_i$ is the traction on Γ to be applied to Ω_i, the discrete variational equation on each subdomain reads:

$$B_{i,\mathrm{stab}}([\boldsymbol{u}_h,p_h],[\boldsymbol{v}_h,q_h]) - \frac{\delta}{\mu_i}\sum_{E_\Gamma}\langle \mathcal{T}_i([\boldsymbol{u}_h,p_h]),\mathcal{T}_i^*([\boldsymbol{v}_h,q_h])\rangle_{E_\Gamma}$$
$$= L_{i,\mathrm{stab}}([\boldsymbol{v}_h,q_h]) + \langle \boldsymbol{t}_i,\boldsymbol{v}_h\rangle_\Gamma - \frac{\delta}{\mu_i}\sum_{E_\Gamma}\langle \boldsymbol{t}_i,\mathcal{T}_i^*([\boldsymbol{v}_h,q_h])\rangle_{E_\Gamma}, \tag{7.39}$$

which holds for all test functions $[\boldsymbol{v}_h,q_h]$ with support on Ω_i, $i=1,2$. The edges E_Γ are now those contained in Γ.

Let us obtain which is the traction $\boldsymbol{t}_i$ that results from the formulation developed in the previous section. Note that $\boldsymbol{t}_1 = -\boldsymbol{\lambda}_2$ and $\boldsymbol{t}_2 = -\boldsymbol{\lambda}_1$. Recall that we have neglected the subscales in the element interiors (and evaluated on the boundary) when computing the fluxes $\tilde{\boldsymbol{\lambda}}$.

If the continuity of fluxes (7.28) is now imposed we find

$$\tilde{\boldsymbol{\gamma}}|_{E_\Gamma} = -\frac{\delta}{\mu_1+\mu_2}[\![\,\mathcal{T}([\boldsymbol{u}_h,p_h])\,]\!]_{E_\Gamma}.$$

Using the basic decomposition assumed for the total fluxes and (7.27) we obtain, on each edge E_Γ,

$$\boldsymbol{\lambda}_i = \mathcal{T}_i([\boldsymbol{u}_h,p_h]) + \tilde{\boldsymbol{\lambda}}_i = \mathcal{T}_i([\boldsymbol{u}_h,p_h]) + \frac{\mu_i}{\delta}\tilde{\boldsymbol{\gamma}}.$$

Combining the last two expressions yields, on each edge E_Γ,

$$\boxed{\boldsymbol{\lambda}_i = \mathcal{T}_i([\boldsymbol{u}_h,p_h]) - \frac{\mu_i}{\mu_1+\mu_2}[\![\,\mathcal{T}([\boldsymbol{u}_h,p_h])\,]\!].} \tag{7.40}$$

This expression for the traction associated to the formulation we propose has two interesting features:

- It *automatically* satisfies $\boldsymbol{\lambda}_1+\boldsymbol{\lambda}_2=\boldsymbol{0}$.
- Instead of the traction $\mathcal{T}_i([\boldsymbol{u}_h,p_h])$ associated to the standard Galerkin method, $\boldsymbol{\lambda}_i$ is a *weighted average* of $\mathcal{T}_1([\boldsymbol{u}_h,p_h])$ and $\mathcal{T}_2([\boldsymbol{u}_h,p_h])$, the weighting coefficients depending on the viscosity on each subdomain. If $i=1$, for example, we see that

  $$\boldsymbol{\lambda}_1 = \frac{\mu_2}{\mu_1+\mu_2}\mathcal{T}_1([\boldsymbol{u}_h,p_h]) + \frac{\mu_1}{\mu_1+\mu_2}(-\mathcal{T}_2([\boldsymbol{u}_h,p_h])),$$

  where $-\mathcal{T}_2([\boldsymbol{u}_h,p_h])$ can be understood as the traction associated to $[\boldsymbol{u}_h,p_h]$ in Ω_2 but computed with the normal $\boldsymbol{n}_1$.

It is worth to remark that (7.40) can be used to compute the fluxes in domain interaction problems as an alternative to the classical fluxes of the Galerkin method and also to the weak computation of these fluxes. It can be used not only in the case of meshes that match on Γ, but also in domain decomposition methods with overlapping (see [72]) or when fluxes are

needed on meshes that do not match the boundaries (as in the problems described in Chapter 4 and Chapter 5). Likewise, they can be modified to accommodate particular conditions that a certain application requires, such as conservation of angular momentum (using for example the methodology proposed in [71]).

The formulation we propose can finally be obtained adding up (7.39) for $i = 1$ and $i = 2$. Writing explicitly the expressions of $\mathcal{T}([\boldsymbol{u}_h, p_h])$ and $\mathcal{T}^*([\boldsymbol{v}_h, q_h])$, it consists of finding $[\boldsymbol{u}_h, p_h]$, defined on the whole computational domain Ω, such that

$$\begin{aligned} &B_{1,\text{stab}}([\boldsymbol{u}_h, p_h], [\boldsymbol{v}_h, q_h]) + B_{2,\text{stab}}([\boldsymbol{u}_h, p_h], [\boldsymbol{v}_h, q_h]) \\ &\quad - \sum_{E_\Gamma} \frac{\delta}{\mu_1 + \mu_2} \langle [\![\mu \partial_n \boldsymbol{u}_h - p_h \boldsymbol{n}]\!], [\![\mu \partial_n \boldsymbol{v}_h + q_h \boldsymbol{n}]\!] \rangle_{E_\Gamma} \\ &\quad = L_{1,\text{stab}}([\boldsymbol{v}_h, q_h]) + L_{2,\text{stab}}([\boldsymbol{v}_h, q_h]), \end{aligned} \tag{7.41}$$

for all test functions $[\boldsymbol{v}_h, q_h]$. It is observed that the term involving integrals over Γ penalizes the jump of the (pseudo-) tractions along this interface. We will observe this effect in the numerical examples.

7.4.4 Matrix structure

In order to write the matrix structure of problem (7.41), consider the splitting of the finite element velocity

$$\boldsymbol{u}_h = \boldsymbol{u}_{h,1} + \boldsymbol{u}_{h,\Gamma} + \boldsymbol{u}_{h,2},$$

where $\boldsymbol{u}_{h,i}$ refers to the component associated to the degrees of freedom *internal* to Ω_i, and vanishing on Γ, whereas $\boldsymbol{u}_{h,\Gamma}$ refers precisely to the degrees of freedom associated to the interacting boundary. This splitting in the one dimensional case and using linear elements is represented in Fig. 7.2. For the pressure, $p_{h,i}$ denotes simply its restriction to Ω_i.

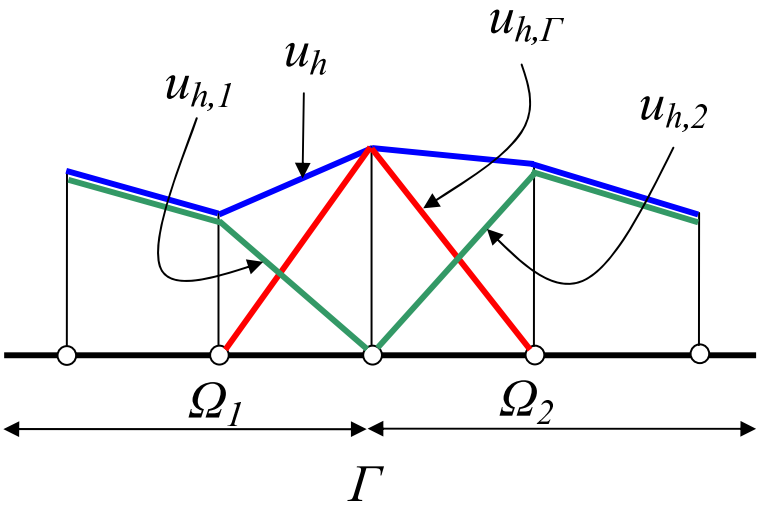


Figure 7.2: Splitting of the unknown

Having introduced this splitting, the matrix structure of the problem will be:

$$\begin{bmatrix} A_{1,1} & A_{1,\Gamma}+A'_{1,\Gamma} & A''_{1,2} \\ A_{\Gamma,1}+A'_{\Gamma,1} & A_{\Gamma,\Gamma}+A'_{\Gamma,\Gamma} & A_{\Gamma,2}+A'_{\Gamma,2} \\ A''_{2,1} & A_{2,\Gamma}+A'_{2,\Gamma} & A_{2,2} \end{bmatrix} \begin{bmatrix} U_1 \\ U_\Gamma \\ U_2 \end{bmatrix} = \begin{bmatrix} F_1 \\ F_\Gamma \\ F_2 \end{bmatrix}. \tag{7.42}$$

In this equation, U_i are arrays of degrees of freedom associated to $\boldsymbol{u}_{h,i}$ and $p_{h,i}$, and U_Γ the degrees of freedom associated to $\boldsymbol{u}_{h,\Gamma}$. The terms from where the different submatrices and components of the right-hand-side appear are obvious.

There are two remarks to be made referred to the algebraic problem (7.42):

- Submatrices with a prime and a double prime are due to the *new interaction term* in (7.41), which would not appear using a classical Gakerkin method for the domain interaction problem (even if stabilized finite element formulations are used within each subdomain).

- $A''_{1,2}$ and $A''_{2,1}$ appear *because of the jump of the derivatives* of the velocities. For example, there are test functions in Ω_1 which vanish on Γ but whose derivative does not vanish (in the case of Lagrangian interpolations, those are the test functions associated to the nodes adjacent to Γ). Thus, the jump of these derivatives is not zero and has to be multiplied against the jump of the velocity derivatives, which involves degrees of freedom of $\boldsymbol{u}_h$ *interior* to Ω_2 (again, in the case of Lagrangian interpolations, those are the velocity degrees of freedom associated to the nodes adjacent to Γ in the interior of Ω_2).

7.4.5 Iteration-by-subdomain strategy

The most popular way to deal with a problem involving the interaction of two subdomains is by using an iteration-by-subdomain strategy, that is to say, an iterative algorithm in which the unknowns are computed in one of the subdomains assuming the data from the other known, and proceeding iteratively until convergence.

To set possible iteration-by-subdomain schemes, it is convenient to consider first the matrix version of the problem. The simplest way to solve (7.42) is to solve for U_1 first and for U_Γ and U_2 in a coupled way. Denoting with a superscript the iteration counter, a solve of this iterative algorithm would be:

$$A_{1,1}U_1^{(i)} = F_1 - (A_{1,\Gamma}+A'_{1,\Gamma})U_\Gamma^{(i-1)} - A''_{1,2}U_2^{(i-1)}, \tag{7.43}$$

$$\begin{bmatrix} A_{\Gamma,\Gamma}+A'_{\Gamma,\Gamma} & A_{\Gamma,2}+A'_{\Gamma,2} \\ A_{2,\Gamma}+A'_{2,\Gamma} & A_{2,2} \end{bmatrix} \begin{bmatrix} U_\Gamma^{(i)} \\ U_2^{(i)} \end{bmatrix} = \begin{bmatrix} F_\Gamma - (A_{\Gamma,1}+A'_{\Gamma,1})U_1^{(i)} \\ F_2 - A''_{2,1}U_1^{(i)} \end{bmatrix}. \tag{7.44}$$

This scheme would in fact be of Gauss-Seidel type, since the value of U_1 just computed in the first step of the iteration is used in the second. A Jacobi-type scheme would be obtained replacing $U_1^{(i)}$ by $U_1^{(i-1)}$ in the second step.

Apart from the straightforward scheme (7.43)-(7.44), there are extensions and/or modifications that are convenient to use in the applications:

- Under-relaxation. Numerical experiments show that it is crucial to use under-relaxation. A simple scheme of the form

$$U_k^{(i)} \leftarrow \alpha U_k^{(i)} + (1-\alpha) U_k^{(i-1)} \tag{7.45}$$

  turns out to be very efficient. The values of the relaxation parameter α that we use are indicated in the numerical examples.

- Other iterative schemes, like GMRES. In principle, this type of schemes can be applied directly to (7.42) with an adequate choice of the preconditioner P. The key issue is to design this preconditioner in a modular way, that is to say, in such a manner that it requires only information of the domain whose unknowns are being computed. Our choice for the preconditioner P is:

$$P = \begin{bmatrix} A_{1,1} & A_{1,\Gamma} + A'_{1,\Gamma} & 0 \\ A_{\Gamma,1} + A'_{\Gamma,1} & A_{\Gamma,\Gamma} + A'_{\Gamma,\Gamma} & 0 \\ 0 & 0 & A_{22} \end{bmatrix},$$

  which leads to the following preconditioned system:

$$AP^{-1}PU = F \tag{7.46}$$

  The only system of equations to be solved in the GMRES iteration is the one associated to P^{-1}, in which we can separate terms associated to the problem in domain 1 from terms associated to the problem in domain 2 (see also [9, 10]).

The iterative scheme is better described using the algebraic form of the problem, but it is also enlightening to write the discrete variational version. The problem corresponding to (7.43)-(7.44) applied to (7.41) is:

$$
\begin{aligned}
&B_{1,\mathrm{stab}}([\boldsymbol{u}_{h,1}^{(i)}, p_{h,1}^{(i)}], [\boldsymbol{v}_{h,1}, q_{h,1}]) \\
&\quad = L_{1,\mathrm{stab}}([\boldsymbol{v}_{h,1}, q_{h,1}] - B_{1,\mathrm{stab}}([\boldsymbol{u}_{h,\Gamma}^{(i-1)}, 0], [\boldsymbol{v}_{h,1}, q_{h,1}])) \\
&\qquad + \sum_{E_\Gamma} \frac{\delta}{\mu_1+\mu_2} \Big\langle \mu_1 \partial_{n_1}(\boldsymbol{u}_{h,1}^{(i)} + \boldsymbol{u}_{h,\Gamma}^{(i-1)}) - p_{h,1}^{(i)} \boldsymbol{n}_1, \mu_1 \partial_{n_1} \boldsymbol{v}_{h,1} + q_{h,1} \boldsymbol{n}_1 \Big\rangle_{E_\Gamma} \\
&\qquad + \sum_{E_\Gamma} \frac{\delta}{\mu_1+\mu_2} \Big\langle \mu_2 \partial_{n_2}(\boldsymbol{u}_{h,\Gamma}^{(i-1)} + \boldsymbol{u}_{h,2}^{(i-1)}) - p_{h,2}^{(i-1)} \boldsymbol{n}_2, \mu_1 \partial_{n_1} \boldsymbol{v}_{h,1} + q_{h,1} \boldsymbol{n}_1 \Big\rangle_{E_\Gamma}, \\
&B_{2,\mathrm{stab}}([\boldsymbol{u}_{h,\Gamma}^{(i)} + \boldsymbol{u}_{h,2}^{(i)}, p_{h,2}^{(i)}], [\boldsymbol{v}_{h,\Gamma} + \boldsymbol{v}_{h,2}, q_{h,2}]) \\
&\quad = L_{2,\mathrm{stab}}([\boldsymbol{v}_{h,\Gamma} + \boldsymbol{v}_{h,2}, q_{h,2}]) - B_{2,\mathrm{stab}}([\boldsymbol{u}_{h,1}^{(i)}, p_{h,1}^{(i)}], [\boldsymbol{v}_{h,\Gamma} + \boldsymbol{v}_{h,2}, q_{h,2}]) \\
&\qquad + \sum_{E_\Gamma} \frac{\delta}{\mu_1+\mu_2} \Big\langle \mu_1 \partial_{n_1}(\boldsymbol{u}_{h,1}^{(i)} + \boldsymbol{u}_{h,\Gamma}^{(i)}) - p_{h,1}^{(i)} \boldsymbol{n}_1, \mu_1 \partial_{n_1} \boldsymbol{v}_{h,\Gamma} \Big\rangle_{E_\Gamma} \\
&\qquad + \sum_{E_\Gamma} \frac{\delta}{\mu_1+\mu_2} \Big\langle \mu_2 \partial_{n_2}(\boldsymbol{u}_{h,\Gamma}^{(i)} + \boldsymbol{u}_{h,2}^{(i)}) - p_{h,2}^{(i)} \boldsymbol{n}_2, \mu_1 \partial_{n_1} \boldsymbol{v}_{h,\Gamma} \Big\rangle_{E_\Gamma} \\
&\qquad + \sum_{E_\Gamma} \frac{\delta}{\mu_1+\mu_2} \Big\langle \mu_1 \partial_{n_1}(\boldsymbol{u}_{h,1}^{(i)} + \boldsymbol{u}_{h,\Gamma}^{(i)}) - p_{h,1}^{(i)} \boldsymbol{n}_1, \mu_2 \partial_{n_2}(\boldsymbol{v}_{h,\Gamma} + \boldsymbol{v}_{h,2}) + q_{h,2} \boldsymbol{n}_2 \Big\rangle_{E_\Gamma} \\
&\qquad + \sum_{E_\Gamma} \frac{\delta}{\mu_1+\mu_2} \Big\langle \mu_2 \partial_{n_2}(\boldsymbol{u}_{h,\Gamma}^{(i)} + \boldsymbol{u}_{h,2}^{(i)}) - p_{h,2}^{(i)} \boldsymbol{n}_2, \mu_2 \partial_{n_2}(\boldsymbol{v}_{h,\Gamma} + \boldsymbol{v}_{h,2}) + q_{h,2} \boldsymbol{n}_2 \Big\rangle_{E_\Gamma}.
\end{aligned}
$$

7.5 Fluid-structure interaction

In the previous section we have considered the interaction between two subdomains in both of which the Stokes problem is solved. In this sense, the situation can be considered as a *homogeneous* interaction. The problem to be solved in each subdomain is (7.39) and, since these equations are dimensionally homogeneous, they can be added up for $i = 1, 2$ to obtain (7.41). In this section however we are interested in the interaction between a *fluid* and a *solid*, and thus the problem can be termed as *heterogeneous*. In this case, it is better to work directly with (7.39). The purpose of what follows is to apply the ideas introduced previously to fluid-structure interaction (FSI) problems and to design an iteration-by-subdomain strategy for this particular problem.

7.5.1 Continuous problem

The nature of the problem to be considered is intrinsically transient (although it is obviously possible that a steady-state is reached). Let $[0, T]$ be the time interval of analysis. In all what follows we will use subscript F to refer to the fluid and subscript S to refer to the solid (and not subscripts 1 and 2, as in the previous section). In particular, Ω_F and Ω_S will be the subdomains occupied by the fluid and the solid, respectively, and $\Gamma = \partial\bar{\Omega}_F \cap \partial\bar{\Omega}_S$ their common boundary.

If $\boldsymbol{d} : [0,T] \times \Omega_S \longrightarrow \mathbb{R}^d$ is the displacement field in the solid, the problem to be solved consists in finding $\boldsymbol{d},\ \boldsymbol{u}$ and p such that

$$\begin{aligned}
\rho_S \partial_{tt}^2 \boldsymbol{d} - \nabla \cdot \boldsymbol{\sigma}_S &= \rho_S \boldsymbol{f} && \text{in } \Omega_S,\\
\boldsymbol{d} &= \boldsymbol{0} && \text{on } \Gamma_{D_S},\\
\boldsymbol{n}_S \cdot \boldsymbol{\sigma}_S &= \boldsymbol{t}_S && \text{on } \Gamma_{N_S},\\
\rho_F \partial_t \boldsymbol{u} - \mu \Delta \boldsymbol{u} + \nabla p &= \rho_F \boldsymbol{f} && \text{in } \Omega_F,\\
\nabla \cdot \boldsymbol{u} &= 0 && \text{in } \Omega_F,\\
\boldsymbol{u} &= \boldsymbol{0} && \text{on } \Gamma_{D_N},\\
-p\boldsymbol{n}_F + \mu \boldsymbol{n}_F \cdot \nabla \boldsymbol{u} &= \boldsymbol{t}_F && \text{on } \Gamma_{N_F},\\
\boldsymbol{n}_S \cdot \boldsymbol{\sigma}_S + (-p\boldsymbol{n}_F + \mu \boldsymbol{n}_F \cdot \nabla \boldsymbol{u}) &= \boldsymbol{0} && \text{on } \Gamma,\\
\partial_t \boldsymbol{d} - \boldsymbol{u} &= \boldsymbol{0} && \text{on } \Gamma,
\end{aligned}$$

together with initial conditions for $\boldsymbol{u}$, $\boldsymbol{d}$ and $\partial_t \boldsymbol{d}$ in the domain where they are defined. A linear elastic behavior will be assumed for the solid, so that the stress tensor there is given by

$$\boldsymbol{\sigma}_S = \boldsymbol{\sigma}_S(\boldsymbol{d}) = \boldsymbol{C} : \nabla^S \boldsymbol{d},$$

where $\boldsymbol{C}$ is the constitutive tensor and $\nabla^S \boldsymbol{d}$ the symmetrical gradient of $\boldsymbol{d}$.

To simplify the exposition, we assume that the solid is *not* incompressible, so that the problem can be approximated without the need to introduce the volumetric stress as a new variable (the extension to this situation would be straightforward and, in fact, we use it in the numerical examples). Therefore, the standard Galerkin method can be used to approximate the governing equations for the solid.

The variational counterpart of the FSI problem consists in finding $\boldsymbol{d},\ \boldsymbol{u},\ p$ and the interaction stress $\boldsymbol{\lambda}$ such that

$$\begin{aligned}
\rho_S(\partial_{tt}^2 \boldsymbol{d}, \boldsymbol{e})_{\Omega_S} + B_S(\boldsymbol{d}, \boldsymbol{e}) - \langle \boldsymbol{\lambda}, \boldsymbol{e} \rangle_\Gamma &= L_S(\boldsymbol{e}) + \langle \boldsymbol{t}_S, \boldsymbol{e} \rangle_{\Gamma_{N_S}} && \forall \boldsymbol{e} \in W,\\
\rho_F(\partial_t \boldsymbol{u}, \boldsymbol{v})_{\Omega_F} + B_F([\boldsymbol{u}, p], [\boldsymbol{v}, q]) + \langle \boldsymbol{\lambda}, \boldsymbol{v} \rangle_\Gamma &= L_F([\boldsymbol{v}, q]) + \langle \boldsymbol{t}_F, \boldsymbol{v} \rangle_{\Gamma_{N_F}} && \forall [\boldsymbol{v}, q] \in V \times Q,\\
\langle \boldsymbol{\mu}, \partial_t \boldsymbol{d} - \boldsymbol{u} \rangle_\Gamma &= 0 && \forall \boldsymbol{\mu} \in F,
\end{aligned}$$

where

$$\begin{aligned}
&W = \{\boldsymbol{e} \in H^1(\Omega_S)^d \mid \boldsymbol{e} = \boldsymbol{0} \text{ on } \Gamma_{D_S}\},\\
&B_S(\boldsymbol{d}, \boldsymbol{e}) = (\boldsymbol{C} : \nabla^S \boldsymbol{d}, \nabla^S \boldsymbol{e})_{\Omega_S},\\
&L_S(\boldsymbol{e}) = \rho_S \langle \boldsymbol{f}, \boldsymbol{e} \rangle_{\Omega_S},
\end{aligned}$$

and for each time $t \in (0,T)$ the unknowns satisfy $\boldsymbol{d} \in W$, $[\boldsymbol{u}, p] \in V \times Q$, $\boldsymbol{\lambda} \in F$ (with the adequate regularity in time), with the appropriate initial conditions at $t = 0$.

