Spectral Element Approximation of the Incompressible Navier-Stokes Equations in a Moving Domain and Applications

THÈSE N^{O} 4529 (2009)

PRÉSENTÉE LE 27 NOVEMBRE 2009

À LA FACULTÉ SCIENCES DE BASE

CHAIRE DE MODÉLISATION ET CALCUL SCIENTIFIQUE

PROGRAMME DOCTORAL EN MATHÉMATIQUES

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Gonçalo PENA

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Suisse
2009

To my parents

Abstract

In this thesis we address the numerical approximation of the incompressible Navier-Stokes equations evolving in a moving domain with the spectral element method and high order time integrators.

First, we present the spectral element method and the basic tools to perform spectral discretizations of the Galerkin or Galerkin with Numerical Integration (G-NI) type. We cover a large range of possibilities regarding the reference elements, basis functions, interpolation points and quadrature points. In this approach, the integration and differentiation of the polynomial functions is done numerically through the help of suitable point sets. Regarding the differentiation, we present a detailed numerical study of which points should be used to attain better stability (among the choices we present).

Second, we introduce the incompressible steady/unsteady Stokes and Navier-Stokes equations and their spectral approximation. In the unsteady case, we introduce a combination of Backward Differentiation Formulas and an extrapolation formula of the same order for the time integration. Once the equations are discretized, a linear system must be solved to obtain the approximate solution. In this context, we consider the solution of the whole system of equations combined with a block type preconditioner. The preconditioner is shown to be optimal in terms of number of iterations used by the GMRES method in the steady case, but not in the unsteady one. Another alternative presented is to use algebraic factorization methods of the Yosida type and decouple the calculation of velocity and pressure. A benchmark is also presented to access the numerical convergence properties of this type of methods in our context.

Third, we extend the algorithms developed in the fixed domain case to the Arbitrary Lagrangian Eulerian framework. The issue of defining a high order ALE map is addressed. This allows to construct a computational domain that is described with curved elements. A benchmark using a direct method to solve the linear system or the Yosida-q methods is presented to show the convergence orders of the method proposed.

Finally, we apply the developed method with an implicit fully coupled and semi-implicit approach, to solve a fluid-structure interaction problem for a simple 2D hemodynamics example.

Keywords: spectral element method, incompressible Navier-Stokes equations, preconditioning, algebraic factorization method, fluid-structure interaction, hemodynamics

Resumé

Dans cette thèse nous nous intéressons à l'approximation numérique des équations incompressibles de Navier-Stokes évoluant dans un domaine en mouvement par la méthode des éléments spectraux et des intégrateurs en temps d'ordre élevé.

Dans une première phase, nous présentons la méthode des éléments spectraux et les outils de base pour effectuer des discrétisations spectrales du type Galerkin ou Galerkin avec intégration numérique (G-NI). Nous couvrons un large éventail de possibilités concernant les éléments de reference, fonctions de base, points d'interpolation et points de quadrature. Dans cette approche, l'intégration et la différentiation des fonctions polynomiales est faite numériquement grâce à l'aide d'ensembles de points convenables. En ce qui concerne la différenciation, nous présentons une étude numérique des points qui doivent être utilisés pour atteindre une meilleure stabilité numérique (parmi les choix que nous avons actuellement).

Deuxièmement, nous introduisons les équations incompressibles stationnaires et non-stationnaires de Stokes et de Navier-Stokes et son approximation spectrale. Dans le cas non-stationnaire, nous introduisons une combinaison de la méthode *Backward Differentiation Formula* (BDF) et une formule d'extrapolation du même ordre pour l'intégration par rapport au temps. Une fois les équations discrétisées, un système linéaire doit être résolu pour obtenir la solution approchée. Dans ce contexte, nous résolvons ce système avec un préconditionneur par blocs. Nous montrons que le préconditionneur est optimal par rapport au nombre d'itérations utilisées par la méthode GMRES dans le cas stationnaire, mais pas dans le cas non-stationnaire. Une autre alternative est d'utiliser les méthodes de factorization algebrique de type Yosida et separer le calcul de la vitesse et de la pression. Un cas test est présenté pour determiner les proprietés de convergence de ce type de méthodes dans notre contexte.

Troisièmement, nous étendons les algorithmes développés dans le cas où le domaine est fixé au cadre de la formulation Arbitraire Lagrange-Euler (ALE). La question de la définition d'une carte ALE d'ordre élevé est abordée. Cela permet de construire un domaine de calcul qui est décrit avec des éléments courbes. Un cas test utilisant une méthode directe et les méthodes Yosida-q pour résoudre le système linéaire est présenté pour montrer les ordres de convergence de la méthode proposée.

Finalement, nous appliquons la méthode développée pour résoudre une un problème d'interaction fluide-structure pour un exemple simple bidimensionnel d'hémodynamique. Nous considérons deux approches: une implicite entièrement couplée et une semi-implicite.

Mots clés: méthode des éléments spectraux, équations incompressibles Navier-Stokes, preconditionnement, méthode de factorisation algébrique, interaction fluide-structure, hémodynamique

Acknowledgements

A doctoral thesis is a project that demands for endurance, commitment and patience. Through the past four years that I have been a PhD student at EPF Lausanne, I encountered people that helped me with the emotional and scientific load that comes along with a work of this kind. It is this text's purpose to acknowledge the people that helped me, in their way, to achieve my doctor's degree.

My first word goes to Professor Alfio Quarteroni. He kindly accepted me as a PhD student and gave me the unique opportunity to work with him and his group in EPF Lausanne. The knowledge I gained while working in his group go beyond the doctoral work.

A second word goes to Professor Christophe Prud'homme. The constant presence during the whole doctoral work, as a tutor and as a friend, made me come this far. Always available, always patient... My gratitude and admiration cannot be expressed in words. It was and is a pleasure to work with him.

I would like to thank Professor Bernard Dacorogna for having accepted to be the president of the jury of this thesis. Also, I thank Professor Paula Oliveira, Professor Michel Deville and Professor Christophe Prud'homme for accepting the invitation to be members of the jury.

I thank also all my collegues in CMCS. The moments (some only available in italian in my head) are among my best memories of the time I spent in Lausanne. In particular, I must mention Simone Deparis that was always available to answer my questions on fluid-structure problems or even system administrator chores.

Apart from everyone I have already mentioned, I want to thank several other people. First, the two guys I lived with during three years, Beni and Fabienne, who supported and resisted a moody portuguese. You are in my heart forever. Annalisa, my dear dear friend. I can only say that true friendship will always unite us, even across the Atlantic. To all my friends in Lausanne, thank you for putting up with me this whole time. Besides the people around me in Switzerland, also my friends in Portugal were a great support in the most difficult hours. Ana, Tânia, Rita, Jorge, thank you for the support.

To my significant other, thank you for supporting and loving me.

One last word of appreciation goes to the institutions that were my financial support during these four years: the *Fundação para a Ciência e Tecnologia* and the Department of Mathematics of the University of Coimbra. A special thanks goes to EPFL for the work conditions and logistic support.

Contents

Nomenclature

Polynomial bases setting:

$\hat{\Omega}$	reference domain
Ω	domain
$\partial\Omega$	boundary of Ω
d	topological dimension of $\hat{\Omega}$
$\mathcal{T}^d$	reference d-simplex
$\mathcal{Q}^d$	reference simplex-product in $\mathbb{R}^d$
L_N	N-th Legendre polynomial
$P_N^{(\alpha,\beta)}$	Jacobi polynomial of degree N and indices (α, β)
$\mathbb{P}_N(-1,1)$	set of polynomials of order N defined in $(-1,1)$
$\mathbb{P}_N(\mathcal{T}^d)$	set of polynomials of total degree smaller or equal to N defined in $\mathcal{T}^d$
$\mathbb{Q}_N(\mathcal{Q}^d)$	set of polynomials of degree smaller or equal to N defined in $\mathcal{Q}^d$
ℓ_i	i-th Lagrange basis function
$\mathcal{B}$	set of basis functions
J_f	Jacobian of application f
$\binom{n}{k}$	number of possible combinations of k objects chosen among a set of n elements
T_N	dimension of $\mathbb{P}_N(\mathcal{T}^d)$ or $\mathbb{Q}_N(\mathcal{Q}^d)$
Λ_{T_N}	Lebesgue constant
$V(\mathcal{B}, X)$	generalized Vandermonde matrix associated with basis $\mathcal{B}$ and point set X
$\mathcal{P}$	polynomial set
$\mathcal{D}_r^n$	set of the n-th derivatives of the polynomials in $\mathcal{P}$ in the r direction
$C_{\mathcal{P}}$	representation matrix of the polynomials in $\mathcal{P}$ in the underlying basis
$\boldsymbol{\varphi}$	geometrical transformation
N_{geo}	polynomial order of the geometrical transformation
$\mathcal{T}_{h,N_{\text{geo}}}$	triangulation of the domain Ω using mesh size h and geometrical transformations of order N_{geo}

Functional setting:

$L^p(\Omega)$	space of p-integrable functions on Ω
$L^p_0(\Omega)$	subspace of $L^p(\Omega)$ of zero mean functions, ie, $\int_\Omega v dx = 0$
$W^{m,p}(\Omega)$	space of functions v in $L^p(\Omega)$ with all partial derivatives up to order m in $L^p(\Omega)$
$H^m(\Omega)$	$W^{m,2}(\Omega)$
$\|\cdot\|_{H^m(\Omega)}$	standard norm for the space $H^m(\Omega)$
$\lvert\cdot\rvert_{H^m(\Omega)}$	standard semi-norm for the space $H^m(\Omega)$

Linear algebra setting:

$\mathcal{K}(A)$	iterative condition number of matrix A
$\lambda_{\min}$	minimum eigenvalue of matrix A
$\lambda_{\max}$	maximum eigenvalue of matrix A
G_N	matrix associated with the discretization of the gradient operator
D_N	matrix associated with the discretization of the divergence operator
H_N	matrix associated with the discretization of the Laplacian operator
M_p	mass matrix for the pressure operator
A_p	discrete Laplacian matrix for the pressure operator

Introduction

The accurate approximation of the incompressible Navier-Stokes equations evolving in a moving domain is an important subject of research in applied mathematics. This type of problem appears in many important fluid dynamics applications, including fluid-structure interaction problems [19, 67, 31] or free surface flows [45, 4]. The main difficulties of simulating this problem are:

(i) how to discretize the system of equations in a domain that evolves in time, see [47, 22]

(ii) the choice of the solution algorithm for the coupled problem, see [19, 29, 42, 31]

(iii) the techniques to solve the linear system that appear in the process of calculating the solution.

The full problem that we propose to solve reads as

$$\begin{aligned}
\rho \left.\frac{\partial \mathbf{u}}{\partial t}\right|_{\mathbf{Y}} + \rho((\mathbf{u}-\mathbf{w})\cdot\boldsymbol{\nabla}_{\mathbf{x}})\mathbf{u} - 2\nu \mathbf{D}_{\mathbf{x}}(\mathbf{u}) + \nabla p &= \mathbf{f}, \quad \text{in } \Omega_t \times I \\
\mathrm{div}_{\mathbf{x}}(\mathbf{u}) &= 0, \quad \text{in } \Omega_t \times I \\
\rho_w h \frac{\partial^2 \eta}{\partial t^2} - kGh\frac{\partial^2 \eta}{\partial z^2} + \frac{Eh}{1-\mu^2}\frac{\eta}{R_0^2} - \gamma_v \frac{\partial^3 \eta}{\partial z^2 \partial t} &= \Phi_r \quad \text{in } \Gamma_0^w \\
\mathbf{u}\circ\boldsymbol{\varphi}_\eta &= \dot{\eta}\mathbf{e}_r, \quad \text{on } \Gamma_0^w \\
-((\mathbf{T}\mathbf{n})\cdot\mathbf{e}_r)\circ\boldsymbol{\varphi}_\eta &= \Phi_r
\end{aligned}$$