7.5.2 Finite element approximation and interaction stresses

Once finite element spaces $W_h \subset W$, $\boldsymbol{V}_h \times Q_h \subset V \times Q$ are chosen, the crucial issue to extend the formulation of the previous section to the present FSI problem in to obtain the interaction

stresses resulting from the introduction of subscales on the element boundaries contained in Γ. Let $\mathcal{T}_S(\boldsymbol{e}) = \boldsymbol{n}_S \cdot \boldsymbol{\sigma}_S(\boldsymbol{e})$. The velocity subscales $\tilde{\boldsymbol{\gamma}}_F$ and the displacement subscales $\tilde{\boldsymbol{\gamma}}_S$ can be obtained from condition (7.28), which now can be written as follows:

$$\begin{aligned} \mathbf{0} &= \mathcal{T}_F([\boldsymbol{u}, p]) + \mathcal{T}_S(\boldsymbol{d}) \\ &\approx \mathcal{T}_F([\boldsymbol{u}_h, p_h]) + \mathcal{T}_F([\tilde{\boldsymbol{u}}, 0]) + \mathcal{T}_S(\boldsymbol{d}_h) + \mathcal{T}_S(\tilde{\boldsymbol{d}}), \end{aligned} \tag{7.47}$$

where $\tilde{\boldsymbol{d}}$ are the displacement subscales. Neglecting the subscales in the element interiors as before, $\mathcal{T}_F([\tilde{\boldsymbol{u}}, 0])$ can be approximated by $\frac{\mu}{\delta}\tilde{\boldsymbol{\gamma}}_F$. The problem is how to approximate $\mathcal{T}_S(\tilde{\boldsymbol{d}})$. Approximating derivatives using finite differences will yield an expression of the form $\mathcal{T}_S(\tilde{\boldsymbol{d}}) \approx \frac{1}{\delta}\boldsymbol{G}\tilde{\boldsymbol{\gamma}}_S$ for a certain matrix $\boldsymbol{G}$ depending on the physical parameters contained in the constitutive tensor $\boldsymbol{C}$. For our reasoning it is enough to approximate $\mathcal{T}_S(\tilde{\boldsymbol{d}}) \approx \frac{G^*}{\delta}\tilde{\boldsymbol{\gamma}}_S$, G^* being a scalar coefficient. Altogether, (7.47) yields, on each edge of Γ,

$$G^*\tilde{\boldsymbol{\gamma}}_S + \mu\tilde{\boldsymbol{\gamma}}_F = -\delta[\mathcal{T}_F([\boldsymbol{u}_h, p_h]) + \mathcal{T}_S(\boldsymbol{d}_h)]. \tag{7.48}$$

The compatibility between the velocity in the fluid and the displacement in the solid implies also the compatibility in the corresponding subscales, that is to say, $\tilde{\boldsymbol{\gamma}}_F = \partial_t\tilde{\boldsymbol{\gamma}}_S$.

In FSI problems of interest, we may assume that the fluid and solid physical properties are such that

$$G^*|\tilde{\boldsymbol{\gamma}}_S| \gg \mu|\partial_t\tilde{\boldsymbol{\gamma}}_S|,$$

which using (7.48) implies that

$$\tilde{\boldsymbol{\gamma}}_S \approx -\frac{\delta}{G^*}[\mathcal{T}_F([\boldsymbol{u}_h, p_h]) + \mathcal{T}_S(\boldsymbol{d}_h)], \quad \tilde{\boldsymbol{\gamma}}_F \approx \mathbf{0},$$

and, consequently,

$$\boldsymbol{\lambda}_S = \mathcal{T}_S(\boldsymbol{d}_h) + \frac{G^*}{\delta}\tilde{\boldsymbol{\gamma}}_S \approx -\mathcal{T}_F([\boldsymbol{u}_h, p_h]), \tag{7.49}$$

$$\boldsymbol{\lambda}_F = \mathcal{T}_F([\boldsymbol{u}_h, p_h]) + \tilde{\boldsymbol{\lambda}}_F \approx \mathcal{T}_F([\boldsymbol{u}_h, p_h]). \tag{7.50}$$

These approximations have an interesting consequence. Let $\boldsymbol{t}_{SF} = -\boldsymbol{\lambda}_F = -\mathcal{T}_F([\boldsymbol{u}_h, p_h])$ be the stress *exerted on the solid* because of the interaction with the fluid, and $\boldsymbol{t}_{FS} = -\boldsymbol{\lambda}_S = \mathcal{T}_F([\boldsymbol{u}_h, p_h])$ the stress *exerted on the fluid* because of the interaction with the solid. Suppose an iterative strategy is used to solve the fluid-solid coupling (within each time step, for example). If the solid is computed with a Neumann-type condition on Γ, $\boldsymbol{t}_{SF} = -\mathcal{T}_F([\boldsymbol{u}_h, p_h])$ has to be used as traction, which corresponds to the common approach: stresses computed in the fluid using the finite element solution are transmitted to the solid. In the fluid a Dirichlet boundary condition on Γ can be used once the displacements in the solid have been computed. However, *it is not possible* to use a Neumann condition on Γ when solving in the fluid domain, since the traction to be used is $\boldsymbol{t}_{FS} = \mathcal{T}_F([\boldsymbol{u}_h, p_h])$, which depends only on the velocities. Thus, the fluid would not "feel" the action exerted by the solid. This agrees with the well known fact that in FSI problems if a Dirichlet-Neumann coupling is used, Neumann boundary conditions have to be applied always to the solid surface, not to the fluid.

A similar situation is encountered in homogeneous interaction problems if one of the subdomains is much "stiffer" than the other. From (7.40) it is observed that if, for example, $\mu_1 \gg \mu_2$ then $\boldsymbol{\lambda}_1 = -\boldsymbol{\lambda}_2 \approx -\mathcal{T}_2([\boldsymbol{u}_h, p_h])$.

7.5.3 Fully discrete problem and iterative coupling

Suppose to simplify that time is discretized using a backward-difference formula, that we denote by D_t to approximate ∂_t and D_{tt} to approximate ∂_{tt}^2. Let δt be the time step size of a uniform partition of $[0,T]$. Consider that the unknowns are computed at time levels $0,1,2,...,n-1$ and we want to compute them at time $t^n = n\delta t$. The fully discrete version of the problem corresponding to (7.39) is: find $\boldsymbol{d}_h^n \in W_h$, $\boldsymbol{u}_h^n \in V_h$ and $p_h^n \in Q_h$ such that

$$\begin{aligned}
&\rho_S(D_{tt}\boldsymbol{d}_h^n, \boldsymbol{e}_h)_{\Omega_S} + B_S(\boldsymbol{d}_h^n, \boldsymbol{e}_h) - \frac{\delta}{G^*}\sum_{E_\Gamma}\langle \mathcal{T}_S(\boldsymbol{d}_h^n), \mathcal{T}_S(\boldsymbol{e}_h)\rangle_{E_\Gamma} \\
&\quad = L_S(\boldsymbol{e}_h) + \langle \boldsymbol{t}_{SF}, \boldsymbol{e}_h\rangle_\Gamma - \frac{\delta}{G^*}\sum_{E_\Gamma}\langle \boldsymbol{t}_{SF}, \mathcal{T}_S(\boldsymbol{e}_h)\rangle_{E_\Gamma}, \qquad (7.51)\\
&\rho_F(D_t\boldsymbol{u}_h^n, \boldsymbol{v}_h)_{\Omega_F} + B_{F,\mathrm{stab}}([\boldsymbol{u}_h^n, p_h^n], [\boldsymbol{v}_h, q_h]) - \frac{\delta}{\mu}\sum_{E_\Gamma}\langle \mathcal{T}_F([\boldsymbol{u}_h^n, p_h^n]), \mathcal{T}_F^*([\boldsymbol{v}_h, q_h])\rangle_{E_\Gamma} \\
&\quad = L_{F,\mathrm{stab}}([\boldsymbol{v}_h, q_h]) + \langle \boldsymbol{t}_{FS}, \boldsymbol{v}_h\rangle_\Gamma - \frac{\delta}{\mu}\sum_{E_\Gamma}\langle \boldsymbol{t}_{FS}, \mathcal{T}_F^*([\boldsymbol{v}_h, q_h])\rangle_{E_\Gamma}, \qquad (7.52)
\end{aligned}$$

which hold for all $\boldsymbol{e}_h \in W_h$, $[\boldsymbol{v}_h, q_h] \in V_h \times Q_h$. This is the *monolithic* fluid-structure system that we propose.

Of particular interest is the design of a simple iterative coupling between the solid and the fluid *using approximations (7.49)-(7.50)*. Let us denote by $f^{n,i}$ an approximation to an unknown f at time step n and iteration i, with the initialization $f^{n,0} = f^{n-1}$. Suppose that the solid is solved first, with $[\boldsymbol{u}_h^{n,i-1}, p_h^{n,i-1}]$ known. Then, $\boldsymbol{t}_{SF} = -\mathcal{T}_F([\boldsymbol{u}_h^{n,i-1}, p_h^{n,i-1}])$ can be used in (7.51) to compute $\boldsymbol{d}_h^{n,i}$. When solving for the fluid, the traction to be used must be $\boldsymbol{t}_{FS} = \mathcal{T}_F([\boldsymbol{u}_h^{n,i-1}, p_h^{n,i-1}])$, since *only in this case one can guarantee that* $\boldsymbol{t}_{SF} + \boldsymbol{t}_{FS} = \boldsymbol{0}$ *at each iteration*. Using this, and noting that $\boldsymbol{v}_h|_\Gamma = \boldsymbol{0}$ if Dirichlet conditions are used to solve in the fluid domain, the algorithm reads

$$\begin{aligned}
&\rho_S(D_{tt}\boldsymbol{d}_h^{n,i}, \boldsymbol{e}_h)_{\Omega_S} + B_S(\boldsymbol{d}_h^{n,i}, \boldsymbol{e}_h) - \frac{\delta}{G^*}\sum_{E_\Gamma}\big\langle \mathcal{T}_S(\boldsymbol{d}_h^{n,i}), \mathcal{T}_S(\boldsymbol{e}_h)\big\rangle_{E_\Gamma} \\
&\quad = L_S(\boldsymbol{e}_h) - \big\langle \mathcal{T}_F([\boldsymbol{u}_h^{n,i-1}, p_h^{n,i-1}]), \boldsymbol{e}_h\big\rangle_\Gamma + \frac{\delta}{G^*}\sum_{E_\Gamma}\big\langle \mathcal{T}_F([\boldsymbol{u}_h^{n,i-1}, p_h^{n,i-1}]), \mathcal{T}_S(\boldsymbol{e}_h)\big\rangle_{E_\Gamma}, \\
&\rho_F(D_t\boldsymbol{u}_h^{n,i}, \boldsymbol{v}_h)_{\Omega_F} + B_{F,\mathrm{stab}}([\boldsymbol{u}_h^{n,i}, p_h^{n,i}], [\boldsymbol{v}_h, q_h]) - \frac{\delta}{\mu}\sum_{E_\Gamma}\big\langle \mathcal{T}_F([\boldsymbol{u}_h^{n,i}, p_h^{n,i}]), \mathcal{T}_F^*([\boldsymbol{v}_h, q_h])\big\rangle_{E_\Gamma} \\
&\quad = L_{F,\mathrm{stab}}([\boldsymbol{v}_h, q_h]) - \frac{\delta}{\mu}\sum_{E_\Gamma}\big\langle \mathcal{T}_F([\boldsymbol{u}_h^{n,i-1}, p_h^{n,i-1}]), \mathcal{T}_F^*([\boldsymbol{v}_h, q_h])\big\rangle_{E_\Gamma},
\end{aligned} \qquad (7.53)$$

with the essential condition $\boldsymbol{u}_h^{n,i}|_\Gamma = D_t\boldsymbol{d}_h^{n,i}|_\Gamma$ for the second equation. It is observed that:

- The second term in the right-hand-side (RHS) of the first equation enforces the continuity of tractions between the solid and the fluid when tested by the displacement test function.

- The third term in the left-hand-side (LHS) and the third term in the RHS of the first equation enforce further this continuity, now by testing the tractions with $\mathcal{T}_S(\boldsymbol{e}_h)$. However, these terms are multiplied by $\frac{\delta}{G^*}$, which is very small when realistic physical properties are used. Thus, their effect is in practice negligible.

- The crucial term is the second one in the RHS of the second equation. If it were evaluated at iteration i, it would cancel with the third term in the LHS, which would in fact lead to the simplest fluid-structure iterative algorithm. However, evaluating it at $i-1$ allows us to guarantee that $\boldsymbol{t}_{SF} + \boldsymbol{t}_{FS} = \boldsymbol{0}$ at each iteration, as it has been said, and also acts as a penalization of the jump of fluid tractions between iterations, given by $\mathcal{T}_F([\boldsymbol{u}_h^{n,i}, p_h^{n,i}]) - \mathcal{T}_F([\boldsymbol{u}_h^{n,i-1}, p_h^{n,i-1}])$. The bottom line of our formulation applied to FSI iterative algorithms can be summarized by these terms, which can *a posteriori* be understood as a modification of the simplest iterative scheme (more sophisticated algorithms could also be used). Numerical experiments show that the improvement in convergence observed well deserves their derivation.

7.6 Numerical examples

In this section we present some numerical examples corresponding to the formulation presented in Sections 7.4 and 7.5. The ability of the method to use arbitrary discontinuous pressure interpolations for the pressure was already demonstrated in Chapter 6.

In all cases we will use the simplest choice $\tilde{P} = I$ in (7.38), where I is the identity (at least when applied to the residual of the finite element solution). This corresponds the most popular stabilized finite element method for the Stokes problem. All the examples have been run using continuous P_1 elements (linear triangles in 2D) for all variables. The algorithmic constant C_1 in (7.33) has been set to $C_1 = 4$. When approximating the elasticity equations, $G^* = E$, the Young modulus, has been chosen in (7.51).

7.6.1 Two examples of domain interaction

In this subsection we present the numerical results for two examples which illustrate the ideas presented in Section 7.4, one of a solid-solid interaction and another of a fluid-fluid interaction.

The first example we consider consists of two incompressible elastic bodies, which we will model by means of the Stokes equations, now μ being the shear modulus. The problem setting and subdomains can be seen in Fig. 7.3. Both bodies are incompressible (Poisson ratio $\nu = 0.5$), but the body on the top is 10 times stiffer (Young modulus $E = 3$) than the one below (Young modulus $E = 0.3$). The unstructured triangular mesh consisting of 1990 triangles used to solve the problem can be seen in Fig. 7.4.

The displacement and pressure fields obtained are shown in Fig. 7.5. We also depict the normal tractions on the solid body interface, which coincide with the component σ_{22} of the stress tensor, in Fig. 7.6. In can be observed that using subscales reduces the jump in tractions between both solid bodies, but the solution is stable and very similar whether subscales on the boundaries are used or not.

In the second example we will consider the stationary cavity flow example for the Stokes problem. The fluid domain is given by $\Omega = [0,1] \times [0,1]$. All the boundaries are set to null

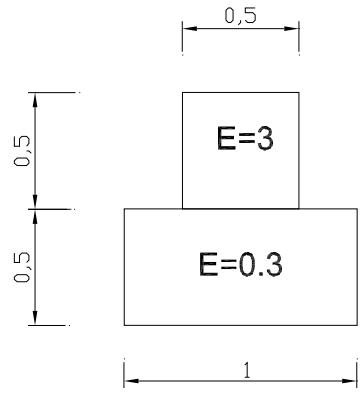


Figure 7.3: Incompressible elastic bodies. Problem setting.

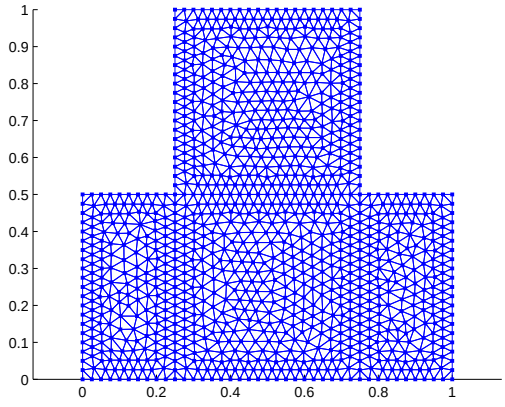


Figure 7.4: Incompressible elastic bodies. Finite element mesh.

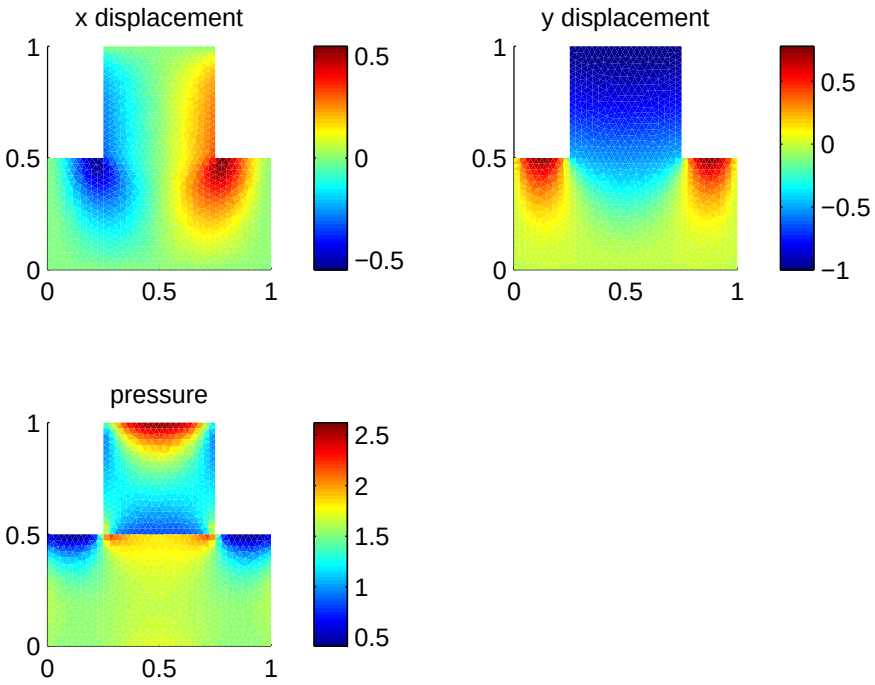


Figure 7.5: Incompressible elastic bodies. Displacement and pressure fields.

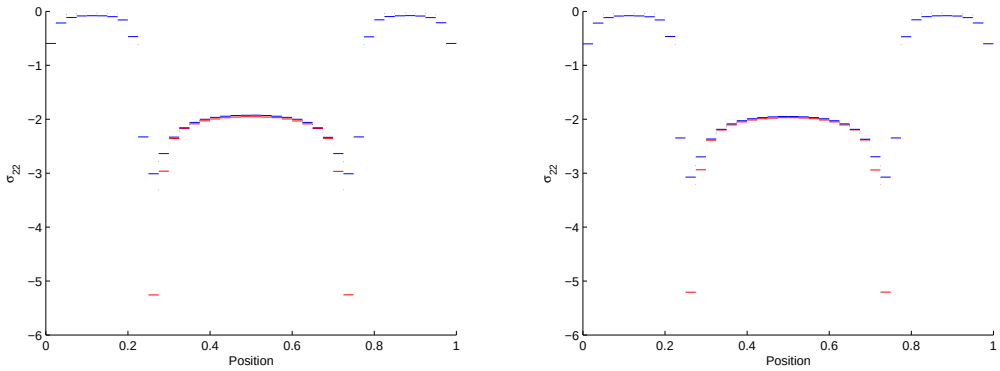


Figure 7.6: Incompressible elastic bodies. Normal tractions along the interface for the upper (red) and the lower (blue) solid bodies. Left: without subscales. Right: with subscales, $\delta_0 = 0.5$

velocity except for the one corresponding to $y = 1$, in which we impose a horizontal velocity $u_x = 1$ and a vertical velocity $u_y = 0$. The fluid viscosity is set to $\nu = 1$. A finite element mesh of 7200 structured triangles ($h = 0.16$) has been used.

We now divide the fluid domain in two: the first subdomain corresponds to $x < 0.1$, while the second subdomain corresponds to $x > 0.1$. We solve this numerical example by means of a domain decomposition method and the GMRES strategy described in Subsection 7.4.5.

The velocity and pressure fields for the Stokes cavity problem are depicted in Fig. 7.7. Since the fluid density and viscosity are the same in both subdomains, there should be no pressure jump in the boundary separating them. However, a pressure jump appears in the numerical solution which is due to the fact that extra pressure degrees of freedom have been added to the nodes belonging to the boundary. This pressure jump can be seen in Fig. 7.8. We can see that using subscales on the element boundaries helps reduce the pressure jump, and as a consequence the pressure field gets closer to the one we would obtain if a monolithic approach was used, in which the pressure field would be continuous.

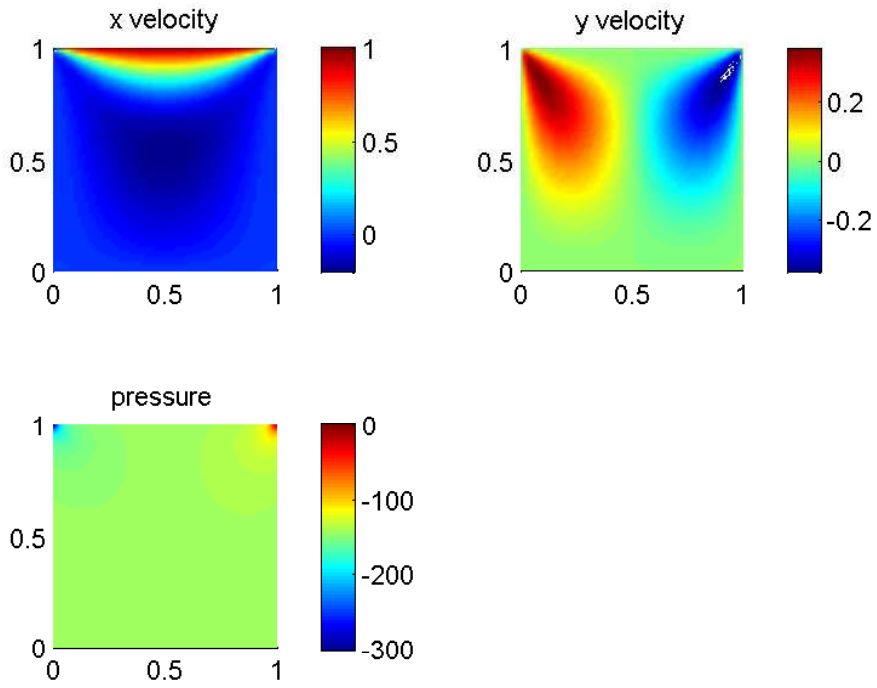


Figure 7.7: Velocity and pressure fields for the Stokes cavity problem

7.6.2 Added-mass effect

In this section we present some numerical results which illustrate the behavior of the subscales on the element boundaries as strategy to alleviate the added mass effect. In order to do so, we will use the example proposed in [24], in which we will couple the 2D Stokes equations with the linear elasticity equations.

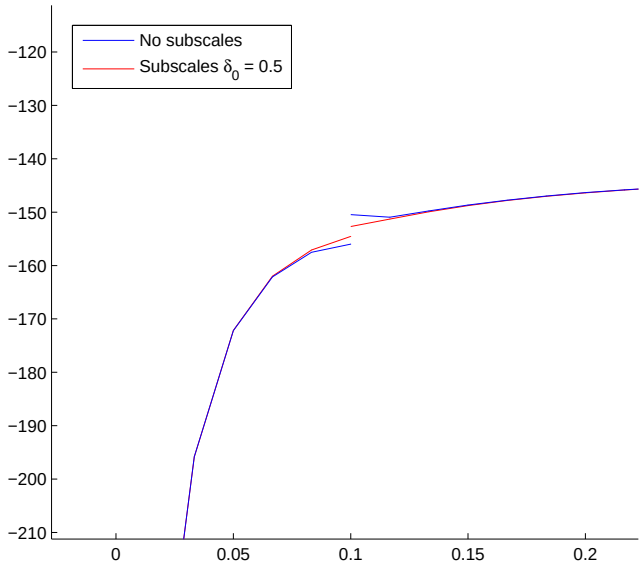


Figure 7.8: Pressure jump at $x = 0.1$.

The fluid domain is given by $\Omega_F = [0, 5] \times [0, 0.5]$ and the solid domain by $\Omega_s = [0, 5] \times [0.5, 0.6]$. Initially, both the fluid and the structure are at rest. The boundary conditions are as follows. In the structure domain, we impose null displacement at $x = 0$ and $x = 5$, while zero traction is applied on $y = 0.6$. For the fluid, we impose slip boundary conditions at $y = 0.0$, and an over-pressure of 10^4 during 5×10^{-3} time units. For the coupling between fluid and structure, we also impose slip boundary conditions, that is:

- Velocity continuity is imposed only in the direction normal to the fluid-structure interface.
- We impose traction continuity in the direction normal to the interface, but we do not consider tangent tractions.