where $\mathbf{u}$ is the velocity of the fluid, p the pressure field associated and η the displacement associated with the arterial wall (with respect to some reference configuration). The incompressible Navier-Stokes equations are coupled with the generalized string model through the interface Γ_t^w (see Figure 1 for a schematic representation of the domains involved). This model and all the notation used are derived and explained in chapter 5.

The goal of this thesis is to develop a numerical method to solve the incompressible Navier-Stokes equations defined in a moving domain that combines a spectral space discretization, a multistep Backward Differentiation formula time discretization, a Yosida type algebraic factorization scheme and apply it to a problem with blood flowing through an artery.

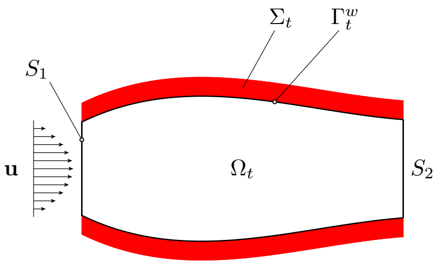


Figure 1: 2D simplified compliant tube.

The Spectral Element Method (SEM)

Spectral methods are a class of spatial discretization techniques for approximating the solution of differential equations within a space of high degree polynomials through the method of weighted residuals. Typically, the spectral approximation of the solution is expanded as a linear combination of certain *basis functions* that span this space. These basis functions, which can be either of nodal or modal type, depending on the domain of the problem, are often the Tchebychev, Legendre, Lagrange or Dubiner polynomials.

Classical (single domain) spectral methods can be categorized in three types: Galerkin (pure or with numerical integration), collocation and tau. The difference between all these methods resides in the choice of the test functions, that is, the set of functions used to insure that the differential equation and/or boundary conditions are enforced to the approximation. The Galerkin method uses the same trial/test functions that individually satisfy the boundary conditions. The collocation method uses as test functions discrete Dirac delta functions centered at, *a priori*, chosen points, and enforces the differential equation exactly at these points. The tau version of this method is similar to the Galerkin method, except that the test functions do not need to satisfy the boundary conditions. These are enforced with a separate set of equations.

The first comprehensive theoretical formulation and analysis of spectral methods was performed by Orzag and Gottlieb [38]. At this point, the spectral method was applied to very simple geometries (basically 1D), defined in one single domain, see Canuto, Hussaini, Quarteroni and Zang [10] for a complete analysis. The development that the SEM went through in the 80's gave rise to the the modern Galerkin (with or without numerical integration) spectral method. The work by Patera in [61] provided the bases for the modern *multidomain spectral method*. This version of the SEM pushes the method to deal with arbitrary geometries, combining its spectral properties with the flexibility of the finite element method. Although in the beginning it was only applied to geometries that were partitioned into quadrangular sub-domains, the monography by Sherwin and Karniadakis [48] provided a further extension of this method to geometries that could be partitioned into simplices, thus giving even more flexibility to the SEM.

The incompressible Navier-Stokes equations

The starting point in discretizing in space the Navier-Stokes equations (in a fixed domain) in the primitive variable formulation is the choice of discrete spaces for velocity and pressure. It is a well known fact that these spaces cannot be chosen independently. This discrete compatibility condition enforces that a certain gap must exist between these spaces. If such a condition is violated, then the linear system associated with the discretization fails to have a unique solution. This is the Brezzi-Babuska *inf-sup condition*, see Quarteroni and Valli [70]. In the literature one can find a few possible choices of spaces that fulfill such condition. Some examples are Bernardi and Maday [2], Schwab and Suri [76], Ainsworth and Coggins [1] and Stenberg and Suri [81]. It is known that, for instance, choosing velocities as polynomials of degree N and pressures as piecewise discontinuous polynomials of degree N or $N-1$ violates the inf-sup condition, see Bernardi and Maday [2]. At an algebraic level, this violation is reflected in the existence of non-constant pressures (defined all over the domain) whose discrete gradient is zero. Such pressures lead to the non uniqueness of the solution of the Stokes/Navier-Stokes equations. One of the most popular and widely used discretizations that does not lead to the presence of spurious pressure modes was studied by Patera, Maday [2] and Rønquist [72]. In their work, they approximate the velocities with polynomials of degree N and pressures with piecewise discontinuous polynomials of degree $N-2$. This introduces a gap sufficiently large between the discrete spaces that allows for the fulfillment of the compatibility condition. However, the error estimates proven are not optimal regarding the polynomial order of the approximation spaces. This is due to the fact that the inf-sup constant decreases as the polynomial order increases. Schwab and Suri investigated in [76] the $\mathbb{P}_N - \mathbb{P}_{N-2}$ and $\mathbb{P}_N - \mathbb{P}_{N-2}^{disc}$ methods. They showed that the inf-sup constant can be bounded from below by CN^{-3}. We remark that other spaces can be found in the *hp*-FEM literature but they all suffer either from spurious pressure modes (and thus stabilizing or filtering should be applied) or have inf-sup constants that decrease to zero with increasing polynomial order.

In the context of the finite/spectral element method, apart from the space discretization issue, several solution strategies have been proposed to solve the unsteady Navier-Stokes equations. We highlight two of them: (i) *coupled methods* and (ii) *splitting methods.* Splitting methods attempt to decouple the calculation of the velocity and pressure field, either by performing a splitting in the differential equations, see for instance [39, 40, 41], or by doing so at the algebraic level, see [68, 69, 75, 34, 33]. Such decoupling of the variables makes the calculation of the solution faster, but at the cost of introducing some error in the approximation, called *splitting error.* The differential type of splitting also introduces an artificial boundary condition (that needs to be derived) for the pressure operator. On the other hand, algebraic splitting methods do not have this requirement. Coupled algorithms do not introduce splitting error. Instead, they try to solve the fully coupled velocity-pressure system of equations, either with an Uzawa approach or with a suitable preconditioner for the whole linear system, see [50, 51, 78].

The Arbitrary Lagrangian Eulerian framework

The numerical simulation of differential equations evolving in a moving domain adds an extra layer of complexity to the numerical methods that are devised to solve such problem. A common technique to keep track of the evolution of the domain is the Arbitrary Lagrangian Eulerian (ALE) framework [47, 22]. This framework introduces a quantity that measures the domain's velocity of deformation. Its numerical approximation has been discussed in the context of the spectral element method, Ho and Rønquist [45] and Bouffanais [4], or the finite element method Nobile [57]. A relevant aspect in devising numerical schemes in the ALE framework is the so called *Geometric Conservation Law* (GCL). A numerical scheme satisfies the GCL if it can represent a constant solution through time. Although it is not a necessary or sufficient condition for convergence/stability of the schemes, in some cases, the satisfaction of the GCL implies stability independently of the domain's rate of deformation, see Nobile [57].

Life and main contributions

The starting point of this thesis is undoubtedly the spectral (element) method and it's implementation. All the simulations and algorithms presented in this thesis are coded in a library called *Life*. *Life* is a versatile library allowing for 1D, 2D and 3D partial differential solves using h/p type Galerkin methods, continuous as well as discontinuous, in sequential and parallel settings. The reader can consult [65, 64, 66] for some references on the code and it's potentialities.