The spatial discretization is carried out by means of a finite element mesh, its size being $h = 0.1$ (see Fig. 7.9), and the time step is set to $\delta t = 10^{-4}$. The backward Euler scheme is used for the integration of the transient Stokes equations in the fluid, and the explicit second order Newmark method is chosen for the time integration of the linear elasticity equations in the solid.

The main purpose of this numerical example is to compare the behavior of the numerical scheme with or without considering the contribution of the subscales on the element boundaries, and in particular regarding the added mass effect.

We will firstly consider no subscales on the element boundaries and a Dirichlet–Neumann coupling strategy: we apply Dirichlet boundary conditions to the fluid domain and Neumann boundary conditions to the solid one. The iterative scheme we use is (7.53), considering the possibility to iterate within each time step to converge to the solution of the monolithic problem or without iterations. In this last case, corresponding to the so called staggered schemes (or also

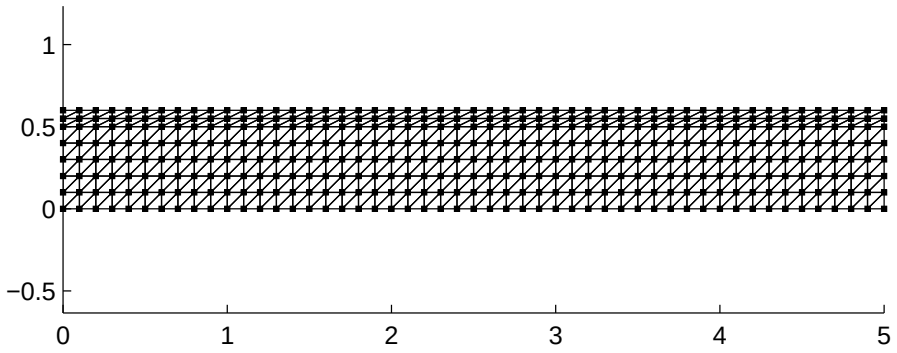


Figure 7.9: Finite element mesh

"loose coupling"), there is an error of the order of the time step size with respect to the solution of the monolithic problem. In (7.53) we will also consider the possibility of removing the terms coming from the subscales on the boundaries.

Initially, we consider a fluid density $\rho_F = 1.1$ and a viscosity $\mu = 0.035$. For the solid we take a density $\rho_S = 1.2$, a Young modulus $E = 3 \times 10^8$ and the Poisson ratio $\nu = 0.0$. If we consider an explicit coupling (we do not iterate until convergence at each time step) the numerical scheme explodes, and we obtain an unstable numerical solution.We now consider a coupling strategy which involves convergence at each time step, in particular we impose that the relative error between the solution obtained at iteration i and iteration $i + 1$ is less than 10^{-3}. Due to the added mass effect, this simple Dirichlet-Neumann scheme does not converge, and we need to use some additional tool in order to achieve the solution of the monolithic problem. Amongst the various existing methods, we can use a relaxation scheme. We have found effective to take a relaxation parameter $\alpha = 0.3$ in (7.45). Fig. 7.10 shows the vertical displacement at a solid point placed in the center of the solid domain. Fig. 7.11 shows the velocity and pressure fluid fields at $t = 75 \times 10^{-4}$. The mean number of iterations at each time step was 26 for this numerical scheme.

Let us now consider the use of the subscales on the element boundaries, for which we will take $\delta_0 = 0.5 \times 10^{-3}$. If we try to use an explicit coupling scheme, the method fails again due to the added-mass effect, and we obtain an unstable solution. However, if we iterate at each time step and we converge to the monolithic solution, we obtain the solution depicted in Fig. 7.12, which is exactly equal to the one obtained without using subscales on the element boundaries, since the additional terms due to the use of these subscales vanish when convergence is achieved. However, we did not need to use relaxation in this case, and the mean number of iterations was substantially reduced to 5, with the subsequent reduction of CPU time required to perform the computations.

Let us now consider a less demanding situation, in which the solid density is $\rho_S = 20$. In this case, the ρ_S/ρ_F ratio is around 20 too. This means that the added mass effect is not as severe as it was in the previous example, and that there might be no need to use an iterative scheme. We now test the explicit scheme with and without subscales on the element boundaries. In the first case we take $\delta_0 = 1 \times 10^{-3}$. Fig. 7.13 shows the results obtained with both schemes. We can see that the scheme becomes unstable after a few time steps if no subscales

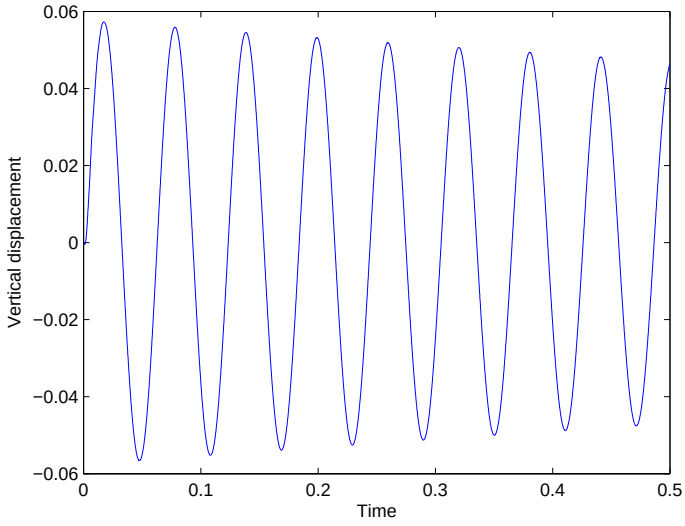


Figure 7.10: Results for the iterative scheme, no subscales, relaxation parameter $\alpha = 0.3$

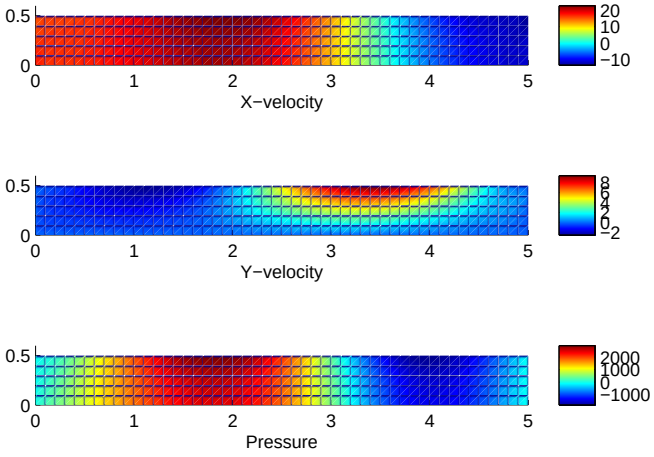


Figure 7.11: Velocity and pressure fluid fields at $t = 75 \times 10^{-4}$

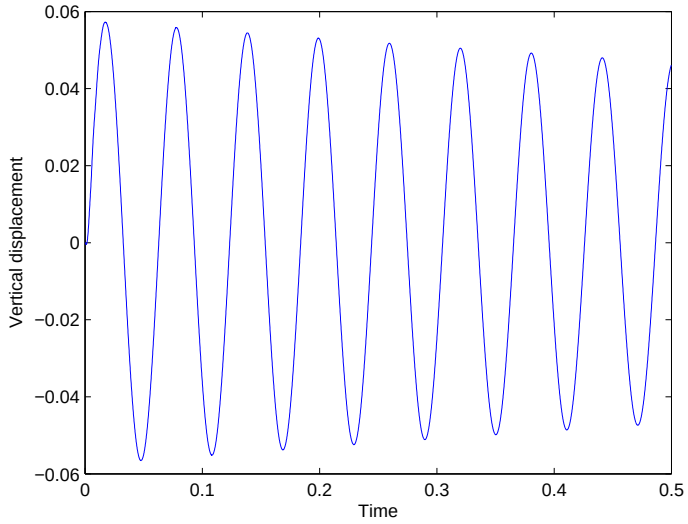


Figure 7.12: Results for the iterative scheme with subscales, no relaxation.

are used, but it remains stable, even for the explicit scheme, if the strategy we propose is used.

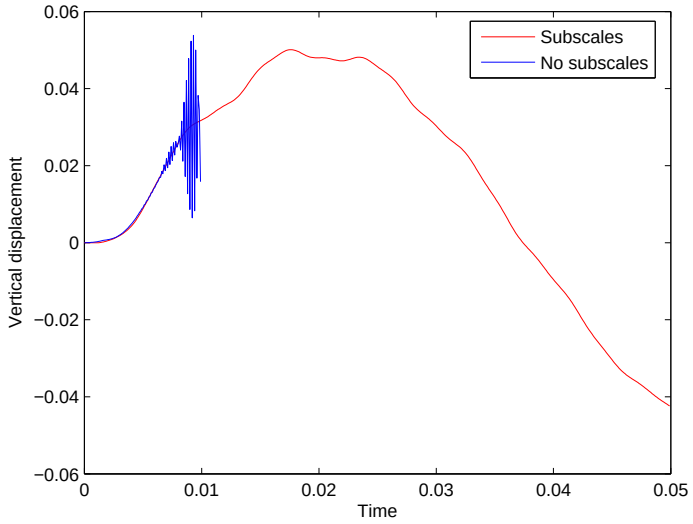
7.7 Conclusions

In this chapter we have motivated the introduction of element boundary terms in the finite element approximation the Stokes and the linear elasticity problems. The starting point has been the splitting of the unknowns of the problem into a conforming part and a discontinuous one, introducing a hybrid formulation only for the latter (Section 7.2). Although this approach could serve for different purposes, in Section 7.3 we propose a finite element approximation in which the discontinuous component of the solution, its traces and fluxes are approximated by expressions that involve only the conforming part of the solution. The resulting formulation is a stabilized finite element method for the Stokes problem which allows arbitrary interpolations of velocities and pressure. Particular emphasis has been put here on the treatment of Neumann-type boundary conditions.

The same ideas have been applied to the homogeneous interaction between two subdomains (Section 7.4). In this case, the benefit of the boundary terms is a stronger enforcement of the continuity of fluxes between subdomains. The matrix structure of the resulting system has been described and iterative schemes to be used in an iteration-by-subdomain environment have been proposed.

The fluid structure interaction problem has then been treated (Section 7.5). The extension of the previous ideas to this case has led to a modification of what can be considered a classical solid-fluid iterative coupling. The boundary terms introduced, which cancel when convergence is achieved, would hardly be motivated from a purely algebraic point of view.

All our predictions have been stated based on physical reasoning, without numerical analysis. Numerical experiments have confirmed the theoretical predictions. In particular, a better enforcement of the continuity of fluxes is found in homogeneous domain interaction problems



,

Figure 7.13: Comparison between results with and without using subscales in the case of a ratio ρ_S/ρ_F around 20 and an explicit scheme.

and, what is probably the most salient result of this work, convergence of solid-fluid iterative coupling algorithms is greatly improved by the terms we suggest to introduce.

Chapter 8

FELAP, a Finite Element Linear Algebra Package

In this chapter we present *FELAP*, a linear system of equations solver package for problems arising from finite element analysis. The main features of the package are its capability to work with symmetric and unsymmetric systems of equations, direct and iterative solvers and various renumbering techniques. Performance is enhanced by considering the finite element mesh graph instead of the matrix graph, which allows to perform highly efficient block computations.

8.1 Introduction

When performing numerical simulations with the finite element method, one invariably ends up with the need of solving a linear system of equations. Most finite element codes use linear system solvers developed by other groups and for other purposes. In most cases, this solvers are designed to cope with the most general kind of systems of equations, which means that they do not take advantage of the particularities of the systems of equations arising from the finite element analysis. This is why we aim to develop a solver package specially designed to solve finite element problems, which we will call *FELAP* (Finite Element Linear Algebra Package). This chapter departs significantly from the scope of the rest of the thesis, and deals mainly with the algorithmic and implementation aspects of the solver library. For completeness, the detailed description of several basic algorithms is also included.

There are many different strategies to deal with the solution of linear system of equations. The two main families of linear systems solvers are *direct* and *iterative* solvers.

In direct solvers the system matrix is factorized into an upper and a lower matrix which are easily invertible. The main feature of direct solvers is that the solution of the linear system of equations is always achieved. However, the memory and CPU time requirements increase very rapidly with the size of the linear system of equations, which makes them inappropriate for solving the very large systems of equations arising in finite element analysis. The earlier strategies to deal with sparse linear systems were based on the skyline storage, in which all the entries of the matrix comprised inside the row bandwidth are stored. Currently algorithms based on compact sparse storage schemes are the most commonly used. These algorithms are

basically composed of four steps [97]:

- Renumbering step, obtain an ordering which minimizes fill-in
- Symbolic factorization, compute fill-in pattern and memory requirements
- Factorization, compute the factors of the original matrix
- Obtain the system solution

The most time consuming step is the factorization of the matrix, but the two previous steps are essential in order to minimize the number of operations needed to compute the factors. Review papers on direct sparse solvers can be found in [46, 57, 67]. Some of the most popular direct sparse solvers can be found in [42, 124, 1].

Although state of the art direct solvers are very efficiently coded and are able to solve relatively large systems of equations, the inconvenient of direct solvers is still the large amount of time and memory required to obtain the solution if very large systems of equations are solved. This is why iterative methods are needed. The iterative algorithm is composed of two main ingredients:

- The driver, which successively reduces the error between the approximated solution and the real solution.
- The preconditioner, which changes the system of equations to be solved into a new equivalent system which is easier for the iterative algorithm to solve.

Drivers can be divided into non-Krylov methods (see [132, 140]) and Krylov methods. Krylov methods are the most powerful and the ones usually used to solve large systems of equations. Amongst them we can highlight the Conjugate Gradient method [88, 68], the GMRES method [134] and the BICGSTAB method [131].

There are two families of preconditioners which have received major attention in the past years: preconditioners deriving from the Algebraic Multigrid method (AMG) and incomplete factorization preconditioners (ILU). AMG was originally developed independently from Krylov methods [21], but it was later reinterpreted as a preconditioner method and used together with Krylov methods, see [19] and the cites therein. After the development of AMG, multigrid ideas were introduced in incomplete factorization preconditioners by means of reordering algorithms which lead to multilevel ILU (MILU) preconditioners [13, 96, 122, 130]. A comparison between AMG and MILU methods can be found in [110]. In our solver package we opt for a multilevel ILU strategy, which benefits from the good scalability of AMG methods and from the versatility of ILU strategies.

The main design characteristics of this solver package are:

- The solver will be designed in order to be able to cope with large systems of equations. To this end storage of matrices will be done in *CS* format (Compact Sparse), and both direct and iterative methods to solve linear systems of equations will be included in the package.

- The graph of the sparse matrix associated to the linear system of equations is associated to the graph of the finite element mesh. The solver will work by blocks: if we denote by *ndof* the number of degrees of freedom associated to each node of the mesh, then each item of the mesh graph will correspond to a block of dimension *ndof* $\times$ *ndof* in the matrix of the system of equations. This has three major advantages:

  1. The number of non-numerical operations due to the *CS* storage is drastically reduced in vectorial problems. For example, for a 3D stationary Stokes problems, the number of non-numerical operations is reduced by 16 (4×4).
  2. Operations are computed by blocks, this allows to improve the performance of each single floating point operation.
  3. There is a clear gain in the memory required to store the indices of the CS storage. Again, for a 3D Stokes problem, the amount of memory is reduced by 16.

- Matrices arising from finite element analysis, although not always symmetric, are always *structurally symmetric*. We take advantage of this fact and we store separately the lower triangular part of the matrix, the diagonal (or block diagonal), and the upper triangular part. In particular, we store the upper part of the matrix by rows and the lower part by columns. This has three major advantages:

  1. Since the matrix is *structurally symmetric* and we store the upper part by rows and the lower part by columns, the graph of both halves coincides. This allows to save half of the memory required for the matrix graph.
  2. This approach leads to *ILUC* preconditioners (Incomplete LU Crout) in iterative solvers, which are much more efficient than classical *ILUT* preconditioners (Incomplete LU Tolerance) associated to the *CSR* (Compact Sparse Row) storage.
  3. *ILUC* preconditioners (or *LUC* factorizations in the case of direct solvers) do the factorization of the matrix in such a way that the Schur complement of a block of the matrix is very easily computed with very little modification of the preconditioner (factorizer) subroutine.

- We aim for our iterative solvers to have a computational cost which increases as close as possible to linearly with the number of unknowns of the system of equations to be solved.

In the following sections we present the strategy followed to achieve these objectives.

8.2 The general strategy

One of the requirements FELAP has to fulfill is being as flexible as possible so that it can be used to manage the several systems of equations which may have to be solved in a finite element code. To achieve this FELAP is given a general entity structure which we believe responds to the needs of most finite element codes.

The general structure of FELAP starts from one or more finite element meshes. Each finite element mesh has an associated graph. Associated to the graph of the mesh we have a renumbering strategy. Several problems may be solved at the same time in a single finite element mesh, this is why there can be several matrices associated to a single mesh, with which the mesh shares the graph. Each matrix has its own (complete or incomplete) factorization. If we were to use always direct solvers all the factorizations corresponding to the matrices of a single mesh would share the graph, but as we may perform incomplete factorizations, there is a different graph attached to each factorization. This general structure can be seen in Fig. 8.1.

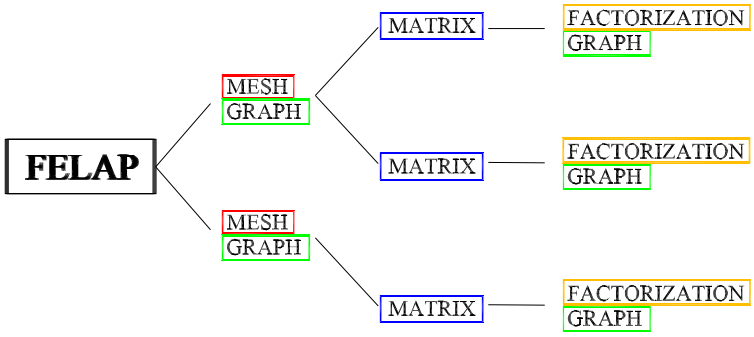


Figure 8.1: FELAP general structure

8.3 Building the mesh graph

FELAP is a solver package specially developed in order to cope with problems arising from finite element analysis problems. This means that linear systems of equations to be solved are supposed to come from a finite element spatial discretization, involving a mesh, elements and nodes. Possibly, multiple physics problems will be solved over the same mesh, and without a doubt problems with several degrees of freedom per mesh node will have to be solved. In all these cases, however, the graph of the matrix resulting of the system of equations to be solved will be somehow related to the graph of the finite element mesh. In particular, for scalar problems, the graph of the finite element mesh will coincide with the graph of the system of equations to be solved. This is why, instead of building the graph of the matrix for each problem to be solved on a finite element mesh, we store only the graph of the mesh. As we will see, positions in the matrix can be very easily accessed with the graph of the mesh, even for multiple degrees of freedom per node. This has the following advantages:

- The number of non-numerical operations is reduced in vectorial problems. For example, for a 3D stationary Stokes problems, the number of non-numerical operations is reduced by 16 (4×4).

- Operations are computed by blocks in problems with multiple degrees of freedom per node, this allows to improve the performance of each single floating point operation.

- There is a clear gain in the memory required to store the graphs of the matrices. Again, for a 3D Stokes problem, the amount of memory is reduced by 16.

In order to build the mesh graph, $FELAP$ requires a list of elements containing the nodes of each element. We require this list in a compact format, allowing for a different number of nodes in each element. The compact format is composed of two vector indices, iEL and jEL, $iEL(i)$ containing the starting position in jEL of the list of nodes of element i. $jEL(iEL(i) : iEL(i+1)-1)$ contains the node list for element i.

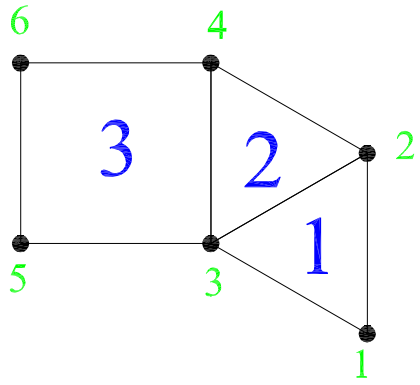


Figure 8.2: Finite element mesh

Consider for example the finite element mesh in Fig. 8.2. Elements 1 and 2 have 3 nodes, whereas element 3 has 4 nodes. The storage of this mesh would produce the following:

Pos.	1	2	3	4	5	6	7	8	9	10
iEL	1	4	7	11						
jEL	1	2	3	3	2	4	5	3	4	6

What we want to compute is the graph of the mesh, that is, for each node, the list of nodes to which it is connected. Two nodes are connected when there is some element to which they both belong. FELAP stores the graph of the mesh in a rather peculiar way, but that has some very advantageous features. We store the mesh graph by means of 3 vector indices, iA, iS, jA, iA and iS of size the number of nodes plus 1, jA of size the total number of connectivities of the mesh. iA and jA work as iEL and jEL, $iA(i)$ pointing to the starting position of the list of connectivities of node i in jA, stored in $jA(iA(i) : iA(i+1)-1)$. However, we put an extra requirement in the way this list is stored: for the list corresponding to node i, we require all the nodes with node number lesser than i to be stored before the nodes with node number greater than i. iS is built in the following manner:

$$iS(1) = 1, \quad iS(i+1) - iS(i) = \text{number of nodes } j \text{ connected to } i \text{ with } j > i$$

iA and jA allow to recover all the nodes connected to node i, but adding the vector iS to the storage scheme allows, if desired, to recover only the nodes j connected to i with $j > i$. Let us see what would this graph storage scheme result in for the mesh in Fig. 8.2:

Pos.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
iA	1	3	6	11	15	18	21													
iS	1	3	5	8	10	11	11													
jA	2	3	1	3	4	1	2	4	5	6	2	3	5	6	3	4	6	3	4	5

We have depicted in green the cells in jA corresponding to nodes j connected to node i such that $j > i$.

Now iA and jA allow to access all the connectivities of a node. For scalar problems this corresponds to the CSR format for sparse matrices to be seen in the next section (except for the fact that, for node i, we do not store i itself in jA). On the other hand, if one wants to access only the nodes j connected to i such that $j > i$, this can be achieved by using iS:

$$* = iA(i+1) - iS(i+1) + iS(i)$$

allows to access the first node j connected to i with $j > i$, which is stored in $jA(*)$. As a consequence, $jA(* : iA(i+1) - 1)$ contains the nodes j connected to i with $j > i$. For scalar problems this corresponds to the CS-Crout format for sparse matrices to be seen in the next section.

The fact is that we will store our matrices in CS-Crout format, but that we will need the graph of the mesh in CSR format for renumbering algorithms. This is the reason why we store the extra array iS which allows us to have both formats in the same structure.

The mesh graph can be easily recovered from the element list by means of an efficient $\mathcal{O}(npoin \times nelem \times nnode)$ algorithm, where $npoin$ is the total number of points of the mesh, $nelem$ is the total number of elements and $nnode$ is the mean number of nodes per element.

8.4 Storing the matrix

Let us consider a non singular square matrix A

$$A := (a_{i,j})_{n \times n}, \quad 1 \leq i \leq n, \quad 1 \leq j \leq n,$$

arising from a finite element analysis. Let b

$$b := (b_i)_{n \times 1}, \quad 1 \leq i \leq n,$$

be the right hand side of the system of equations to be solved and

$$x := (x_i)_{n \times 1}, \quad 1 \leq i \leq n,$$

the vector of unknowns satisfying

$$Ax = b. \tag{8.1}$$

Matrices arising from linear systems of equations to be solved in finite element analysis can be very large, but, at the same time, they are generally very sparse. In order to reduce the required memory, the storage of matrix A should be done in such a way that the zero entries of A are not stored. The most common and popular way of storing large sparse matrices are *Compact Sparse* (CS) formats.

8.4.1 The CSR format

In CS formats only the non-zero entries of a matrix and the necessary indices to ubicate them in the matrix. The most popular CS format is *Compact Sparse Row*. Let us denote by nzA the number of non-zero entries of the matrix A. CSR consists in storing the non-zero entries of matrix A in a vector nA of size nzA, and constructing two vector indices iA and jA of size $n+1$ and nzA respectively:

$$nA(k) = a_{i,j}, \quad j = jA(k), \quad iA(i) \leq k \leq iA(i+1) - 1, \quad iA(1) = 1. \tag{8.2}$$

This implies that the non-zero values of a given row are stored sequentially, and that rows are stored in an increasing order. Vector indices iA and jA form what we call *the graph of the matrix* for a the given compact sparse format.