Looking at the status of the library in the beginning of this work and the point that it has reached today, the evolution is quite significant. When this thesis was started, *Life* could handle, for instance in 2D, the Lagrange basis built only with Equidistributed points. The connectivity issues were already handled in 2D, but not in 3D. The Domain Specific Embedded Language was already available, but could only tackle scalar functions. For example, a solver for the Navier-Stokes equations could be coded, but component by component. Now, a standard variational formulation of these equations in vectorial form can be easily programmed.

All the points sets, high order geometrical mappings, integration in faces, 3D connectivity, interface with linear algebra solvers (Trilinos), Yosida schemes, ALE framework and the fluid-structure interaction solvers are some of the main contributions of the author to the library.

All of these algorithms, by themselves, or their combinations in a fluid-structure interaction solver, are the main contribution of this thesis. For instance, all the algebraic factorization Yosida methods, up to the author's knowledge, had never been implemented in such a general framework, independent of the type of mesh and it's topological dimension.

Another contribution is the reproducibility (see [23]) of many results that can be found in the literature. Some examples are the convergence of a certain type of spectral element or the condition number of the generalized Vandermonde matrix using several bases and point sets.

Some of the results shown in this thesis are already quite known, while others are being published for the first time.

Thesis Outline

In chapter 1 we introduce the basic tools to formulate spectral discretizations, with special focus on the Galerkin and Galerkin with Numerical Integration (G-NI) variants. We show how to construct the basis functions used by the spectral method in dD, $d = 1, 2, 3$, in several reference elements, and on how to differentiate and integrate them. In our construction we express polynomial sets in terms of an orthonormal basis (Dubiner for simplices and Legendre for tensorized domains), namely, the Lagrange polynomials associated with a certain point set. A numerical study is presented to estimate the Lebesgue constant associated with the point sets described in this chapter. Also, due to the approach we use to differentiate polynomials, we study the conditioning of the generalized Vandermonde matrix since it will influence the stability of the numerical algorithm.

In chapter 2 we present the multidomain extension of the spectral method. The issue of connectivity is addressed both for modal and nodal type bases and we explain how to create a map that relates the indices of the functions defined in the reference element with the ones defined in a triangulation of the domain. This construction is described for globally continuous bases functions, but also for expansions where the basis functions are continuous within each element of the partition and discontinuous across the faces. These basis functions are suitable for the discontinuous Galerkin methods. We finish this chapter with the study of a Poisson equation, in dD, $d = 1, 2, 3$. We show the spectral convergence of the method as well as some properties of condition number/eigenvalues of the stiffness/mass matrix associated with the method.

In chapter 3 we address the numerical approximation of the incompressible Stokes and Navier-Stokes equations in a fixed domain. We use the spectral element method to approximate the variables of the problem (velocity and pressure), but also for the geometry, i.e., we consider geometrical transformations that are polynomials of higher degree bigger than one. Regarding time integration, we introduce a combination of Backward Differentiation Formulas and an extrapolation formula of the same order. We propose some preconditioning strategies for the linear system obtained after standard space-time discretization and investigate their optimality in terms of number of iterations used by the GMRES method. Also, we test the Yosida-q methods in solving the same linear system (in the unsteady case) and show the expected convergence orders in time.

In chapter 4 we extend the algorithms developed in chapter 3 to the Arbitrary Lagrangian Eulerian (ALE) framework. Since we consider the case of having high order geometries describing the domain, we describe in detail the construction of a high order ALE map, responsible, at each time step, for description of the computational domain where the Navier-Stokes equations are to be solved. The Yosida-q schemes are also extended to this context. The last part of this chapter is dedicated to testing the developed solver with a simple benchmark problem.

In chapter 5 we apply the previous solver to simulate blood flow in large arteries. We use as test case a simple 2D hemodynamics problem and describe in detail the fluid and structure solvers and their coupling. We propose two strategies to tackle the FSI problem: an implicit fully coupled method and a semi-implicit method.

Chapter 1

The Spectral Method

In this chapter, we present the basic tools to formulate spectral discretizations with special focus on the Galerkin or Galerkin with Numerical Integration (G-NI) variants.

Let $\Omega \subset \mathbb{R}^d, d = 1, 2, 3$, denote the domain of the problem. In the following, we show how to construct the basis functions used by the spectral method in dD, $d = 1, 2, 3$. The basic ingredients of our construction are: (i) the reference (or parent) element $\hat{\Omega}$, (ii) a polynomial basis $\mathcal{B}$ defined in it, and (iii) a geometrical transformation that maps the reference element into the domain Ω, $\boldsymbol{\varphi} : \hat{\Omega} \longrightarrow \Omega$. This construction will be carried out in simplices and tensorized domains, with bases both of nodal and modal type.

1.1 Reference element

A reference element is defined by a domain $\hat{\Omega} \subset \mathbb{R}^d, d = 1, 2, 3$, containing the geometrical information associated, i.e., the relation between the several topological entities that compose the domain itself, and the coordinates of its vertices. The description of the reference element in terms of subentities is of crucial importance when dealing with bases defined in a mesh of several elements. Moreover, the numbering and orientation of these subentities will play a key role in addressing two problems: (i) defining continuous polynomial expansions in a multidomain setting, see section 2.1 and (ii) matching points in faces, see section 2.2. The former is related with the gluing of neighbor basis functions in a continuous Galerkin setting and the later is related with face integration in the discontinuous Galerkin method.

In this section, we consider the following reference elements: (i) the interval $(-1, 1)$, for $d = 1$ (ii) quadrilaterals and triangles, for $d = 2$ and (iii) tetrahedra and hexahedra for $d = 3$. We remark that in the three dimensional case, we could also consider prisms and pyramids as reference elements. The construction in terms of polynomial bases that will be exposed in this thesis is still valid, see Sherwin and Karniadakis [48]. We will refer to simplices, for segments, triangles and tetrahedra, and tensorized domains for quadrilaterals and hexahedra.

1.1.1 The reference interval

We define the reference element used when dealing with 1D geometries, the $(-1,1)$ interval and we denote it by $\mathcal{Q}^1$ or $\mathcal{T}^1$.

From a geometrical point of view, a closed interval is composed of two vertices and one edge connecting them. Let $\mathcal{E}_i^1$ denote the set of indices associated with a topological sub entity of dimension i in the reference interval. The decomposition of $\mathcal{Q}^1$ in topological subentities is given by

$$\begin{aligned} \mathcal{E}_0^1 &= \{0,1\} \\ \mathcal{E}_1^1 &= \{0\} \end{aligned}$$

where $\mathcal{E}_0^1$ and $\mathcal{E}_1^1$ denote the set of global indices of vertices and edges, respectively. This numbering system is depicted in Figure 1.1. For the edge defined by the two vertices of

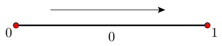


Figure 1.1: Local and global numbering for the interval.

the interval, we also assign a local vertex numbering (that in this case coincides with the global one). This introduces an orientation in the edge (illustrated by the arrow in Figure 1.1).

From an implementation point of view, the description of the relations between the reference element and its topological subentities can be achieved by defining an appropriate set of functions: `elementToPoint`, `elementToEdge` and `edgeToPoint`. They are defined as follows

$$\begin{aligned} \texttt{elementToPoint}: &\ \mathcal{E}_1^1 \times \mathcal{E}_0^1 \longrightarrow \mathcal{E}_0^1, && \texttt{elementToPoint}(0,i) = i, && \forall i \in \mathcal{E}_0^1, \\ \texttt{elementToEdge}: &\ \mathcal{E}_1^1 \times \mathcal{E}_1^1 \longrightarrow \mathcal{E}_1^1, && \texttt{elementToEdge}(0,0) = 0, && \\ \texttt{edgeToPoint}: &\ \mathcal{E}_1^1 \times \mathcal{E}_0^1 \longrightarrow \mathcal{E}_0^1, && \texttt{edgeToPoint}(0,i) = i, && \forall i \in \mathcal{E}_0^1. \end{aligned}$$

The function `edgeToPoint` coincides with `elementToPoint` in this case (it will be different for the forthcoming reference elements). These functions are responsible for providing an orientation to the reference element.