Let us consider the following matrix:

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & 0 & 0 \\ a_{21} & a_{22} & 0 & a_{24} & 0 \\ a_{31} & 0 & a_{33} & a_{34} & a_{35} \\ 0 & a_{42} & a_{43} & a_{44} & 0 \\ 0 & 0 & a_{53} & 0 & a_{55} \end{pmatrix} \tag{8.3}$$

If we store this matrix in CSR we obtain :

Pos.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
iA	1	4	7	11	14	16									
jA	1	2	3	1	2	4	1	3	4	5	2	3	4	3	5
nA	a_{11}	a_{12}	a_{13}	a_{21}	a_{22}	a_{24}	a_{31}	a_{33}	a_{34}	a_{35}	a_{42}	a_{43}	a_{44}	a_{53}	a_{55}

In the same way, the *Compact Sparse Column* format is a compact sparse format which stores the matrix by columns instead of by rows.

Although this two formats are the most common ones, they have a major drawback: they are not suitable for symmetric matrices, in which:

$$a_{ij} = a_{ji}, \quad 1 \leq i \leq n, \quad 1 \leq j \leq n, \tag{8.4}$$

and in order to reduce memory requirement we are interested in storing only the diagonal and the upper or lower triangular part, but not both. Taking this into account, we opt by the *Compact Sparse Crout* format (CS-Crout) which is suitable for both symmetric and unsymmetric matrices.

8.4.2 The CS-Crout format

In the CS-Crout format we divide matrix A into a lower triangular matrix L, its diagonal D and an upper triangular matrix U, such that:

$$\begin{aligned} A &= L + D + U \\ L_{ij} &= 0 \quad \forall j \geq i \\ U_{ij} &= 0 \quad \forall j \leq i \\ D_{ij} &= 0 \quad \forall j \neq i \end{aligned}$$

We denote by nzL the number of non-zero entries in D and by nzU the number of non-zero entries in U. As a consequence, $nzA = n + nzL + nzU$, since n is the number of entries D, which are stored independently of their value.

CS-Crout consists in:

- Storing the diagonal D in a vector dA of size n
- Storing the non-zero entries of L in a vector lA of size nzL, and constructing to vector indices iL and jL of size $n+1$ and nzL respectively
- Storing the non-zero entries of U in a vector uA of size nzU, and constructing to vector indices iU and jU of size $n+1$ and nzU respectively.

The non-zero value vectors and the indices are constructed in the following way:

$$\begin{aligned} &dA(i) = d_{i,i}, \quad 1 \le i \le n, \\ &lA(k) = L_{i,j}, \quad i = jA(k), \quad iL(j) \le k \le iL(j+1) - 1, \quad iL(1) = 1, \\ &uA(k) = U_{i,j}, \quad j = jA(k), \quad iL(i) \le k \le iL(i+1) - 1, \quad iU(1) = 1. \end{aligned}$$

Note that in this case the upper triangular part of the matrix is stored by rows, while the lower part is stored by columns. This may seem an unnecessary complication, but we will see that leads to a very advantageous feature.

Let us consider again the matrix in (8.3). This matrix, when stored in CS-Crout format, yields:

Pos.	1	2	3	4	5	6
dA	a_{11}	a_{22}	a_{33}	a_{44}	a_{55}	
iU	1	3	4	6	6	6
jU	2	3	4	4	5	
uA	a_{12}	a_{13}	a_{24}	a_{34}	a_{35}	
iL	1	3	4	6	6	6
jL	2	3	4	4	5	
uL	a_{21}	a_{31}	a_{42}	a_{43}	a_{53}	

There is no gain in using CS-Crout instead of CSR by now. The amount of memory required for the indices is the same (the jA positions corresponding to the diagonal in CSR -size n- are replaced by the extra iL vector index in CS-Crout -size $n+1$-). Let us now consider a *structurally symmetric* matrix, which means:

$$a_{ij} \neq 0 \Leftrightarrow a_{ji} \neq 0 \quad 1 \le i \le n, \quad 1 \le j \le n$$

but $a_{ij} \neq a_{ji}$ in general. Matrices arising from finite element analysis are structurally symmetric. If this is the case, as in (8.3), *indices iL and iU coincide in the CS-Crout format*. This means that there is no need to store both of them. What we do is to store a single index which we call iA. The same happens with jL and jU, which we condense in a single index jA. As a consequence, the amount of memory required for storing a structurally symmetric matrix in CS-Crout format is significantly smaller than the memory required to store the same matrix in CSR. Still, this is not the major advantage of using CS-Crout instead of CSR.

Let us now consider a symmetric matrix. Symmetric matrices are obviously structurally symmetric. Moreover, in a symmetric matrix vectors uA and lA in the CS-Crout format are equal. This means that there is no need to store both vectors, and that *memory requirements for the storage of the matrix can be reduced to the half*. This is the deciding factor which led us to opt for CS-Crout instead of CSR. CS-Crout is a format in which symmetric and structurally symmetric matrices are treated in a very natural way, and that takes advantage of this fact in order to reduce storage memory requirements. Of course, this alone is not enough, since another requirement for compact sparse formats is that operations involving matrices can be performed in an efficient way. As we will see in the following sections, this requirement is satisfactorily met by the CS-Crout format.

8.5 Matrix times vector product

There is a single operation to be performed with the original matrix A when solving a linear system of equations. This is matrix times vector product:

$$\begin{aligned} c &= Ab, \\ c_i &= \sum_{j=1}^{n} a_{ij} b_j \end{aligned} \tag{8.5}$$

For full matrices the algorithm to compute the product is straightforward, but if matrices are stored in a sparse format, the operations involved in (8.5) have to be performed in the correct order so that *the need for searching through the vectors in which the matrix is stored is avoided*. Let us illustrate this point. Consider the matrix times vector product in (8.5) for a full matrix. The algorithm associated to this operation can be seen in Algorithm 1.

Algorithm 1 Matrix times vector, full matrix, (i, j)

```
1: for i = 1 : n do
2:     c(i) = 0
3:     for j = 1 : n do
4:         c(i) = c(i) + A(i, j) * b(j)
5:     end for
6: end for
```

If one wants to change the order of the loops there is no major problem (except for the jumps in memory positions, which can be an issue in high performance computations), as seen in Algorithm 2.

8.5.1 Matrix times vector in CSR

Let us now perform the same operation with the CSR storage. Algorithm 1 now turns to Algorithm 3. Performed this way the algorithm is efficient and the complexity of the algorithm (accounting also for non-numerical operations deriving from the need of reading the sparse format indices) is of $\mathcal{O}(n \times \overline{nzr})$, where we have denoted by $\overline{nzr}$ the mean number of non-zero entries in a row of matrix A.

Algorithm 2 Matrix times vector, full matrix, (j, i)

```
1: c(:) = 0
2: for j = 1 : n do
3:    for i = 1 : n do
4:       c(i) = c(i) + A(i, j) * b(j)
5:    end for
6: end for
```

Algorithm 3 Matrix times vector, CSR, (i, j)

```
1: for i = 1 : n do
2:    c(i) = 0
3:    for k = iA(i) : iA(i + 1) − 1 do
4:       c(i) = c(i) + nA(k) * b(jA(k))
5:    end for
6: end for
```

When trying to reproduce Algorithm 2 with the CSR format we find that it is very difficult to travel through a column of matrix A due to the way in which non-zero coefficients in A are stored. A first approach to Algorithm 2 would be Algorithm 4. But now the complexity of the algorithm is of $\mathcal{O}(n \times n \times \overline{nzr})$. This is obviously unaffordable. With this simple example we have demonstrated that even the most simple operations have to be done in the correct order if the matrix is stored in a compact sparse format. Not doing so implies increasing the number of operations by a factor of n. As a consequence, opting for an specific storage format implies that any algorithm involving the sparse matrix will have to be specifically coded for the chosen format.

Algorithm 4 Matrix times vector, CSR, (j, i)

```
1: c(:) = 0
2: for j = 1 : n do
3:    for i = 1 : n do
4:       k = iA(i)
5:       while jA(k) ≠ j and k < iA(i + 1) do
6:          k = k + 1
7:       end while
8:       if jA(k) = j then
9:          c(i) = c(i) + nA(k) * b(jA(k))
10:      end if
11:   end for
12: end for
```

Algorithm 4 demonstrates that travelling through the matrix in an *unnatural* way (referred to the sparse matrix storage format) must be avoided by all means. However, there will be times, and for certain algorithms, when it will be peremptory to do so. If this is the case, there are tools which allow us to keep the number of operations being of the same order than if we

moved through the matrix in the *natural* way. These are linked lists.

8.5.2 Linked lists

Linked lists consist of a sequence of nodes, each containing arbitrary data fields and a reference pointing to the next node. In order to keep the number of operations in Algorithm 4 of $\mathcal{O}(n \times \overline{nzr})$ we would require, for each column j, a linked list with which we could travel through the non-zero values of the column. Let us see how to build a linked list for the matrix in (8.3).

A linked list will consist of the indices $first$, $last$ and the vector index $list$ of dimension $(2, n)$. Let us consider the second column of A:

$$A_{:,2} = \begin{pmatrix} a_{12} \\ a_{22} \\ 0 \\ a_{42} \\ 0 \end{pmatrix}$$

For the second column, the linked list pretends to travel through the non-zero entries of matrix A which belong to the column. As a consequence we assign to $first$ the row of the first non-zero entry of column 2, which is a_{12}:

$$first = 1$$

In the same way, we assign to $last$ the row of the last non-zero entry of column2, which is a_{42}:

$$last = 4$$

For column 2, $list$ would contain:

$$list = \begin{pmatrix} 2 & 5 & 0 & 11 & 0 \\ 2 & 4 & 0 & -1 & 0 \end{pmatrix} \tag{8.6}$$

$list(1, i)$ contains the positions in vector nA in which a_{i2} is contained. a_{12} is stored in $nA(2)$, a_{22} is stored in $nA(5)$ and a_{42} is stored in $nA(11)$. $list(2, i)$ contains which is the next row which has a non-zero value in column 2. In this case, row 1 points to row 2, which points to row 4. As there is no row greater than 4 whose component in column 2 is different from 0, row 4 points to -1. If we have a linked list for each of the columns of matrix A we will be able to travel through the columns of A, stored in CSR. However, keeping a linked list for each column will require a lot of memory. Algorithm 5 shows how to dynamically build a linked list which allows to columnwise traverse A even if this is stored by rows, at the sole cost of two vectors, $first$ and $last$, of size n, and the vector $list$ of dimensions $(2, n)$. The algorithm has one requirement: column indices for a row in jA must be stored in increasing order.

Let us apply Algorithm 5 to traverse matrix A columnwise when it is stored in a CSR format. Lines 1 to 3 initialize the linked list to zero. Lines 4 to 15 put the first component of each row in the linked list. As we will see, only one non-zero entry of each will be in the linked list at the same time. Once this non-zero entry has been used, it will be replaced by the next non-zero of the row. After line 15 the linked list vectors for matrix A stored in CSR are:

Algorithm 5 CSR, (j,i), Linked list

```
first(:) = -1
last(:) = -1
list(:,:) = 0
for i = 1 : n do
  if iA(i) ≠ iA(i+1) then
    k = iA(i)
    if last(jA(k)) = -1 then
      first(jA(k)) = i
    else
      list(2, last(jA(k))) = i
    end if
    list(1,i) = k
    last(jA(k)) = i
  end if
end for
for j = 1 : n do
  i = first(j)
  while i ≠ -1 do
    k = list(1,i)
    print nA(k)
    if list(1,i) < iA(i+1) - 1 then
      if last(jA(list(1,i)+1)) = -1 then
        first(jA(list(1,i)+1)) = i
      else
        aux = last(jA(list(1,i)+1))
        list(2,aux) = i
      end if
      last(jA(list(1,i)+1)) = i
    end if
    list(1,i) = list(1,i) + 1
    aux = i
    i = list(2,i)
    list(2,aux) = -1
  end while
end for
```

Pos.	1	2	3	4	5
$first$	1	4	5	-1	-1
$last$	3	4	5	-1	-1
$list(1,:)$	1	4	7	11	14
$list(2,:)$	2	3	-1	-1	-1

As we can see, after this first loop the linked list is ready to traverse column 1 but it is not ready to traverse any other column. The first iteration in the loop starting in line 16 prints the first column of A, but it also modifies the linked list so that at the end of this first iteration the linked list is ready to traverse column 2:

Pos.	1	2	3	4	5
$first$	1	4	5	-1	-1
$last$	3	2	3	-1	-1
$list(1,:)$	2	5	8	11	14
$list(2,:)$	2	-1	-1	1	3

This procedure is repeated for each column of A. This pseudocode applied to matrix A prints:

a_{11}
a_{21}
a_{31}
a_{42}
a_{12}
a_{22}
a_{53}
a_{33}
a_{13}
a_{43}
a_{24}
a_{34}
a_{44}
a_{55}
a_{35}

where we have grouped the coefficients of each column. Note that, for a given column, *coefficients are not recovered in a row-increasing order*. Algorithm 5 can be modified into a matrix times vector product simply by replacing line 20 by:

$$c(i) = c(i) + nA(k) * b(jA(k))$$

The complexity of the algorithm is now $\mathcal{O}(n \times \overline{nzr})$. For a matrix times vector product Algorithm 5 is slower than Algorithm 3. For more complex computations, in which the complexity of the operation in line 20 in Algorithm 5 is of $\mathcal{O}(n)$ or $\mathcal{O}(\overline{nzr})$, Algorithm 3 and Algorithm 5 will perform almost equally. This is in fact what happens in the CS-Crout matrix factorization algorithm, as we will see.

8.5.3 Matrix times vector in CS-Crout

The efficient algorithm for computing a matrix times vector product if the matrix is stored in CS-Crout format is Algorithm 6. Note that the complexity is of $\mathcal{O}(n \times \overline{nzr})$ and that the number of operations is exactly the same than in Algorithm 3.

Algorithm 6 Matrix times vector, CS-Crout

```
1: c(:) = 0
2: for i = 1 : n do
3:    for k = iA(i) : iA(i + 1) − 1 do
4:       c(i) = c(i) + uA(k) ∗ b(jA(k))
5:       c(jA(k)) = c(jA(k)) + lA(k) ∗ b(i)
6:    end for
7: end for
```

8.6 Direct solvers

When dealing with the system of equations (8.1), the exact solution can be found by *factorizing* matrix A and performing a backward and forward substitution. Factorizing a matrix means finding two matrices L lower triangular and U upper triangular, the diagonal of L full of ones, such that:

$$\begin{aligned} u_{ij} &= 0 \quad \forall i > j, \quad 1 \leq i \leq n, \quad 1 \leq j \leq n, \\ l_{ij} &= 0 \quad \forall i < j, \quad 1 \leq i \leq n, \quad 1 \leq j \leq n, \\ l_{ii} &= 1, \quad 1 \leq i \leq n, \\ LU &= A. \end{aligned} \tag{8.7}$$

Once these matrices have been factorized the solution of the original system can be found by solving:

$$\begin{aligned} Ly &= b \\ Ux &= y \end{aligned} \tag{8.8}$$

These two systems are very easily solved since they are triangular.

8.6.1 Matrix factorization

There are various algorithms which compute the factorization of a matrix (which is unique), depending on the order in which the factorized matrix coefficients are computed. Each of these algorithms is associated to a particular compact sparse storage scheme. Here we present the Crout algorithm, which is the one associated to the CS-Crout format we use.

For full matrices, the Crout algorithm can be seen in Algorithm 7, where the factorized matrices have been stored in the same memory space occupied by A.

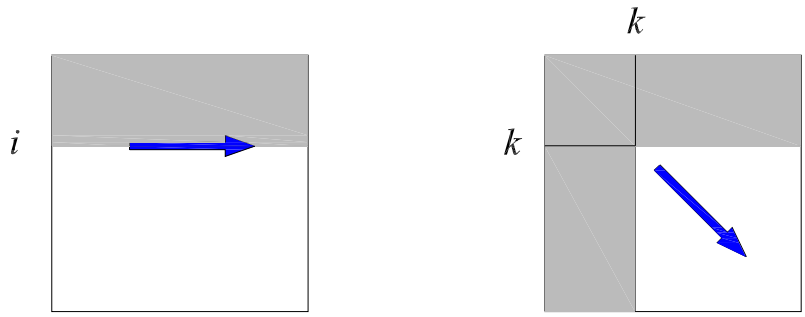


Figure 8.3: Factorization order in CSR (left) and CS-Crout (right). Areas in gray are factorized, in white, to be factorized

Algorithm 7 Crout algorithm

for $k = 1 : n$ **do**
  for $i = 1 : k - 1$ **do**
    for $j = k : n$ **do**
      $A(k, j) = A(k, j) - A(i, k) * A(i, j)$
      $A(j, k) = A(j, k) - A(k, i) * A(j, i)$
    end for
  end for
  $A(k, k) = 1/A(k, k)$
  for $j = k + 1 : n$ **do**
    $A(k, j) = A(k, j) * A(k, k)$
  end for
end for

Obviously this algorithm crashes if $A(k,k) = 0$ in line 8. When this happens there are a series of pivoting algorithms which allow to change the matrix ordering in such a way that (for non-singular matrices) no zeros are found in the diagonal during the factorization process. We have to say, however, that in the finite element analysis linear systems of equations we have solved this problem has not appeared (even for problems with Lagrange multipliers in which the original matrix has zeros on the diagonal).

When dealing with the factorization in Algorithm 7, which is the most suitable in the case of a matrix stored in CS-Crout format, we face with the problem that the loop in i accesses the factorized matrices L and U in an *unnatural* order, as Fig. 8.4 shows. This corresponds to coefficients $A(i,k)$ and $A(k,i)$ in lines 4 and 5 in Algorithm 7, which for $1 \leq i < k$ correspond to Fig. 8.4 right. This is the reason why we have to resort to the linked list strategy described in Algorithm 5.

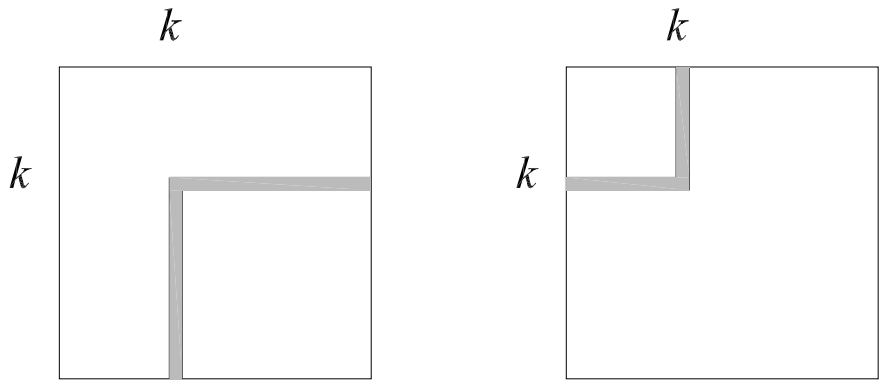


Figure 8.4: For a given row and column k, positions *naturally* (left) and *unnaturally* (right) consecutively accessed with the CS-Crout format

When dealing with the factorization of a matrix it is very convenient, in order to keep the complexity of the algorithm as low as possible, to work with the uncompressed row/column k. To this end, at the beginning of each k iteration we uncompress the row and column k into vectors rwa and $rwa2$ of maximum size n and two vector indices iwa of maximum size n and $seen$ of size n. The exact procedure is shown in Algorithm 8.

A position corresponding to column/row j in row/column k will have to be accessed in general for two purposes:

- Firstly to add the contribution of previous rows/columns i to row/column k. To this end the most efficient way of temporarily storing row/column k would be in a completely uncompressed vector. Any contribution to column/row j in row/column k would be done in $rwa(j)$ with no further complication.

- Secondly to store the non-zero coefficients corresponding to column/row j of the factorized row/column k in the CS-Crout format. In this case it would be convenient to work only with the non-zero coefficients of the factorized row/column k in a compressed way.

Algorithm 8 Uncompressing row/column k for a fast factorization procedure

```
1: iwa(:) = 0
2: seen(:) = 0
3: rwa(:) = 0
4: rwa2(:) = 0
5: rwa(1) = dA(k) {Start Uncompressing}
6: iwa(1) = k
7: seen(k) = 1
8: nz = 1
9: for j = iA(k) : iA(k + 1) − 1 do
10:    nz = nz + 1
11:    rwa(nz) = uA(j)
12:    rwa2(nz) = lA(j)
13:    iwa(nz) = jA(j)
14:    seen(jA(j)) = nz
15: end for
```

Algorithm 8 shows a working storage scheme which fulfills the requirements of both operations. In the first case, $seen(j)$ contains an index which leads to the corresponding position in the compressed arrays iwa, rwa and $rwa2$ without the need of searching through the compressed arrays. For the second case non-zero entries are grouped in the compressed arrays iwa, rwa and $rwa2$ and there is no need to traverse a full size n array and check for non-zero entries.

Now we have the necessary tools to write Algorithm 7 for sparse matrices. The factored matrix will be stored in CS-Crout matrices L and U. We recall that one of the requirements of the linked list strategy in Algorithm 5 was that the matrices were stored in a row/column-increasing order. This is why working arrays iwa, rwa and $rwa2$ are reordered before being stored into L and U in CS-Crout format. There is no need for matrix A to be stored in a row/column-increasing order. Note that since A is *structurally* symmetric the factored matrices L and U have the same sparsity structure: iL and jL coincide with iU and jU, and we have condensed them to iC and jC. Moreover the diagonal of L is full of ones and there is no need to store it. Likewise, a single linked list will be enough to traverse L and U. Algorithm 9 is the sparse version of Algorithm 7 using the strategies described in Algorithm 5 and Algorithm 8. The resulting algorithm is of complexity $\mathcal{O}(n \times \overline{nzr} \times \overline{nzr})$, where now $\overline{nzr}$ denotes the mean number of non-zero entries in the *factorized* matrix.

In the case of positive definite symmetric matrices, the Crout factorization is very easily modified to the so called *Cholesky* factorization by replacing Algorithm 7 by Algorithm 10. In this case, matrix A is factorized into:

$$\begin{aligned} LL^T &= A, \\ Ly &= b, \\ L^T x &= y. \end{aligned} \tag{8.9}$$

Note that only the lower triangular part of matrix A is required and factorized. This can be

Algorithm 9 Sparse Crout factorization using linked lists and fast working arrays

1: $ic(1) = 1$ {Initialize}
2: $nxt = 1$
3: $iwa(:) = 0$
4: $rwa(:) = 0$
5: $rwa2(:) = 0$
6: $seen(:) = 0$
7: $last(:) = -1$
8: $first(:) = -1$
9: $list(:) = -1$
10: **for** $k = 1 : n$ **do**
11: $rwa(1) = dA(k)$ {Uncompress}
12: $iwa(1) = k$
13: $seen(k) = 1$
14: $nz = 1$
15: **for** $j = iA(k) : iA(k+1) - 1$ **do**
16: $nz = nz + 1$
17: $rwa(nz) = uA(j)$
18: $rwa2(nz) = lA(j)$
19: $iwa(nz) = jA(j)$
20: $seen(jA(j)) = nz$
21: **end for**
22: $i = first(k)$ {Factorize row/column}
23: **while** $i \neq -1$ **do**
24: $iz = list(1, i)$
25: $t = lC(iz)$ {Row pivot}
26: $t2 = uC(iz)$ {Col pivot}
27: $rwa(1) = rwa(1) - t1 * t2$ {Contribution to the diagonal}
28: **for** $j = iz : iC(i+1) - 1$ **do**
29: $s = t * uC(j)$
30: $s2 = t2 * lC(j)$
31: **if** $seen(jC(j)) \neq 0)$ **then**
32: $rwa(seen(jC(j))) = rwa(seen(jC(j))) - s$
33: $rwa2(seen(jC(j))) = rwa2(seen(jC(j))) - s2$
34: **else**
35: $nz = nz + 1$
36: $seen(jC(j)) = nz$
37: $iwa(nz) = jC(j)$
38: $rwa(nz) = -s$
39: $rwa2(nz) = -s2$
40: **end if**
41: **end for**

```
42:       if list(1,i) < iC(i+1) − 1 then {Update linked list}
43:          if last(jC(list(1,i)+1)) = −1 then
44:             first(jC(list(1,i)+1)) = i
45:          else
46:             list(2,last(jC(list(1,i)+1))) = i
47:          end if
48:          last(jC(list(1,i)+1)) = i
49:       end if
50:       list(1,i) = list(1,i) + 1
51:       aux = i
52:       i = list(2,i) {Next item in the linked list for k}
53:       list(2,aux) = −1
54:    end while
55:    rwa(1) = 1/rwa(1) {A(k,k) = 1/A(k,k)}
56:    dC(k) = rwa(1) {Storing the values in dC, lC and uC}
57:    SORT iwa, rwa, and rwa2 so that they are ordered in increasing row/column order
58:    rwa2(2 : nz) = rwa2(2 : nz) ∗ dC(k)
59:    for j = 2 : nz do
60:       jC(nxt) = iwa(j)
61:       uC(nxt) = rwa(j)
62:       lC(nxt) = rwa2(j)
63:       nxt = nxt + 1
64:    end for
65:    iC(k + 1) = 1
66:    if nz > 1 then {Update linked list}
67:       if last(iwa(2)) = −1 then {Only the first component of the row/column is added to
       the list}
68:          first(iwa(2) = k
69:       else
70:          list(2,last(iwa(2))) = k
71:       end if
72:       last(iwa(2)) = k
73:    end if
74:    for j = 1 : nz do
75:       seen(iwa(j)) = 0
76:    end for
77: end for
```

also done to Algorithm 9, reducing both the number of operations and the memory requirements to the half.

Algorithm 10 Cholesky algorithm

```
1: for k = 1 : n do
2:    for i = 1 : k − 1 do
3:       for j = k : n do
4:          A(j, k) = A(j, k) − A(k, i) ∗ A(j, i)
5:       end for
6:    end for
7:    A(k, k) = 1/sqrt(A(k, k))
8:    for j = k + 1 : n do
9:       A(j, k) = A(j, k) ∗ A(k, k)
10:   end for
11: end for
```

In the same manner, it is possible to modify Algorithm 7 so that it suits symmetric but non-positive definite matrices. In this case A is factorized into:

$$\begin{aligned} LDL^T &= A, \\ Ly &= b, \\ Dz &= y, \\ L^T x &= z. \end{aligned} \tag{8.10}$$

where D is a diagonal matrix. This algorithm avoids the $\sqrt{A(k,k)}$ appearing in line 7 in Algorithm 10 which crashes in the case of non-positive definite matrices. Again, it is very easy, due to the chosen CS format, to modify Algorithm 9 into its equivalent LDL^T version.

8.6.2 Forward and Backward substitutions

Once matrix A has been factorized into L and U, the only thing which remains in order to solve the system of equations (8.1) is to solve the systems in (8.8). These systems are easily solved, since they are triangular. Algorithm 11 and Algorithm 12 show how to do so with the factored matrices stored in the CS-Crout storage format. This algorithm can be easily extended to the case of positive definite symmetric matrices or symmetric non-definite matrices.

Algorithm 11 Forward substitution

```
1: for i = 1 : n do
2:    for k = iC(i) : iC(i + 1) − 1 do
3:       b(jC(k)) = b(jC(k)) − b(i) ∗ lC(k)
4:    end for
5: end for
```

Algorithm 12 Backward substitution

```
1: for i = n : −1 : 1 do
2:     for k = iC(i + 1) − 1 : iC(i) : −1 do
3:         b(i) = b(i) − lC(k) ∗ b(jC(k))
4:     end for
5:     b(i) = b(i) ∗ dC(i)
6: end for
```

8.6.3 Symbolic factorization for structurally symmetric matrices

Algorithm 9 allows us to perform the factorization of matrix A stored in CS-Crout format. However, the issue of memory allocation is still unresolved. When entering Algorithm 9 the final memory requirement for jC, lC and uC should be known, so that they can be correctly allocated. To this end, we perform a *symbolic factorization*, its requirements being:

- At the end of the process, it must return nzC, the size of arrays jC, lC and uC.
- The complexity of the algorithm must be lower than the complexity of the real factorization.
- The memory requirements of the algorithm must be low.

Algorithm 9 can be easily transformed into a symbolic factorization by:

1. Not performing the floating point operations. We are interested only in the sparsity pattern of the factorized matrix, not in the values.
2. Skipping the update of the linked list between lines 42 and 52. The fill-in caused by row/column i in the following will be contained in the fill-in caused by row/column k in the following, so there is no need to take it into account. *This reduces the complexity of the algorithm to* $\mathcal{O}(n \times \overline{nzr})$.
3. Once row/column i has been once in the loop in 23 it will no longer be needed. The information stored in $jC(iC(i) : iC(i + 1) - 1)$ is no longer needed and this memory space can be deallocated. Obviously the storage scheme for jC should not be continuous, and the use of structure data types to store jC is required so that we are capable of dynamically allocating and deallocating the part of jC associated to row/column i.

The complexity of this symbolic factorization is of $\mathcal{O}(n \times \overline{nzr})$.

8.7 Iterative solvers

Direct solvers are the most convenient option when the systems to be solved are of relatively small size. However, for very large systems of equations, the memory requirements of direct solvers are too large for this methods to be used. This is the reason why one relies in iterative solvers. Most usual iterative methods for solving systems of equations are *Conjugate*

Gradient (CG), *Generalized Minimum Residual* (GMRES) and *Biconjugate Gradient Stabilized* (BICGSTAB) which will not be described here. See [121] for a detailed explanation of these methods. What we are interested in is that these methods convergence depends on the condition number of matrix A, which for normal matrices can be defined as:

$$\kappa(A) = \frac{|\lambda_{\max}(A)|}{|\lambda_{\min}(A)|}, \qquad 1 \leq \kappa(A) < \infty. \tag{8.11}$$

where $\lambda_{\max}(A)$ and $\lambda_{\min}(A)$ are the maximum and the minimum eigenvalues of matrix A respectively. The closer $\kappa(A)$ is to 1 the faster and more robust the iterative methods are. However, for many problems, the condition number of the system of equations can be extremely large. In this case one should use preconditioners.

8.7.1 Preconditioned systems of equations

Preconditioning a system of equations consists in solving an equivalent system (in the sense that it has the same solution, or that the solution of both systems can be easily related), but with a condition number as close as possible to 1. This allows for the iterative methods to solve the system much faster.

In general, the preconditioned system of equations is obtained by pre- or post-multiplying the original system by the inverse of a matrix M which we call the *preconditioner*. The system of equations in (8.1) now turns into:

$$AM^{-1}Mx = b,$$

if the preconditioner is applied by the right, which is what we do: preconditioning by the right has the advantage that the residual of the preconditioned system is the same that the residual of the original system. As a consequence, stopping criteria are the same in preconditioned and the original systems. If we now call:

$$B := AM^{-1},$$
$$y := Mx,$$

the preconditioned system to be solved is:

$$By = b.$$

When the preconditioned system has been solved, the unknowns vector x is recovered by solving:

$$x = M^{-1}y.$$

Obviously, M has to fulfill two basic design requirements:

- Its inverse has to be easily computable (or the result of multiplying its inverse times a vector). In fact, computing the inverse of M has to be much cheaper than computing the inverse of A.

- $\kappa(AM^{-1})$ has to be as close to 1 as possible.

$M = A$ obviously fulfills the second requirement, since the condition number of the identity matrix is 1, but it does not fulfill the first requirement, that its inverse is easily computable. A very usual manner of building preconditioners is by means of *Incomplete Factorizations*.

8.7.2 ILUC, an Incomplete LU Crout factorization

Incomplete factorizations are performed by using algorithms similar to complete factorizations, the main difference being that not all the elements in L and U are kept. This reduces the memory required to store L and U and the number of operations required to perform the factorization. Obviously,

$$LU \neq A,$$

but hopefully

$$\kappa(A(LU)^{-1}) \approx 1.$$

The dropping strategies followed to build the preconditioners for the iterative methods are of two kind:

- Limiting fill-in: in each row/column k the number of non-zero entries to be stored is limited to $nfil$. This means that only a maximum of $nfil$ components is stored, for each k, in uC and lC. We choose the ones with greatest absolute value.

- Setting a drop tolerance. Only the values of the factorized matrices

  $$l_{jk}, \quad u_{kj}, \quad k < j \leq n,$$

  such that

  $$\max |l_{jk}, u_{kj}| > tol * |a_{kk}|$$

  are kept.

Compared to other preconditioners like ILUT [120] (the preconditioner associated to the CSR storage format), ILUC [95] has the advantage of computing the sort in line 57 of Algorithm 9 outside the i loop. In ILUC this SORT operation can be done by means of a $\mathcal{O}(nfil)$ computation, while in other preconditioner algorithms as ILUT the equivalent operation complexity is of $\mathcal{O}(nfil \times nfil)$. This allows for the ILUC preconditioner to admit a much greater number of fill-ins with much less increase of the CPU time required to build the preconditioner.

The values dropped during the incomplete factorization are added to the diagonal of U. In particular, entries dropped in L_{jk} are added to U_{jj}, while dropped entries in U_{kj} are added to U_{kk}. This enhances the behaviour of the preconditioner, since for constant solution vectors the factorization is exact: let us consider a solution vector x of the form:

$$x = [x_1, x_2, ..., x_n]^T \quad | \quad x_1 = x_2 = ... = x_n.$$

By definition:

$$Ax = b,$$

but for a constant solution vector x the following also holds:

$$Dx = b,$$

where D is a diagonal matrix its diagonal entries being:

$$d_{ii} = \sum_{j=1}^{n} a_{ij}$$

This is the reason why adding the dropped entries to the diagonal of U leads to a better preconditioner.

8.8 Renumbering strategies

Renumbering strategies are a critical issue in solvers for linear systems of equations. No matter how efficiently the solver algorithms are coded, the cost of solving the linear system will be unaffordable unless a proper renumbering strategy is used.

Renumbering strategies respond to two basic objectives:

- In direct solvers, renumbering must minimize the amount of memory and the number of operations required to factorize the matrix.
- In iterative solvers and for a given amount of memory assigned to the preconditioner, renumbering must lead to the best possible preconditioner.

As we can see the objective of renumbering is different depending on the type of solver we are using, and different renumbering strategies should be used for direct and iterative solvers.

8.8.1 Nested dissection

Nested dissection is the optimum renumbering strategy for direct solvers for sparse storage schemes. It minimizes the amount of fill-in, which implies that the amount of memory required to perform the factorization of the matrix is minimum, and so are the number of operations and the CPU time required to solve the system of equations.

In the nested dissection algorithm the finite element mesh is partitioned by means of domain decomposition techniques. If the nodes which separate the different subdomains are assigned node numbers greater than the nodes in the interior of the subdomains, fill-in does not occur between subdomains. If the resulting subdomains are recurrently partitioned, we obtain the nested dissection algorithm. Interior nodes of each final level subdomain are numbered successively, using for example a minimum degree ordering.

When using direct solvers, we use the METIS package [84], which provides an optimal node ordering using nested dissection techniques.

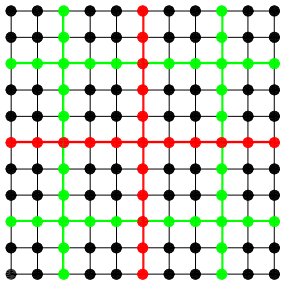


Figure 8.5: Two level nested dissection. The first level separator nodes are depicted in red, while the second level separators are depicted in green.

8.8.2 Cuthill-McKee reordering

In the Cuthill-McKee renumbering strategy, instead of minimizing the amount of fill-in, we minimize the bandwidth, that is to say, the maximum difference between the node number of connected nodes. The Cuthill-McKee algorithm is not as efficient as the nested dissection algorithm for direct methods, but it works much better in the case of iterative solvers. This is due to the fact that in the case of the nested dissection algorithms, *information* is concentrated in separator nodes, and thus if dropping occurs in fill-ins due to separator nodes the quality of the incomplete factorization is affected. In the Cuthill-McKee reordering, on the other hand, information is equally distributed, resulting in a much more efficient reordering for iterative solvers.

In order to minimize the bandwidth, node numbering starts from an edge node. The first layer of nodes corresponds to all the nodes connected to the first node. The second layer corresponds to all the nodes connected to the first layer which do not belong to any layer yet, and so on. A second possibility is to mark as first layer nodes all the nodes in the border of the mesh. An example of the Cuthill-McKee algorithm can be seen in Fig. 8.6.

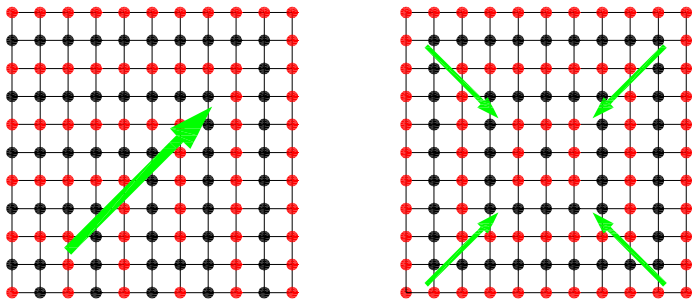


Figure 8.6: Cuthill-McKee ordering. Successive layers are depicted in different colors. Left: Cuthill-McKee starting from one node. Right: Cuthill-McKee starting from a first layer of nodes.

8.8.3 Multigrid reorderings

Although the Cuthill-McKee reordering algorithm provides a suitable ordering for iterative solvers, there are a family of reordering techniques which give the optimal orderings for incomplete factorizations in iterative solvers. These are multigrid reorderings. In multigrid reorderings the nodes of the mesh are separated into nodes belonging to fine and coarse meshes. Nodes belonging to fine meshes are numbered first while nodes belonging to coarse meshes receive the larger node numbers. Nodes in the fine mesh are factorized first. In a complete factorization for direct solvers this implies that the Schur complement of the fine mesh onto the coarse mesh is computed. This can be understood as solving the equivalent problem in the coarse mesh and projecting back the results onto the fine mesh, which is the basis of multigrid algorithms. In multigrid algorithms the complexity of the operations required to solve the system of equations increases linearly with the number of unknowns. This makes this kind of reordering very suitable for iterative methods. A discussion on some methods for building multilevel reorderings for incomplete factorization preconditioners can be found in [101]

We have developed our own multigrid renumbering algorithm, based on the graph of the mesh. We take advantage that our system of equations arises from a finite element analysis and we start from nodes in which Dirichlet boundary conditions are prescribed. These are in general nodes on the boundary of the mesh. It is very convenient to start the matrix factorization from these nodes, since equations attached to them are diagonal and cause no fill-in. From the Dirichlet nodes we build a provisional numbering by means of the Cuthill-McKee algorithm by considering Dirichlet nodes to be the first layer of nodes.

Now the first two layers are considered to be fine. After this, and following the Cuthill-McKee order, the coarsening algorithm starts. Algorithm 13 shows this coarsening algorithm. It consists of the following: when an unclassified node is found, it is classified as Coarse, and all of its unclassified neighbors are classified as Fine. This ensures that there will not be any pair of neighbor nodes classified as Coarse.

Algorithm 13 Coarsening algorithm

```
1: for i = 1 : n do
2:    if i is unclassified then
3:       Classify i as Coarse
4:       for j = iA(i) : iA(i + 1) − 1 do
5:          if jA(j) is unclassified then
6:             Classify jA(j) as Fine
7:          end if
8:       end for
9:    end if
10: end for
```

This procedure can be recursively repeated on the coarse mesh. The coarse mesh graph is built by considering that two Coarse nodes are connected if they have a common neighbor according to the previous level mesh graph. The final ordering is obtained by numbering first the nodes on the finer mesh, then the nodes on the first level coarse mesh, etcetera, and finally the nodes of the coarsest mesh. For each level, the nodes are numbered following the original Cuthill-McKee ordering. Fig. 8.7 shows the three level coarsening of an example mesh.

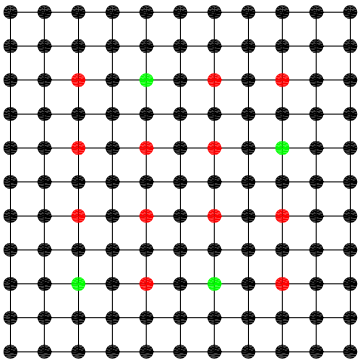


Figure 8.7: Multigrid ordering. Fine, intermediate and coarse mesh for a three level coarsening. Fine nodes are depicted in black, intermediate nodes are depicted in red and coarse nodes are depicted in green.

8.9 Block computations for problems with more than one degree of freedom per node

All the computations we have described for a scalar matrix can be extended to matrices arising from vectorial problems, that is to say, to finite element problems with more than one degree of freedom per node. The strategy we follow is to use the graph of the mesh, which corresponds exactly to the graph of the matrix in the case of scalar problems, to access the non-zero entries of the matrix arising from vectorial problems. This is done simply by understanding that each connectivity in the mesh graph refers to a *non-zero block* instead of a single non-zero entry. Now, each scalar operation in the factorization algorithms, matrix times vector or forward and backward substitutions turns into a matrix operation. As simple as it sounds, there are a number of issues that have to be taken into account:

- Since we want our matrix storage scheme to be *symmetric*, the storage of blocks in the upper triangular part of the matrix has to be the transpose of the storage of blocks in the lower triangular part of the matrix. This means that if full blocks are stored by rows in the upper part, they must be stored by columns in the lower part.

- In the factorization of the matrix (Algorithm 9) we store the inverse of the diagonal and we multiply the row (the part of it in the upper triangular part) by it. This operation requires some attention since now we are dealing with a diagonal block. Instead of computing the inverse of the block (which could be done since the dimension of the block is small enough in most cases) we prefer to perform a LU decomposition of each diagonal block. Now multiplying the row by the inverse of the diagonal turns into applying a backwards substitution to each block of the block row and a forward substitution to each block of the block column. The same happens when multiplying by the inverse of the diagonal in the backwards and forwards substitutions algorithms.

- In order to have a significative gain in the performance of the solver when compared to an scalar solver it is necessary that all the loops corresponding to block operations are *unrolled*. When dealing with a loop of the form:

	t_{ndof3}/t_{ndof1}	$t_{ndof3}/(9 \cdot t_{ndof1})$
Matvec+dot	4.19	0.4655
Backward-Forward Substitution	5.54	0.6155
ILUC	3.12	0.34

Table 8.1: CPU time ratio between scalar and 3 DOF vectorial problem

```
1: for i = 1 : ndof do
2:     ...
3: end for
```

the programming languages treat it in a different way depending on $ndof$ being a *variable* or a *parameter*. If $ndof$ is a parameter its value is known during compilation time. As a consequence the loop can be unrolled, meaning that the processor will not need to ask if it has to end the loop at each iteration, since the number of iterations is known in compilation time and they can be written one after the other instead of inside a loop. On the other hand, if $ndof$ is a variable its value is not known in compilation time, so in execution time the processor will have to ask if the loop has ended at the end of each iteration. Treating $ndof$ as a parameter is a key ingredient in the success of the by blocks formulation. This would mean to rewrite each routine for all the possible degrees of freedom per node. Fortunately, this can be avoided by means of the use of the so called *templates*, which allow to duplicate a routine for different parameters without replicating it.

This is a very useful approach. First of all, there is no need to store the particular graph for the matrix arising in each finite element problem, it is enough to store the graph of the mesh. At the same time, the non-numerical operations due to the compact sparse storage scheme are reduced by the number of degrees of freedom per node to the power of two. Finally, computations can be done by blocks and loops can be unrolled, which enhances the performance.

We have performed a small test in order to illustrate the increase in performance obtained by using this approach. We have built a 25000 element finite element mesh, and, on this mesh, we have solved a system of equations arising from a scalar problem, and a system of equations which arises from a 3 degrees of freedom vectorial problem. Both systems of equations share the matrix graph, and as a consequence, the number of floating point operations we have to perform in each routine of the 3 degrees of freedom problem is 9 times larger (3*3) than the number of operations needed in the scalar problem. However, the CPU time ratio between both problems is smaller than 9 thanks to the use of block computations. Table 8.1 summarizes the gain obtained by performing block computations.

In this case the time is reduced to the half approximately. For a greater number of degrees of freedom per node, for example the 3D incompressible Navier-Stokes equations with 4 degrees of freedom per node, the gain will be even larger.

8.10 Some numerical examples

In this section we present some numerical examples in which the FELAP package has been applied to solve the linear systems of equations arising from finite element analysis problems. It is not our intention to deeply analyze the performance of the various algorithms but to give an overview of the package capabilities.

8.10.1 Poisson problem

We first present an example of the performance of the solver routines on the Poisson problem. This is one of the simplest examples if arising from a thermal problem, but it also corresponds to the pressure phase in fractional step methods used to deal with the Navier-Stokes equations. The differential equation to be solved is:

$$-\Delta u = f,$$

with the proper boundary conditions.

The Poisson problem is solved in a 2D structured square mesh, in the border of which the unknown is prescribed to zero. Fig. 8.8 shows the performance of the Conjugate Gradient iterative solver. The solver stops when the relative error is 10^{-8}. In this plot we compare the performance of the algorithm using the multigrid renumbering strategy. If we compare the slopes of the different cases, we see that with diagonal preconditioning the global complexity is of $\mathcal{O}(n^{1.5})$. The slope diminishes slightly if the ILUC preconditioner is used (fill-in 30) but the drops are not added to the diagonal. The best result is obtained for the multigrid reordering, the ILUC preconditioner and diagonal compensation, for which the global complexity is of $\mathcal{O}(n^{1.13})$, close to the desired linear complexity.

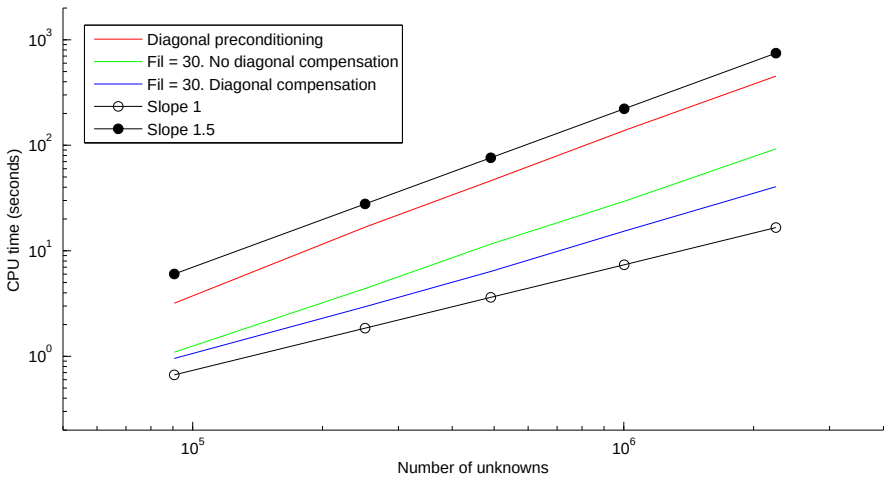


Figure 8.8: Computational cost versus number of unknowns for a 2D Poisson problem.

8.10.2 Stokes problem

Here we present an example on the stationary Stokes problem. The differential equation to be solved is:

$$-\Delta \boldsymbol{u} + \nabla p = \boldsymbol{f},$$
$$\nabla \cdot \boldsymbol{u} = \boldsymbol{0},$$

with the suitable boundary conditions.

We have considered the typical 3D cavity problem in which a confined flow occurs in a cube in the faces of which velocity is prescribed to zero except for one of the faces, in which it is prescribed to 1 in one of the axes tangent to the face. We use a stabilized formulation which allows us to use equal ($P1/P1$) interpolation for both velocity and pressure.

This is an example in which we can make use of block computations: for each connectivity of the graph we have a 4×4 block and compact sparse non-numerical operations are reduced by 16.

Firstly we will consider solving this problem with a direct solver. For this we perform a complete LU by blocks factorization, using the nested dissection renumbering provided by $METIS$. We consider a 48000 elements mesh, for which the memory requirements if using a direct solver are close to 2 Gbytes. In this case our solver takes 26 seconds to solve the system.

We compare this time against the commercial direct solver code $MUMPS$, which takes only 17 seconds: $MUMPS$ is a multifrontal solver. Its main strategy consists in grouping non-zeros in large blocks, which results in a good performance, even at the cost of some extra memory requirement and the storage of extra zero entries. However, this kind of strategy is not convenient for incomplete factorizations, since it is not known a priory which entries of the factorization will be kept. This is the reason why we do not rely on multifrontal strategies. However, our by-blocks strategy leads to good enough results for the direct solver.

Let us now solve the same system of equations with the iterative solvers. The system of equations arising from this problem is non symmetric (it can be symmetrized, but then it is not definite), thus, we use the BICGSTAB accelerator. We use the multigrid reordering strategy. In this case we take only 2 seconds to reach the 10^{-8} residual. Moreover, the amount of memory required for the preconditioner (fill-in 10) is 20 times smaller than the memory required in the complete factorization. Thus, iterative solvers are useful both for reducing storage and CPU time requirements if we are considering large systems of equations. Fig. 8.9 shows the computational cost versus number of unknowns plot. In this case it is not convenient to use diagonal compensation, since, at least for the block ordering we have, it may cause the preconditioner to fail due to zeros on the diagonal of the factorized matrices. The complexity is of $\mathcal{O}(n^{1.5})$. Diagonal preconditioning is not plotted since the solver does not converge for very fine meshes.

8.10.3 Navier-Stokes equations

Finally we turn into a real problem, in which the incompressible Navier-Stokes equations have to be solved:

$$\partial_t \boldsymbol{u} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} - \nu \Delta \boldsymbol{u} + \nabla p = \boldsymbol{f},$$
$$\nabla \cdot \boldsymbol{u} = \boldsymbol{0},$$

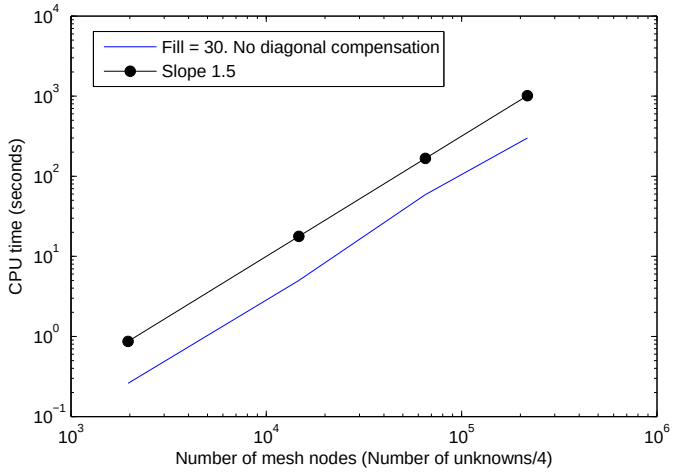


Figure 8.9: Computational cost versus number of unknowns for a 3D stationary Stokes problem.

with the suitable boundary conditions.

The aim of this example is not to study the algorithms performance but to see how the $FELAP$ general structure can be used to deal with different kinds of problems. When dealing with the incompressible Navier-Stokes equations in real-life simulations it is very convenient to use fractional step methods, which uncouple the Navier-Stokes equations, with 4 unknowns per node, into 4 scalar systems of equations of 1 unknown per node. $FELAP$ allows to easily deal with this problem by attaching the four systems of equations to the same finite element mesh, with which they share the graph.

In the example presented in Fig. 8.10, $FELAP$ has been attached to a computational fluid dynamics code ($FAUST$), and used to solve the systems of equations which arise from the finite element simulation of the flow in a water tank. The objective of the simulation is to compute the chlorine concentration. Water enters the tank through a hole in the upper part of the tank and leaves the tank through the holes in the lower part of the tank. For the transient analysis, a fractional step method has been used.

8.11 Conclusions

In this chapter we have introduced the $FELAP$ package to deal with the linear systems of equations arising from finite element analysis problems. The main features of the package are its capability to work with symmetric and unsymmetric systems of equations, direct and iterative solvers and various renumbering techniques, including a mix of Cuthill-McKee and multigrid type reorderings. Performance is enhanced by considering the finite element mesh graph instead of the matrix graph, which allows to perform highly efficient block computations. This graph is stored in a particular way so that it is suitable for both CSR and CS-Crout storage formats. Some numerical examples have been presented showing the capabilities of the package.

However, a very important point in linear system solvers is the possibility of performing

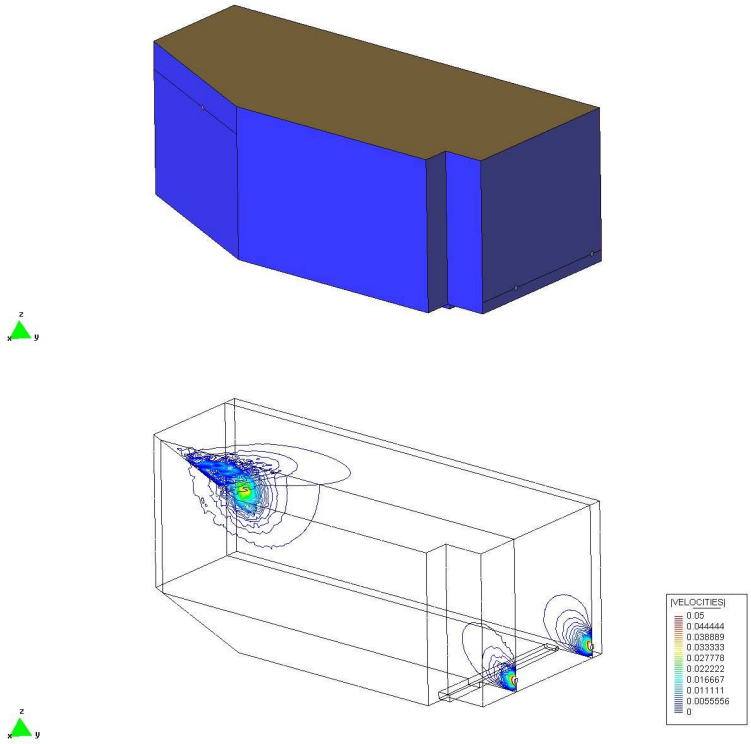


Figure 8.10: Real 3D problem. Incompressible Navier-Stokes equations solved by means of a fractional step method.

parallel computations, which are unavoidable if large system of equations are to be solved. Future work will be developed so that the package is able to do large scale parallel computations.

Chapter 9

Final 3D examples

9.1 Introduction

In this chapter we extend the methods proposed in the previous chapters to three dimensional problems and we apply them to solve some numerical examples. Special attention is paid to some of the implementation issues of the Fixed Mesh-ALE method in 3D. The linear systems of equations which arise from these 3D problems are solved with the *FELAP* solver.

9.2 Mesh-Mesh intersection in 3D

In this section we deal with the implementation of the algorithms for finding the intersection between two finite element meshes which superimpose in space. These algorithms are needed in order to find the integration domain for the finite element equations in the Fixed Mesh-ALE method. There are two main steps needed to compute mesh-mesh intersections:

- In order to find the intersection between two different meshes we will need to find the intersection between the elements of the fixed mesh (corresponding to the fluid), and the elements of the moving Lagrangian mesh (corresponding to the solid geometrical definition). If we check the intersection of all the elements of the fluid mesh against all the elements of the Lagrangian mesh the number of tests at each time step is of $\mathcal{O}(nelem_{fixed} \times nelem_{lag})$, where $nelem_{fixed}$ is the number of elements of the fixed mesh and $nelem_{lag}$ is the number of elements of the Lagrangian mesh. This is not affordable, and therefore a search strategy which allows us to reduce the number of intersection tests is needed.

- The second step is the actual intersection test between elements, with which we obtain the intersection surfaces between two elements.

9.2.1 Spatial search strategy

The problem we need to solve is: given an element on the fixed mesh, which are the elements of the Lagrangian mesh which are *close* to the fixed mesh element and could potentially intersect

it? By knowing this, we can reduce the number of intersection tests needed for each fixed mesh element from $nelem_{lag}$ (which could be millions!) to just a few elements.

There are several spatial strategies available to deal with this problem, amongst them the well known quadtree and octree algorithms (see [51]). This search strategies are very convenient when data is non uniformly distributed in space, for example when the mesh is strongly refined in a certain region of the space. In our case, however, we will not implement these tree structures, and instead we will use a simple *bins* strategy which consists of two steps:

- Preprocess: The bins strategy consists of uniformly partitioning the domain into several cells, rectangles (in 2D) or rectangular prism (in 3D). Each of these cells is assigned an identity number, and a data structure is created in order to store the fixed-mesh elements which are contained in the cell. Fig. 9.1 shows a uniform partition example for a 2D square domain. The element in the figure would be included in cells F6, F7, G5, G6, G7, H5, H6, H7 and H8. For ease of implementation, we also include the element in cells F5, F8 and G8: in this way we can check wether an element is contained in a cell just by computing the minimum and maximum x, y and z coordinates of the element nodes. For fixed meshes, this can be done once at the beginning of the computation.

- Search: when we want to test the intersection of a Lagrangian element against the fixed mesh elements, we check which are the cells of the bins partitioning in which the Lagrangian element is contained, and we perform the intersection test only against the fixed elements which are also inside these cells.

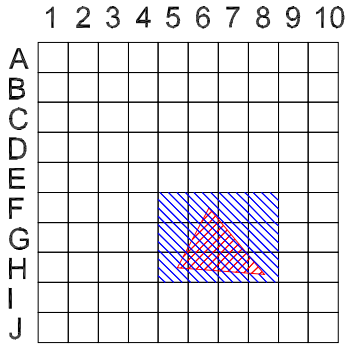


Figure 9.1: Bins strategy for a fast element intersection algorithm. The element in red is contained in the blue cells.

9.2.2 Element-element intersection

In the examples in the following sections we work with linear tetrahedra for both the fluid and the solid problems. We will describe here the algorithms used to compute tetrahedron-tetrahedron intersections. These algorithms can be adapted to higher order elements.

As explained in Chapter 4, in the FM-ALE method we integrate the finite element equations only over the region of each element which is inside the physical domain. We need to find the intersection between the boundary faces of the Lagrangian mesh and the fixed mesh elements, as illustrated in Fig. 9.2. To do this we test the intersection of all the faces of the fixed element against the boundary faces of the Lagrangian element. The information we keep from the intersection are the intersection points on the *edges* of the fixed mesh elements.

Figure 9.2: Intersection between a fixed mesh element (green) and a boundary face in a Lagrangian element (blue).

For the triangle-triangle intersection tests we use the algorithm proposed in [105], which is widely used in the computer graphics community. This algorithm is intended to check wether two triangles intersect in space, but it does not actually compute the intersection segments or lines. We have slightly modified it so that we obtain also the intersection points. The algorithm reads as follows:

Let us denote the two triangles T_1 and T_2, and its vertices by $\boldsymbol{V}_0^1$, $\boldsymbol{V}_1^1$ and $\boldsymbol{V}_2^1$ for T_1 and $\boldsymbol{V}_0^2$, $\boldsymbol{V}_1^2$ and $\boldsymbol{V}_2^2$ for T_2.

- Find the equations of the planes containing T_1 and T_2, which we denote by π_1 and π_2:

$$\begin{aligned} \pi_i : \boldsymbol{N}_i \cdot \boldsymbol{X} + d_i = 0 & \\ \boldsymbol{N}_i = (\boldsymbol{V}_1^i - \boldsymbol{V}_0^i) \times (\boldsymbol{V}_2^i - \boldsymbol{V}_0^i) & \\ d_i = -\boldsymbol{N}_i \cdot \boldsymbol{V}_0^i & \end{aligned} \tag{9.1}$$

  where $\boldsymbol{X}$ is any point on π_i.

- Compute the signed distance from the vertices of T_1 to π_2 by inserting them in the equation for π_2:

$$d_{V_j^1} = \boldsymbol{N}_2 \cdot \boldsymbol{V}_j^1 + d_2, j = 0, 1, 2 \tag{9.2}$$

- Compute the signed distance from the vertices of T_2 to π_1 in the same manner.

- If all the $d_{V_j^1}$ have the same sign then T_1 lies on one side of π_2 and the test is rejected. We do the same for T_2.

- We can now compute the intersection points between T_1 edges and π_2. We do this only for the triangle edges in which the sign of $d_{V_j^1}$ is different for each of the vertices of the edge (that is to say, the vertices lie in different sides of π_2). The intersection point for the edge connecting V_i^1 and V_j^1 can be found by computing:

$$\boldsymbol{I}_{ij}^1 = \boldsymbol{V}_i^1 + \frac{d_{V_i^1}}{d_{V_i^1} - d_{V_j^1}} \cdot (\boldsymbol{V}_j^1 - \boldsymbol{V}_i^1) \tag{9.3}$$

  This yields two intersection points for T_1. We do the same for T_2.

- Now we have the intersection line segments for T_1 and T_2. This segments are both on the same line, as shown in Fig. 9.3. We only need to check wether the intersection points of T_1 are contained on the intersection interval of T_2, and vice versa. This can be done by computing the line equation and assigning a line parameter λ to each intersection point:

$$\boldsymbol{X} = \boldsymbol{I}_{ij}^1 + \lambda \cdot (\boldsymbol{I}_{ij}^2 - \boldsymbol{I}_{ij}^1) \tag{9.4}$$

  where $\boldsymbol{I}_{ij}^1$ and $\boldsymbol{I}_{ij}^2$ are two of the previously computed intersection points.

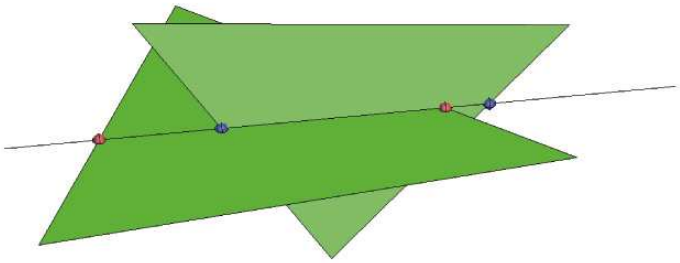


Figure 9.3: Intersection between two element faces. Intersection line between π_1 and π_2 and intersection points on the line. Blue points correspond to T_1 intersection points, red points correspond to T_2 intersection points.

9.3 Subelement integration in 3D tetrahedra

In the previous section we have seen how to find the intersection points between fixed elements and Lagrangian elements. The intersection algorithm returns the intersection points in the edges of the tetrahedron. We now need to subintegrate inside the elements. There are three different subintegration cases:

- One of the tetrahedron nodes is inside the physical domain and the other three nodes are outside.

- Two of the tetrahedron nodes are inside the physical domain and the other two nodes are outside.

- Three of the tetrahedron nodes are inside the physical domain and the other one is outside.

Fig. 9.4 shows each of these subintegration cases, and how the subintegration region is decomposed into subelements in order to properly integrate in this region.

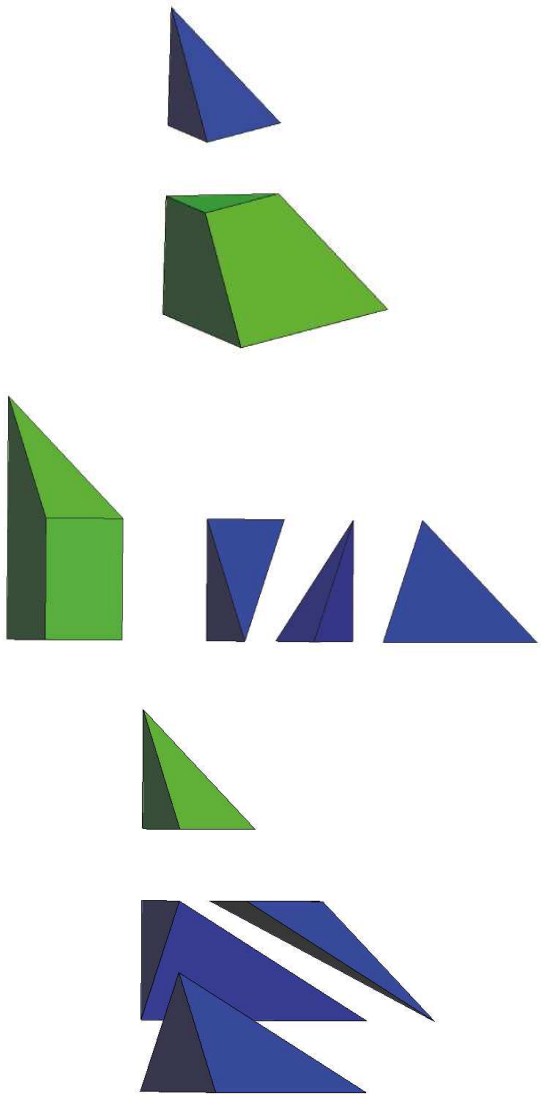


Figure 9.4: Subelement integration. Blue: integration region. Green: discarded region. Top: 1 node in case, integration region decomposed into 1 subelement. Middle: 2 node in case, integration region decomposed into 3 subelements. Bottom: 3 node in case, integration region decomposed into 3 subelements.

9.4 Numerical examples

9.4.1 Flow over a bending plate

In this section we consider the incompressible flow over a bending plate. The considered plate is set inside a channel with dimensions $3 \times 0.5 \times 1$, and the plate dimensions are $0.1 \times 0.4 \times 0.6$. Boundary conditions are set to $u_x = 1$ in the inflow wall, and $\boldsymbol{u} = \mathbf{0}$ in the lateral walls. A *do nothing* (zero-traction) boundary condition is set in the outflow. In the interface between fluid and structure the usual non-slip Dirichlet boundary conditions and continuity of tractions condition are applied. Fluid viscosity is set to $\nu = 0.01$, yielding a Reynolds number $Re = 100$ based on the inflow velocity and the channel width. For the structure we consider a Young modulus of $E = 500$ and Poisson ratio $\nu = 0.48$. Although the plate undergoes large deformations in this problem setting, we only consider a linear elastic material for this simple illustrative example.

The strategies described in the previous chapters are used in order to perform the simulation, including the FM-ALE strategy, the strong imposition of boundary conditions in embedded grids strategy, the use of stabilized formulations for the computation of the fluid dynamics (although we do not apply the subscales on the boundary strategy here) and the use of the FELAP solver to deal with the linear systems of equations to be solved. A light preconditioner (sparsity ratio equal to 2) is computed prior to the solution of each linear system of equations in order to improve the performance of the GMRES iterations. We use a semi-implicit approach in which the domain for the simulation is computed explicitly at the beginning of each time step and implicit integration schemes (backward Euler for the fluid, Newmark-beta scheme for the solid) are used for the solution of the dynamics equations in each subdomain. An explicit iteration by subdomain approach is used to deal with the coupling between fluid and structure at each time step.

After an initial transient, the plate reaches an stationary position. Fig. 9.5 shows the final position of the bending plate and the velocity and pressure fields in the channel. We can observe that the fluid is forced to flow around the plate and at the same time exerts some pressure on the it which makes the plate bend. As soon as the flow has surpassed the plate it returns to the center of the channel.

Regarding the behavior of the linear systems of equations solver, the mean number of iterations needed to achieve convergence in the GMRES algorithm was 102, with little variation in the number of iterations between different time steps.

9.4.2 The water entry of a decelerating sphere

In this section we consider the 3D numerical simulation of an sphere falling into water. In particular we try to reproduce the experimental results reported in [3]. Several experiments are performed in this work, we will simulate the impact of a Nylon sphere into water. The problem setting is the following: a one inch (2.54 cm) diameter nylon sphere is dropped into a water tank. The tank has dimensions of $30 \times 50 \times 60\ \mathrm{cm}^3$ and the sphere is dropped from a $25\ \mathrm{cm}$ height, which yields an impact vertical velocity of $2.17\ \mathrm{m/s}$. Water density is $\rho = 1000\ \mathrm{kg/m}^3$, and its viscosity is $\mu = 0.00089\ \mathrm{Ns/m}^2$. Nylon density is $\rho_N = 1140\ \mathrm{kg/m}^3$, its Young modulus being $E = 3\ \mathrm{GPa}$, the Poisson coefficient $\nu = 0.2$.

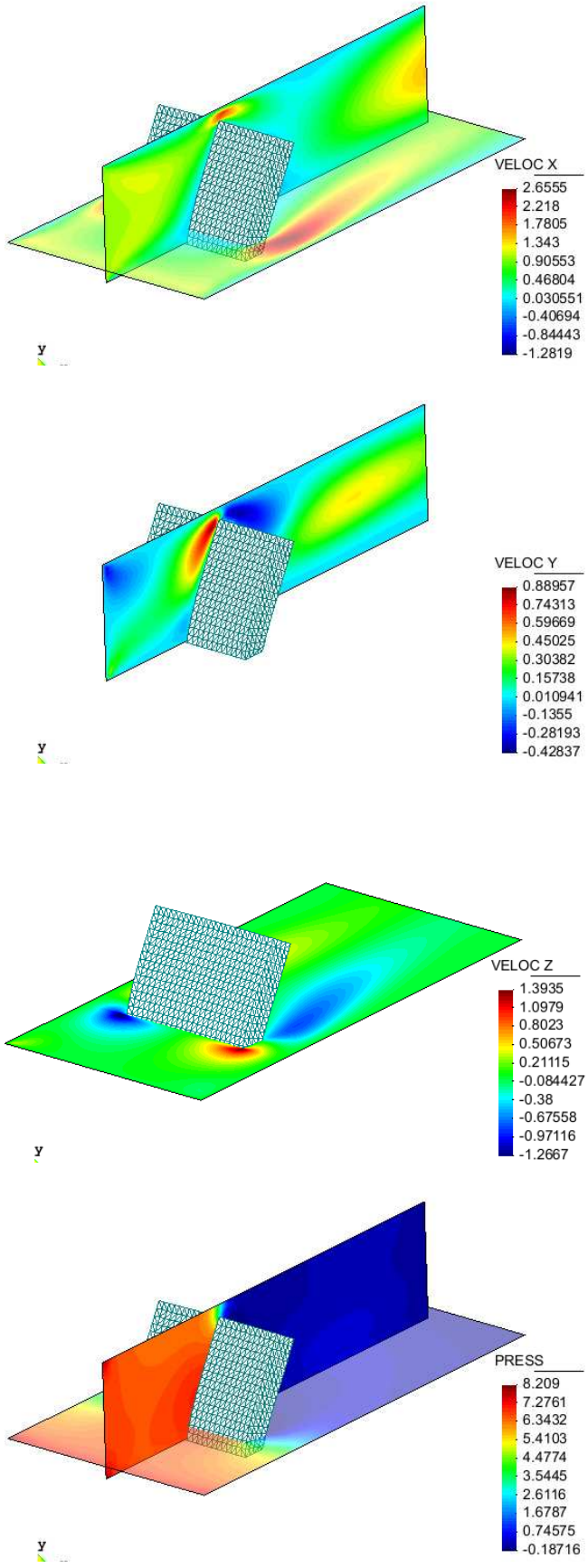


Figure 9.5: Velocity and pressure fields for the bending plate example after the stationary state is reached.

The physical domain of the water tank is much larger than the sphere, and in fact the sphere water impact has little effect on the flow in the boundaries of the tank. Taking this into account, and in order to minimize the required computational cost, we only simulate a region of $10 \times 10 \times 20\ \mathrm{cm}^3$. In order to take into account that the physical domain is much larger than the computational domain, we do not impose Dirichlet boundary conditions on the walls of the computational domain but we let water freely flow through. Instead, we impose Neumann boundary conditions on these walls in which we prescribe tractions to be:

$$\boldsymbol{\sigma} \cdot \boldsymbol{n} = \rho \boldsymbol{g} h \tag{9.5}$$

where ρ is the fluid density, $\boldsymbol{g}$ is the gravity field and h is the depth with respect to the original position of the free surface. Finally, we prescribe tractions to be null in the free surface:

$$\boldsymbol{\sigma} \cdot \boldsymbol{n} = \mathbf{0} \tag{9.6}$$

For the boundary conditions in the contact between the sphere and the water free surface, we use slip boundary conditions. Slip boundary conditions are a good approximation to the real boundary conditions due to the following:

- In order to promote cavity formation and minimize drag spheres were sprayed with an hydrophobic coat in the experiments.
- The cartesian meshes used to perform the numerical experiments are not capable of reproducing the boundary layer in the fluid- sphere contact.

Moreover, and due to the presence of the hydrophobic spray coat, we do not allow the boundary conditions to prevent the water surface from separating from the sphere in the upper half of the solid body. At each iteration we test wether the water surface is trying to separate from the solid body. If it is trying to separate, then Neumann (instead of slip) boundary conditions are applied in the contact surface. Surface tension is not considered in the simulations, although according to [3] its effect can be neglected at the considered impact speeds. A schematic of the boundary conditions of the problem can be found in Fig. 9.6.

The strategies described in the previous chapters are used in order to perform the simulation, including the FM-ALE strategy, the strong imposition of boundary conditions in embedded grids strategy, the use of stabilized formulations for the computation of the fluid dynamics (although we do not apply the subscales on the boundary strategy here) and the use of the FEAP solver to deal with the linear systems of equations to be solved. A robust preconditioner (sparsity ratio equal to 5) is computed prior to the solution of each linear system of equations in order to improve the performance of the GMRES iterations. We use a semi-implicit approach in which the domain for the simulation is computed explicitly at the beginning of each time step and implicit integration schemes (backward Euler for the fluid, second order Newmark for the solid) are used for the solution of the dynamics equations in each subdomain. The time step is set to $\delta t = 0.002\,\mathrm{s}$. An explicit iteration by subdomain approach is used to deal with the coupling between fluid and structure at each time step.

The Reynolds number is $Re = 64844$ based on the impact speed, the sphere diameter and the water viscosity. The LES Smagorinsky model is used in order to take into account the turbulent subscales (see, e.g. [114, 123, 39] for background). This model is tight to the

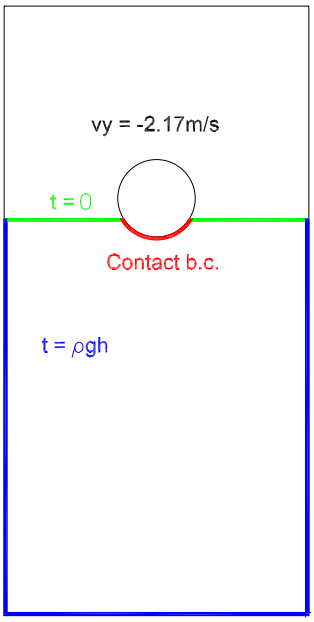


Figure 9.6: Problem setting for the water entry of a decelerating sphere.

numerical discretization in space of the flow equations, which in our case is performed using the finite element method. The turbulent kinematic viscosity associated to this model is

$$\nu_{\text{tur}} = \rho_0^{-1} c h^2 \left[\nabla^s(\boldsymbol{u}) : \nabla^s(\boldsymbol{u})\right]^{1/2},$$

where c is a constant, usually taken as $c = 0.01$, the colon stands for the double contraction of second order tensors and h is the length of the element of the finite element discretization described later where the turbulent kinematic viscosity is to be computed. The total viscosity will be $\nu = \nu_{\text{mol}} + \nu_{\text{tur}}$, ν_{mol} being the molecular viscosity.

Two different meshes are used to perform the computations. Although qualitatively correct results are obtained with a 514000 element mesh, a better numerical approximation for the time evolution of the position of the sphere is obtained with a 2469600 element mesh. Fig. 9.7 shows the time evolution of the sphere vertical position for the experimental results and the numerical solution. A quite good agreement is obtained for the finer mesh. Fig. 9.8 shows the velocity and pressure fields at some representative time instants. Immediately after impact, the sphere pushes water radially out, and the cavity behind it increases its size as time evolves (first row). At a certain point, gravitational force compensates for the momentum the sphere has transmitted to water and the cavity begins to collapse (second row). After cavity collapse, two vertical jets pointing upwards and downwards are formed due to the water incompressibility constraint (third row). The upwards pointing jet achieves a vertical velocity of the same order of the sphere impact speed (forth row). Several snapshots are presented in Fig. 9.9, where the numerical and the experimental shape of the free surface are compared. We can conclude that the physical phenomena is being correctly represented. Despite the large Reynolds number a smooth free surface is obtained, and the collapse of the cavity behind the sphere due to gravitational forces is correctly recovered.

Regarding the solution of the linear system of equations, most of the problems to be solved

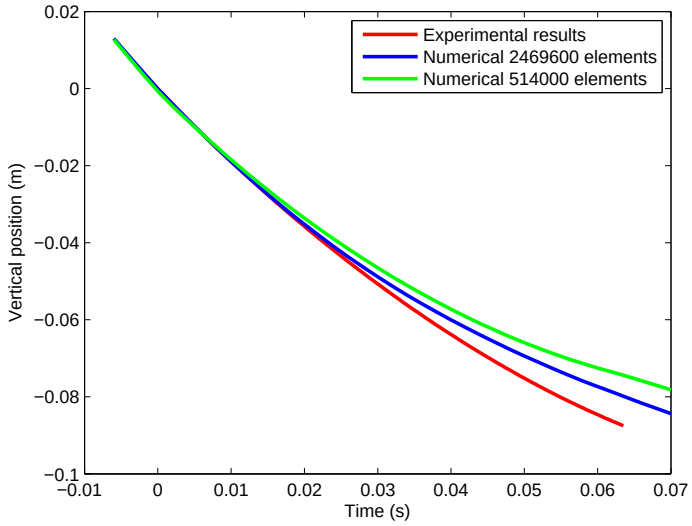


Figure 9.7: Comparison between experimental and numerical position of the sphere

were easily solved with the FELAP package: the solid dynamics problem, the advection of the level-set function, the L_2 projection involved in the FM-ALE method... The most challenging problem, as expected, was the solution of the transient Navier-Stokes equations. The number of iterations required to achieve convergence (which was set to a relative residual of 10^{-10}) was between the range of 30-200 iterations. The number of iterations was larger in critical steps like the initial impact instant and the cavity collapse instant, but there was also a certain randomness in the number of iterations per time step which was probably due to the dependence of the condition number on the way the boundary of the domain cuts the elements. Unexpectedly, the number of required iterations was smaller for the finer mesh, although the same sparsity ratio was used for building the preconditioner. This might be due to the fact that in this highly convection-dominated problem the finer mesh yields a more stable finite element matrix.

9.5 Conclusions

In this chapter we have applied the FM-ALE method to solve fluid-structure interaction problems in 3D. We have paid special attention to the algorithms needed to compute the mesh-mesh intersections and the subelement integration, which are a bit more complex when extended to 3D. The *FELAP* package for solving linear systems of equations has been used. The behavior of both algorithms has been tested in two numerical experiments with satisfactory results. We can conclude that the FM-ALE method and the *FELAP* package provide an interesting tool to deal with multiphysics problems in time evolving domains.

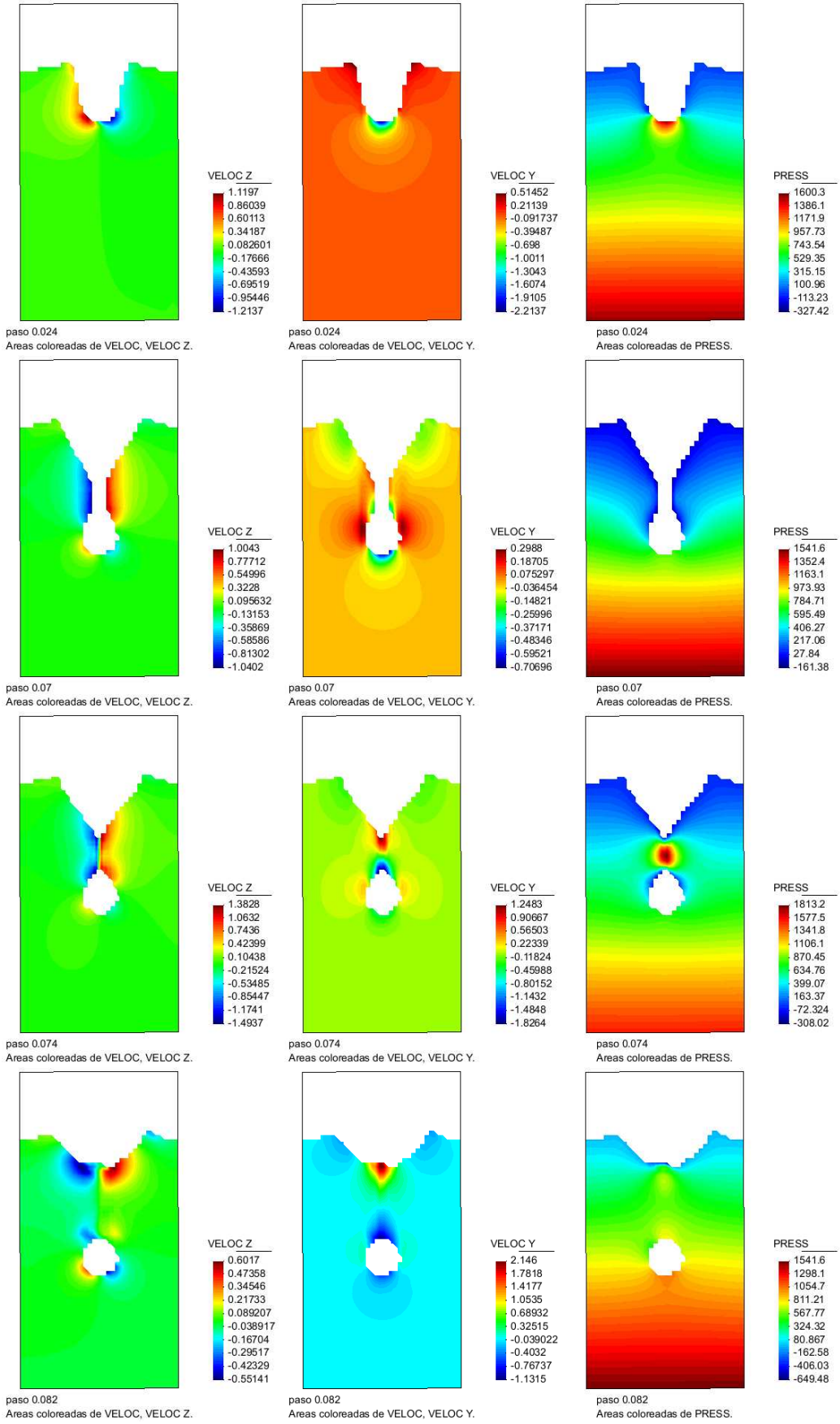


Figure 9.8: Velocity and pressure fields at different steps of the simulation

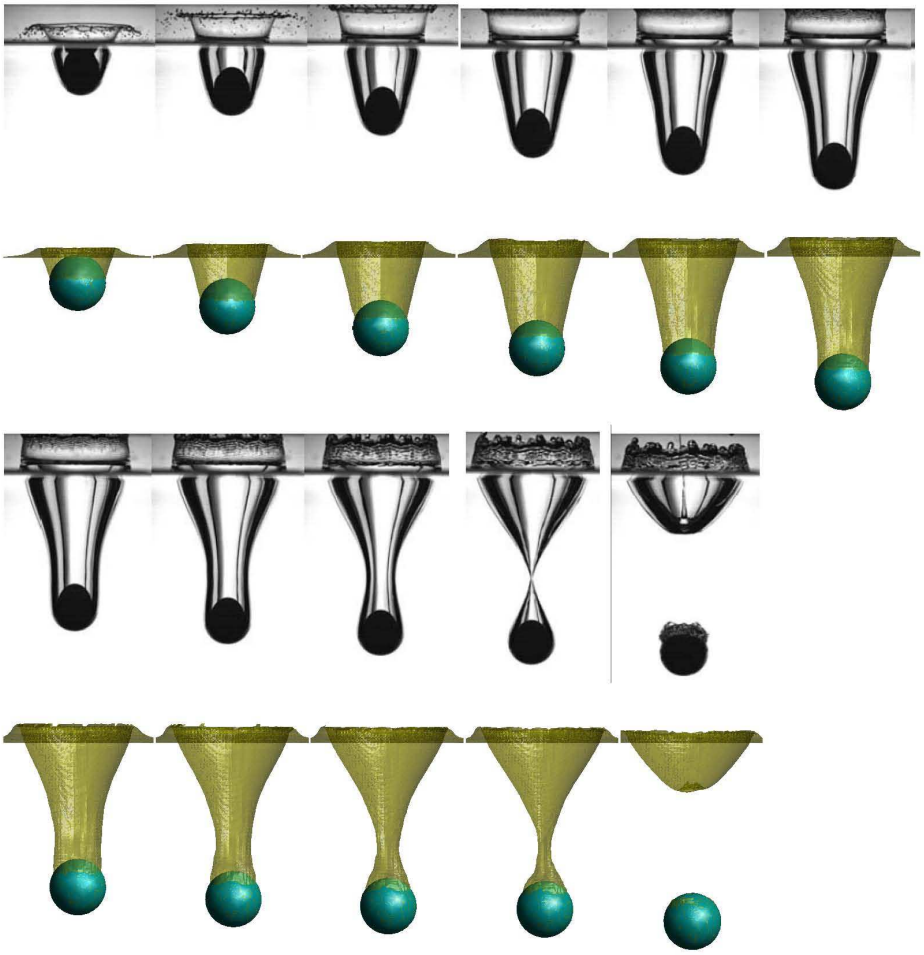


Figure 9.9: Comparison between experimental and numerical results

Chapter 10

Conclusions

In this chapter we present the achievements and conclusions obtained during the preparation of the present work, and we state some of the possible future lines of research.

10.1 Achievements

We have presented a series of works in the fields of fixed mesh methods and stabilized numerical formulations:

- In Chapter 2 we have proposed a way to strongly prescribe Dirichlet boundary conditions for immersed boundary methods. The main idea is to use as degrees of freedom for this imposition those associated to the nodes adjacent to the boundary of the computational domain. The method proposed turns out to be accurate (second order for linear elements) and robust. We have checked its numerical performance in a variety of situations in flow problems, paying particular attention to problems that require stabilization. From the implementation point of view, the method satisfies the main design condition of using only the degrees of freedom of the mesh of Ω_h.

- In Chapter 3 we have proposed a way to weakly prescribe Dirichlet boundary conditions in embedded grids. The key feature of the proposed method is that we do not need a large penalty parameter to ensure stability and that it is symmetric for symmetric problems. The method turns out to be accurate (second order for linear elements) and robust for all the problems tested except for the pure transport equation, in which we are not able to recover quadratic convergence. Further work needs to be developed to find a proper definition of the weighting terms for the imposition of boundary conditions in the pure transport equation. When compared to the method described in Chapter 2 we can conclude that both methods perform similarly well and are suitable for flow problems.

- In Chapter 4 we have introduced in detail the concept of the FM-ALE approach. which consists in using the standard ALE method but remeshing at each time step so as to use always the same given mesh. Ad-hoc approximations to account for the advection of information that can be found in several fixed-grid methods are avoided. This is in particular reflected by the treatment of the so called newly created nodes. Results have been

compared to those of other fixed grid methods, showing the need of correctly computing the advection of information between time steps.

- In Chapter 5 the FM-ALE approach has been applied to solid mechanics and Fluid-Structure Interaction problems. For solid mechanics problems the FM-ALE method is of special interest when the solid body is subject to very large strains. In this case Lagrangian formulations cannot be used due to the ill-conditioning caused by the large element stretch. The FM-ALE method, on the other hand, avoids element stretching by using a fixed mesh. Results show that the method is robust and accurate. In the case of Fluid-Structure Interaction problems, the FM-ALE method can be applied to solve the flow and the solid mechanics problems. The main feature of using this approach is the possibility of using a single background mesh to solve both mechanical problems. For free surface problems the FM-ALE method avoids the need for remeshing which appears in classical Lagrangian or ALE methods. Moreover, the free surface is tracked in a very natural way with the level set function strategy, allowing for the solid body *breaking the free surface* without any further algorithmic steps. We have paid special attention to the interaction between the level set function and the solid boundary function which define the fluid domain. The proposed method has been used to solve the problem of rigid bodies falling into water, and has proved to be robust and provide smooth solution fields, even at the critical instant in which the solid body contacts the free surface.

- In Chapter 6 we introduce a way to approximate the subscales on the boundaries of the elements in a variational two-scale finite element approximation to flow problems. The key idea is that the subscales on the element boundaries must be such that the transmission conditions for the unknown, split as its finite element contribution and the subscale, hold. The final result is that the subscale on the interelement boundaries must be proportional to the jump of the flux of the finite element component and the average of the subscale calculated in the element interiors. This allows for the use of discontinuous pressure interpolations in the Stokes problem, like for example $P1/P0$ elements.

- In Chapter 7 we have applied the subscales on the boundaries strategy to deal with domain interaction problems. Particular emphasis has been put here on the treatment of Neumann-type boundary conditions. The same ideas have been applied to the homogeneous interaction between two subdomains. In this case, the benefit of the boundary terms is a stronger enforcement of the continuity of fluxes between subdomains. The matrix structure of the resulting system has been described and iterative schemes to be used in an iteration-by-subdomain environment have been proposed. The fluid structure interaction problem has then been treated. The extension of the previous ideas to this case has led to a modification of what can be considered a classical solid-fluid iterative coupling. The boundary terms introduced, which cancel when convergence is achieved, would hardly be motivated from a purely algebraic point of view. All our predictions have been stated based on physical reasoning, without numerical analysis. Numerical experiments have confirmed the theoretical predictions. In particular, a better enforcement of the continuity of fluxes is found in homogeneous domain interaction problems and convergence of solid-fluid iterative coupling algorithms is greatly improved by the terms we suggest to introduce.

- In Chapter 8 the $FELAP$ package to deal with the linear systems of equations arising from finite element analysis problems has been presented. The main features of the package are its capability to work with symmetric and unsymmetric systems of equations, direct and iterative solvers and various renumbering techniques, including a mix of Cuthill-McKee and multigrid type reorderings. Performance is enhanced by considering the finite element mesh graph instead of the matrix graph, which allows to perform highly efficient block computations. This graph is stored in a particular way so that it is suitable for both CSR and CS-Crout storage formats. Some numerical examples have been presented showing the capabilities of the package.

- In Chapter 9 we have applied the FM-ALE method to solve fluid-structure interaction problems in 3D. We have paid special attention to the algorithms needed to compute the mesh-mesh intersections and the subelement integration, which are a bit more complex when extended to 3D. The *FELAP* package for solving linear systems of equations has been used. The behavior of both algorithms has been tested in two numerical experiments with satisfactory results. We can conclude that the FM-ALE method and the *FELAP* package provide an interesting tool to deal with multiphysics problems in time evolving domains.

10.2 Future work

We succinctly describe here the open lines of research:

- To apply the FM-ALE formulation to two phase flow problems. The classical way of dealing with two phase flow problems consists of solving the problem in an Eulerian manner, considering velocity, velocity gradients and pressure to be continuous across the surface which separates the two immiscible fluids. As in fact only the velocity field is continuous across this surface (not the velocity gradient nor the pressure), this causes spurious velocity fields to appear. Some methods have been devised to minimize this effect, in particular the *X-FEM* method which allows velocity and pressure to be discontinuous across the surface. The method we have presented in Chapter 5, which consists on duplicating the degrees of freedom of the nodes belonging to elements cut by the body border, can be understood as an X-FEM method which could be applied to two phase flow problems. However, our main contribution would be to treat the problem in an ALE framework near the discontinuity: as explained in subsection 4.2.3, values of the unknowns at the nodes of the first fluid are uncoupled from those at the second fluid, and as a consequence they cannot be used to compute temporal derivatives on nodes in the first fluid. The FM-ALE method copes with this problem by using ALE instead of purely Eulerian formulations, which we hope will avoid the spurious velocity fields to appear.

- To add some local-refinement capability to our FM-ALE code by means of the use of *hanging nodes*. In the FM-ALE method we favour the use of Cartesian grids due to the ease of generation and the fact that there is no need for the boundary of the mesh to match the boundary of the domain. However, it is still interesting to be able to have some

local refinement around the fluid-solid interface region. A way to achieve this without the need of using an unstructured mesh is to refine by subdividing elements in refined regions and couple the refined elements to the original coarse elements with the so called *hanging nodes*, and the use of *discontinuous Galerkin* type strategies.

- To develop and incorporate to the FM-ALE algorithm a library which includes a more precise mesh-mesh and levelset-mesh intersection algorithm. In all the numerical examples presented, mesh-mesh and levelset-mesh intersections were simplified to points in the edges of the elements. It would be interesting to improve the performance of the FM-ALE algorithm by using a library which is capable of correctly dealing with sharp edges and the complex to integrate geometries of fluid volumes in elements cut at the same time by the solid body boundary and the levelset function.

- To find a computationally efficient formulation for the stabilized $P1/P0$ element for the Stokes problem developed in Chapter 6. We have looked for an efficient implementation of the $P1/P0$ interpolation, which consists of condensing the pressure unknowns by sending the off-diagonal terms corresponding to the pressure test function equations to the right hand-side. Several iterative and explicit methods have been presented which are suitable for stationary and transient problems respectively. However, although some of these methods work, none of them shows a fast enough convergence to be competitive with the $P1/P1$ interpolation. Further research will be carried out in order to consider more complex iterative schemes (starting with, for example, Gauss-Seidel iterations) which might allow to condense the pressure unknowns and at the same time obtain convergence in very few iterations.

- To continue developing the $FELAP$ package of solvers for linear systems of equations arising from the finite element method analysis. The main structure and the basic tools of $FELAP$ have already been coded. However, some of the design requirements for $FELAP$ have not been reached yet, amongst them, to make the solver capable of dealing with parallel computations, which are unavoidable if large system of equations are to be solved.

Bibliography

[1] P. R. Amestoy, I. S. Duff, J. Koster, and J.-Y. L'Excellent. A fully asynchronous multifrontal solver using distributed dynamic scheduling. *SIAM Journal of Matrix Analysis and Applications*, 23(1):15–41, 2001.

[2] R. Araya, G. Barrenechea, and F. Valentin. Stabilized finite element methods based on multiscale enrichment for the Stokes problem. *SIAM Journal on Numerical Analysis*, 44:322–348, 2006.

[3] J.M. Aristoff, T.T. Truscott, A.H. Techet, and J.W.M. Bush. The water entry of decelerating spheres. *Physics of Fluids*, 22:032102, 2010.

[4] F. Armero and E. Love. An arbitrary Lagrangian-Eulerian finite element method for finite strain plasticity. *International Journal for Numerical Methods in Engineering*, 57:471–508, 2003.

[5] D.N. Arnold. An interior penalty finite element method with discontinuous elements. *SIAM Journal on Numerical Analysis*, 19:742–760, 1982.

[6] M. Astorino, F. Chouly, and M.A. Fernandez. An added-mass free semi-implicit coupling scheme for fluid-structure interaction. *C. R. Acad. Sci. Paris, Ser. I*, 347:99–104, 2009.

[7] S. Badia and R. Codina. Analysis of a stabilized finite element approximation of the transient convection-diffusion equation using an ALE framework. *SIAM Journal on Numerical Analalysis*, 44:2159–2197, 2006.

[8] S. Badia, F. Nobile, and C. Vergara. Fluid-structure partitioned procedures based on Robin transmission conditions. *Journal of Computational Physics*, 227:7027–7051, 2008.

[9] S. Badia, F. Nobile, and C. Vergara. Robin-Robin preconditioned Krylov methods for fluid-structure interaction problems. *Computer Methods in Applied Mechanics and Engineering*, 198:2768–2784, 2009.

[10] S. Badia, A. Quaini, and A. Quarteroni. Modular vs non-modular preconditioners for fluid-structure systems with large added-mass effect. *Computer Methods in Applied Mechanics and Engineering*, 197:4216–4232, 2008.

[11] S. Badia, A. Quaini, and A. Quarteroni. Splitting methods based on algebraic factorization for fluid-structure interaction. *SIAM J. Sci. Comput.*, 30 I4:1778–1805, 2008.

[12] C. Baiocchi, F. Brezzi, and L.P. Franca. Virtual bubbles and Galerkin/least-squares type methods (Ga.L.S). *Computer Methods in Applied Mechanics and Engineering*, 105:125–141, 1993.

[13] R.E. Bank and R.K. Smith. An algebraic multilevel multigraph algorithm. *SIAM Jounal of Scientific Computing*, 23:1572, 2002.

[14] H.J.C Barbosa and T.J.R Hughes. The finite element method with Lagrangian multipliers on the boundary: circumventing the Babuška-Brezzi condition. *Computer Methods in Applied Mechanics and Engineering*, 85:109–128, 1991.

[15] Y. Bazilevs and T.J.R Hughes. Weak imposition of dirichlet boundary conditions in fluid mechanics. *Computers and Fluids*, 36:12–26, 2007.

[16] R. Becker, P. Hansbo, and R. Stenberg. A finite element method for domain decomposition with non-matching grids. *Mathematical Modelling and Numerical Analysis*, 37:209–225, 2003.

[17] T. Belytschko, W.K. Liu, and B. Moran. *Nonlinear finite elements for continua and structures.* John Wiley, 2001.

[18] D.J. Benson. An efficient, accurate, simple ALE method for nonlinear finite element programs. *Computer Methods in Applied Mechanics and Engineering*, 72:305–350, 1989.

[19] M. Benzi. Preconditioning techniques for large linear systems: a survey. *J. Comput. Phys.*, 182:418–477, 2002.

[20] D. Boffi and L. Gastaldi. Stability and geometric conservation laws for ALE formulation. *Computer Methods in Applied Mechanics and Engineering*, 193:4717–4739, 2004.

[21] A. Brandt, S.F. McCormick, and J.W. Ruge. *Algebraic multigrid (AMG) for sparse matrix equations, in Sparsity and its Applications*. Cambridge University Press, 1984.

[22] S.C. Brenner and L.R. Scott. *The mathematical theory of finite element methods*. Springer–Verlag, 1994.

[23] F. Brezzi and M. Fortin. *Mixed and hybrid finite element methods*. Springer Verlag, 1991.

[24] E. Burman and M.A. Fernandez. Stabilization of explicit coupling in fluid-structure interaction involving fluid incompressibility. *Computer Methods in Applied Mechanics and Engineering*, 198:766–784, 2009.

[25] E. Burman, M.A. Fernández, and P. Hansbo. Continuous interior penalty finite element method for Oseen's equations. *SIAM Journal on Numerical Analysis*, 44:1248–1274, 2006.

[26] E. Burman and P. Hansbo. Edge stabilization for Galerkin approximations of convection-diffusion-reaction problems. *Computer Methods in Applied Mechanics and Engineering*, 193:1437–1453, 2004.

[27] Y. Cheny and O. Botella. The LS-STAG method: A new immersed boundary/level-set method for the computation of incompressible viscous flows in complex moving geometries with good conservation properties. *Journal of Computational Physics*, 2008. Submitted.

[28] R. Codina. A discontinuity-capturing crosswind-dissipation for the finite element solution of the convection-diffusion equation. *CMAME*, 110:325–342, 1993.

[29] R. Codina. Comparison of some finite element methods for solving the diffusion-convection-reaction equation. *Computer Methods in Applied Mechanics and Engineering*, 156:185–210, 1998.

[30] R. Codina. Stabilization of incompressibility and convection through orthogonal subscales in finite element methods. *Computer Methods in Applied Mechanics and Engineering*, 190:1579–1599, 2000.

[31] R. Codina. A stabilized finite element method for generalized stationary incompressible flows. *Computer Methods in Applied Mechanics and Engineering*, 190:2681–2706, 2001.

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

[33] R. Codina. Analysis of a stabilized finite element approximation of the Oseen equations using orthogonal subscales. *Applied Numerical Mathematics*, 58:264–283, 2008.

[34] R. Codina. Finite element approximation of the three field formulation of the Stokes problem using arbitrary interpolations. *SIAM journal of Numerical Analysis*, 47:699–718, 2009.

[35] R. Codina and G. Houzeaux. Implementation aspects of coupled problems in CFD involving time dependent domains, in *Verification and Validation Methods for Challenging Multiphysics Problems*, G. Bugeda, J.C Courty, A. Guilliot, R. Höld, M. Marini, T. Nguyen, K. Papailiou, J. Périaux and D. Schwamborn (Eds.), pages 99–123. CIMNE, Barcelona, 2006.

[36] R. Codina, J. Principe, O. Guasch, and S. Badia. Time dependent subscales in the stabilized finite element approximation of incompressible flow problems. *Computer Methods in Applied Mechanics and Engineering*, 196:2413–2430, 2007.

[37] R. Codina and O. Soto. A numerical model to track two-fluid interfaces based on a stabilized finite element method and the level set technique. *International Journal for Numerical Methods in Fluids*, 40:293–301, 2002.

[38] R. Codina and O.C. Zienkiewicz. CBS versus GLS stabilization of the incompressible Navier-Stokes equations and the role of the time step as stabilization parameter. *Communications in Numerical Methods in Engineering*, 18:99–112, 2002.

[39] C.E. Colosqui and A.A. Oberai. Generalized smagorinsky model in physical space. *Computers and Fluids*, 37(3):207–217, 2008.

[40] H. Coppola-Owen and R. Codina. Improving Eulerian two-phase flow finite element approximation with discontinuous gradient pressure shape functions. *International Journal for Numerical Methods in Fluids*, 49:1287–1304, 2005.

[41] H. Coppola-Owen and R. Codina. A finite element model for free surface flows on fixed meshes. *International Journal for Numerical Methods in Fluids*, 54:1151–1171, 2007.

[42] James W. Demmel, Stanley C. Eisenstat, John R. Gilbert, Xiaoye S. Li, and Joseph W. H. Liu. A supernodal approach to sparse partial pivoting. *SIAM J. Matrix Analysis and Applications*, 20(3):720–755, 1999.

[43] M. Do-Quang and G. Amberg. The splash of a solid sphere impacting on a liquid surface: numerical simulation of the influence of wetting. *Phys. Fluids*, 21, 2009.

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

[45] J. Donea, P. Fasoli-Stella, and S. Giuliani. Lagrangian and eulerian finite element techniques for transient fluid structure interaction problems. In *Transactions Fourth SMIRT*, page B1/2, 1977.

[46] I.S. Duff, A.M. Erisman, and J.K Reid. *Direct Mehtods for Sparse Matrices*. Clarendon, Oxford, 1986.

[47] T. Dunne and R. Rannacher. Adaptive finite elementt approximation of Fluid-Structure Interaction based on an Eulerian variational formulation.

[48] A. Ern and J.-L. Guermond. *Theory and Practice of Finite Element*. Springer-Verlag, 2004.

[49] M. A. Fernandez, J-F. Gerbeau, and C. Grandmont. A projection semi-implicit scheme for the coupling of an elastic structure with an incompressible fluid. *International Journal for Numerical Methods in Engineering*, 69:794–821, 2007.

[50] S. Fernández-Méndez and A. Huerta. Imposing essential boundary conditions in mesh-free methods. *Computer Methods in Applied Mechanics and Engineering*, 193(12-14):1257–1275, 2004.

[51] R. Finkel and J.L. Bentley. Quad trees: A data structure for retrieval on composite keys. *Acta Informatica*, 4(1):1–9, 1974.

[52] L. Formaggia and F. Nobile. A stability analysis for the Arbitrary Lagrangian Eulerian formulation with finite elements. *East-West J. Num. Math.*, 7:105–132, 1999.

[53] L. Formaggia and F. Nobile. Stability analysis of second-order time accurate schemes for ALE-FEM. *Computer Methods in Applied Mechanics and Engineering*, 193:4097–4116, 2004.

[54] C. Forster, W.A. Wall, and E. Ramm. Artificial added mass instabilities in sequential staggered coupling of nonlinear structures and incompressible viscous flows. *Computer Methods in Applied Mechanics and Engineering*, 196:1278–1293, 2007.

[55] L.P. Franca and C. Farhat. Bubble functions prompt unusual stabilized finite element methods. *Computer Methods in Applied Mechanics and Engineering*, 123:299–308, 1994.

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

[57] A. George and J.W. Liu. *Computer Solution of Large Sparse Positive Definite Systems*. Prentice-Hall, Englewood Cliffs, 1981.

[58] A. Gerstenberger and W.A. Wall. An embedded Dirichlet formulation for 3D continua. *International Journal for Numerical Methods in Engineering*, 82:537–563, 2010.

[59] S. Ghosh and N. Kikuchi. An arbitrary Lagrangian-Eulerian finite element method for large deformation analysis of elastic-viscoplastic solids. *Computer Methods in Applied Mechanics and Engineering*, 86:127–188, 1991.

[60] A. Gilmanov and F. Sotiropoulos. A hybrid Cartesian/immersed boundary method for simulating flows with 3D, geometrically complex, moving bodies. *Journal of Computational Physics*, 207:457–492, 2005.

[61] R. Glowinski. *Finite element methods for Incompressible Viscous Flows*, in Numerical Methods for Fluids (Part 3), Handbook of Numerical Analysis, 9. North-Holland, Elsevier, 2003.

[62] R. Glowinski, T.-W. Pan, and J. Périaux. A fictitious domain method for Dirichlet problems and applications. *Computer Methods in Applied Mechanics and Engineering*, 111:203–303, 1994.

[63] R. Glowinski, T.W. Pan, T.I. Hesla, D.D. Joseph, and J. Périaux. A distributed Lagrange multiplier/fictitious domain method for flows around moving rigid bodies: application to particulate flow. *International Journal for Numerical Methods in Fluids*, 30:1043–1066, 1999.

[64] A. Hansbo and P. Hansbo. An unfitted finite element method, based on Nitsche's method, for elliptic interface problems. *Computer Methods in Applied Mechanics and Engineering*, 191:5537–5552, 2002.

[65] P. Hansbo and M.G. Larson. Discontinuous Galerkin methods for incompressible and nearly incompressible elasticity by Nitsche's method. *Computer Methods in Applied Mechanics and Engineering*, 191:1895–1908, 2002.

[66] D. Hartmann, M. Meinke, and W. Schröder. Differential equation based constrained reinitialization for level set methods. *Journal of Computational Physics*, 227:6821–6845, 2007.

[67] M. Heath, E. Ng, and B. Peyton. Parallel algorithms for sparse linear systems. *SIAM Review*, 33:420–460, 1991.

[68] M.R. Hestenes and E.L. Stiefel. Methods of conjugate gradientes for solving linear systems. *Journal of Research of the National Bureau of Standards*, Section B(49):409–436, 1952.

[69] C.W. Hirt and B.D. Nichols. Volume of fluid (VOF) method for the dynamics of free boundaries. *Journal of Computational Physics*, 39:201–225, 1981.

[70] J. Dolbow H.M. Mourad and I. Harari. A bubble-stabilized finite element method for Dirichlet constraints on embedded interfaces. *International Journal for Numerical Methods in Engineering*, 69:772–793, 2007.

[71] G. Houzeaux and R. Codina. Transmission conditions with constraints in finite element domain decomposition methods for flow problems. *Communications in Numerical Methods in Engineering*, 17:179–190, 2001.

[72] G. Houzeaux and R. Codina. A Chimera method based on a Dirichlet/Neumann (Robin) coupling for the Navier-Stokes equations. *Computer Methods in Applied Mechanics and Engineering*, 192:3343–3377, 2003.

[73] G. Houzeaux and R. Codina. A finite element model for the simulation of lost foam casting. *International Journal for Numerical Methods in Fluids*, 46:203–226, 2004.

[74] A. Huerta and W.K. Liu. Viscous flow with large free surface motion. *Computer Methods in Applied Mechanics and Engineering*, 69:277–324, 1988.

[75] J. Huétink. On the simulation of thermo-mechanical forming processes. *Dissertation*, 1986.

[76] T.J.R. Hughes. Multiscale phenomena: Green's function, the Dirichlet-to-Neumann formulation, subgrid scale models, bubbles and the origins of stabilized formulations. *Computer Methods in Applied Mechanics and Engineering*, 127:387–401, 1995.

[77] T.J.R. Hughes, G.R. Feijóo, L. Mazzei, and J.B. Quincy. The variational multiscale method–a paradigm for computational mechanics. *Computer Methods in Applied Mechanics and Engineering*, 166:3–24, 1998.

[78] T.J.R. Hughes, L.P. Franca, and G.M. Hulbert. A new finite element formulation for computational fluid dynamics: VII. The Stokes problem with various well-posed boundary conditions: symmetric formulations that converge for all velocity/pressure spaces. *Computer Methods in Applied Mechanics and Engineering*, 65:85–96, 1987.

[79] T.J.R. Hughes, W.K. Liu, and T.K. Zimmerman. Lagrangian-eulerian finite element formulation for incompressible viscous flows. *Computer Methods in Applied Mechanics and Engineering*, 29:329–349, 1981.

[80] T.J.R. Hughes, A. Masud, and J. Wan. A stabilized mixed discontinuous Galerkin method for Darcy flow. *Computer Methods in Applied Mechanics and Engineering*, 195:3347–3381, 2006.

[81] S. R. Idelsohn, F. Del Pin, R. Rossi, and E. Oñate. Fluid-structure interaction problems with strong added-mass effect fluid incompressibility. *International Journal for Numerical Methods in Engineering*, 80:1261–1294, 2009.

[82] H. Ji and J.E. Dolbow. On strategies for enforcing interfacial constraints and evaluating jump conditions with the extended finite element method. *International Journal for Numerical Methods in Engineering*, 61:2508–2535, 2004.

[83] M. Juntunen and R. Stenberg. Nitsche's method for general boundary conditions. Helsinki University of Technology, Institute of Mathematics, Research Reports A530, 2007.

[84] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *International Conference on Parallel Processing*, pages 113–122, 1995.

[85] N. Kechkar and D. Silvester. Analysis of locally stabilized mixed finite element methods for the Stokes problem. *Mathematics of Computation*, 58:1–10, 1992.

[86] D. Kima and H. Choi. Immersed boundary method for flow around an arbitrarily moving body. *J. Comput. Phys.*, 212:662–680, 2006.

[87] Ming-Chih Lai and C.S. Peskin. An immersed boundary method with formal second-order accuracy and reduced numerical viscosity. *Journal of Computational Physics*, 160:705–719, 2000.

[88] C. Lanczos. Solution of systems of linear equations by minimized iterations. *Journal of Research of the National Bureau of Standards*, (49):33–53, 1952.

[89] A. Legay, J. Chessa, and T. Belytschko. An Eulerian-Lagrangian method for fluid-structure interaction based on level sets. *Computer Methods in Applied Mechanics and Engineering*, 195:2070–2087, 2006.

[90] M. Lesoinne and C. Farhat. Geometric conservation laws for flow problems with moving boundaries and deformable meshes, and their impact on aerolastic computations. *Computer Methods in Applied Mechanics and Engineering*, 134:71–90, 1996.

[91] R.J. LeVeque and Z. Li. The immersed interface method for elliptic equations with discontinuous coefficients and singular sources. *SIAM Journal on Numerical Analysis*, 31 (4):1019–1044, 1994.

[92] R.J. LeVeque and Z. Li. Immersed interface method for incompressible Navier-Stokes equations. *SIAM Journal on Scientific and Statistical Computing*, 18 (3):709–735, 1997.

[93] A.J. Lew and G.C. Buscaglia. A discontinuous-Galerkin-based immersed boundary method. Submitted.

[94] J. Li, M. Hesse, J. Ziegler, and A. Woods. An arbitrary Lagrangian-Eulerian method for moving-boundary problems and its application to jumping over water. *Journal of Computational Physics*, 208:289, 2005.

[95] N. Li, Y. Saad, and E. Chow. Crout versions of ILU for general sparse matrices. *SIAM J. Sci. Comput.*, 25(2):716–728, 2004.

[96] N. Li, Y. Saad, and M. Sosonkina. pARMS: a parallel version of the algebraic recursive multilevel solver. *Technical report UMSI-2001-100 (Minnesota Supercomputer Institute, Univ. Minnesota, Minneapolis*, 2001.

[97] X. Li. Direct solvers for sparse matrices. 2010.

[98] P. Lin. A fixed-grid model for simulation of a moving body in free surface flows. *Computers and Fluids*, 36:549–561, 2007.

[99] W.K. Liu, T. Belytschko, and H. Chang. An arbitrary Lagrangian-Eulerian finite element method for path-dependent materials. *Computer Methods in Applied Mechanics and Engineering*, 58:227–245, 1986.

[100] R. Löhner, J.R. Cebral, F.F. Camelli, J.D. Baum, and E.L. Mestreau. Adaptive embedded/immersed unstructured grid techniques. *Archives of Computational Methods in Engineering*, 14:279–301, 2007.

[101] S. Maclachlan and Y. Saad. A greedy strategy for coarse-grid selection. *SIAM J. Sci. Comput.*, 29(5):1825–1853, 2007.

[102] A. Masud and T.J.R. Hughes. A stabilized mixed finite element method for Darcy flow. *Computer Methods in Applied Mechanics and Engineering*, 191:4341–4370, 2002.

[103] R. Mittal and G. Iaccarino. Immersed boundary methods. *Annual Review of Fluid Mechanics*, 37:239–261, 2005.

[104] J. Mohd-Yusof. Combined immersed boundaries/B-splines methods for simulations of flows in complex geometries. *CTR annual research briefs, Stanford University, NASA Ames*, 1997.

[105] T. Möller. A fast triangle-triangle intersection test. 1997.

[106] K.B. Nakshatrala, D.Z. Turner, K.D. Hjelmstad, and A. Masud. A stabilized mixed finite element method for Darcy flow based on a multiscale decomposition of the solution. *Computer Methods in Applied Mechanics and Engineering*, 195:4036–4049, 2006.

[107] N.M. Newmark. A method of computation for structural dynamics. *Journal of Engineering Mechanics Division, ASCE.*, 85:67–94, 1959.

[108] J. Nitsche. Über ein Varationsprinzip zu Lösung von Dirichlet-Problemen bei Verwendung von Teilräumen, die keinen Randbedingungen unterworfen sind. *Abhandlungen aus dem Mathematisches Seminar der Universität*, 36:9–15, 1971.

[109] F. Nobile. *Numerical Approximation of Fluid-Structure Interaction problems with application to Haemodynamics*. PhD thesis, École Polytechnique Fédérale de Lausanne, 2001.

[110] Y. Notay. Algebraic multigrid and algebraic multilevel methods: a theoretical comparison. *Numer. Linear Algebra Appl.*, 12:419–451, 2005.

[111] S. Okazawa, K. Kashiyama, and Y. Kaneko. Eulerian formulation using stabilized finite element method for large dormation solid dynamics. *International Journal for Numerical Methods in Engineering*, 72:1544–1559, 2007.

[112] S. Osher and R.P. Fedkiw. Level set methods: and overview and some recent results. *Journal of Computational Physics*, 169:463–502, 2001.

[113] C.S. Peskin. Flow patterns around heart valves: A numerical method. *Journal of Computational Physics*, 10:252–271, 1972.

[114] S.B. Pope. *Turbulent Flows*. Cambridge University Press, 2000.

[115] A. Quarteroni and A. Valli. *Domain Decomposition Methods for Partial Differential Equations*. Oxford Science Publications, 1999.

[116] G. Rapin and G. Lube. A stabilized three-field formulation for advection-diffusion equations. *Computing*, 73:155–178, 2004.

[117] A. Rodriguez-Ferran, F. Casadei, and A. Huerta. ALE stress update for transient and quasiestatic processes. *International Journal for Numerical Methods in Engineering*, 43:241–262, 1998.

[118] A. Russo. Bubble stabilization of finite element methods for the linearized incompressible Navier-Stokes equations. *Computer Methods in Applied Mechanics and Engineering*, 132:335–343, 1996.

[119] P. Durbin S. Majumdar, G. Iaccarino. RANS solvers with adaptive structured boundary non-conforming grids. *Annual Research Briefs, NASA Ames Research Center/Stanford University Center for Turbulence Research, Stanford, CA, 2001, pp.353-366*.

[120] Y. Saad. ILUT: a dual threshold incomplete ILU factorization. *Numer. Linear Algebra Appl.*, 1:387–402, 1994.

[121] Y. Saad. *Iterative methods for sparse linear systems*. Society for Industrial Mathematics, 2000.

[122] Y. Saad and B. Suchomel. ARMS: an algebraic recursive multilevel solver for general sparse linear systems. *Numer. Linear Algebra Appl.*, 9:359, 2002.

[123] P. Sagaut. *Large Eddy Simulation for Incompressible Flows*. Scientific Computing, Springer, 2001.

[124] O. Schenk, M. Bollhoefer, and R. Roemer. On large-scale diagonalization techniques for the anderson model of localization. *SIAM Review*, 50:91–112, 2008.

[125] P.J. Silvester and N. Kechkar. Stabilised bilinear-constant velocity-pressure finite elements for the conjugate gradient solution of the Stokes problem. *CMAME*, 79:71–86, 1990.

[126] I. Babuška. Error bounds for finite element method. *Numerische Mathematik*, 16:322–333, 1971.

[127] T.E. Tezduyar. Finite element methods for flow problems with moving boundaries and interfaces. *Archives of Computational Methods in Engineering*, 8(2):83–130, 2001.

[128] A. Toselli and O. Windlun. *Domain decomposition methods—Algorithms and theory*. Springer, 2005.

[129] S. Turek and J. Hron. Proposal for numerical benchmarking of fluid–structure interaction between an elastic object and laminar incompressible flow. In H. J. Bungartz and M. Schäfer, editors, *Fluid–Structure Interaction – Modelling, Simulation, Optimization*, number 53 in Lecture Notes in Computational Science and Engineering, pages 371–385. Springer, Berlin, 2006. ISBN 3-540-34595-7.

[130] A. van der Ploeg, E.F.F. Botta, and F.W. Wubs. Nested grids ILU decomposition (NGILU). *J. Comput. Appl. Math.*, 66:515, 1996.

[131] H.A. van der Vorst. Bicgstab: a fast smoothly converging variant of bi-cg for the solution of non-symmetric linear systems. *Siam Journal on Scienteific and Statistical Computing*, 12:631–644, 1992.

[132] R.S. Varga. *Matrix iterative analysis*. Prentice-Hall, Englewood Cliffs, 1962.

[133] J.V. Voorde, J. Vierendeels, and E. Dick. Flow simulations in rotary volumetric pumps and compressors with the fictitious domain method. *Journal of Computational and Applied Mathematics*, 168:491–499, 2004.

[134] H.F. Walker. Implementation of the fmres method using householder transformations. *Siam Journal of Scienteific Computing*, 9:152–163, 1988.

[135] W.A. Wall and E. Ramm. Fluid-structure interaction based upon a stabilized (ALE) finite element method. In E. Oñate and S.R. Idelsohn, editors, *Proceedings of the Fourth World Congress on Computational Mechanics (WCCM IV)*, June 29 - July 2, 1998, Buenos Aires, Argentina, pages 1988–1995. CIMNE, Barcelona.

[136] C.Z. Wang and G.X. Wu. Analysis of second-order resonance in wave interactions with floating bodies through a finite-element method. *Ocean Engineering*, 35:717–726, 2008.

[137] S. Xu and Z.J. Wang. An immersed interface method for simulating the interaction of a fluid with moving boundaries. *Journal of Computational Physics*, 216:454–493, 2006.

[138] S. Yan and Q.W. Ma. Numerical simulation of fully nonlinear interaction between steep waves and 2D floating bodies using the QALE-FEM method. *Journal of Computational Physics*, 221:666–692, 2007.

[139] J.H. Ferziger Y.H. Tseng. A ghost-cell immersed boundary method for flow in complex geometry. *Journal of Computational Physics*, 192:593–623, 2003.

[140] D.M. Young. *Iterative solution of large linear systems*. Academic Press, New York, 1971.

[141] H. Zhao, J.B. Freund, and R.D. Moser. A fixed-mesh method for incompressible flow-structure systems with finite solid deformations. *J. Comput. Phys.*, 227:3114–3140, 2008.