1.1.2 The reference triangle and square

For the reference triangle and square, we consider

$$\mathcal{T}^2 = \left\{(x_1,x_2) \in \mathbb{R}^2 \mid -1 < x_1, x_2 < 1,\ x_1 + x_2 < 0\right\}$$

and

$$\mathcal{Q}^2 = \left\{(x_1,x_2) \in \mathbb{R}^2 \mid -1 < x_1, x_2 < 1\right\}$$

and the vertices and edges of its closure.

Let us introduce again the decomposition of each reference element in terms of its subentities. For the triangle, we define

$$\begin{aligned}\mathcal{E}_0^2 &= \{0,1,2\}\\ \mathcal{E}_1^2 &= \{0,1,2\}\\ \mathcal{E}_2^2 &= \{0\}\end{aligned}$$

and for the square

$$\begin{aligned}\mathcal{E}_0^2 &= \{0,1,2,3\}\\ \mathcal{E}_1^2 &= \{0,1,2,3\}\\ \mathcal{E}_2^2 &= \{0\}.\end{aligned}$$

For both reference elements considered, we define the functions `elementToPoint`, `elementToEdge` and `edgeToPoint`. The first two are defined in the same way for both reference triangle and square:

$$\texttt{elementToPoint} : \mathcal{E}_2^2 \times \mathcal{E}_0^2 \longrightarrow \mathcal{E}_0^2, \qquad \texttt{elementToPoint}(0,i) = i,\ \forall i \in \mathcal{E}_0^2$$

$$\texttt{elementToEdge} : \mathcal{E}_2^2 \times \mathcal{E}_1^2 \longrightarrow \mathcal{E}_1^2, \qquad \texttt{elementToEdge}(0,i) = i,\ \forall i \in \mathcal{E}_1^2$$

The function `edgeToPoint` is different for the triangular and quadrangular reference elements. In the first case it is defined as

$$\texttt{edgeToPoint} : \mathcal{E}_1^2 \times \mathcal{E}_0^1 \longrightarrow \mathcal{E}_0^2, \qquad \texttt{edgeToPoint}(i,j) = (i+j+1)\%3,\ \forall (i,j) \in \mathcal{E}_1^2 \times \mathcal{E}_0^1.$$

and in the quadrangular case, as

$$\texttt{edgeToPoint} : \mathcal{E}_1^2 \times \mathcal{E}_0^1 \longrightarrow \mathcal{E}_0^2, \qquad \texttt{edgeToPoint}(i,j) = (i+j)\%4,\ \forall (i,j) \in \mathcal{E}_1^2 \times \mathcal{E}_0^1.$$

where % denotes the remainder of the integer division. We remark that this last function introduces an orientation in the edges of the reference elements. For both reference triangle

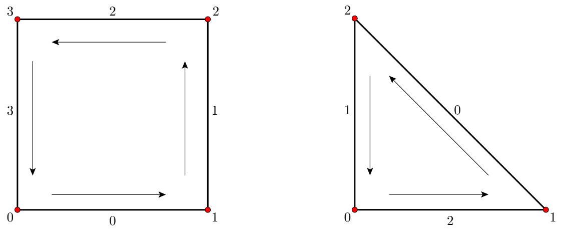


Figure 1.2: Local numbering and orientation for vertices and edges in the quadrilateral and triangle.

and square, the numbering and orientations just defined are depicted in Figure 1.2. The same figure also shows the global index of the vertices of each reference element.

We finish by remarking that the numbering/orientation described here is different from the ones described in Canuto, Hussaini, Quarteroni and Zang [12] and Sherwin and Karniadakis [48].

1.1.3 The reference tetrahedron and hexahedron

Regarding the tetrahedron and hexahedron, we consider the following domains

$$\mathcal{T}^3 = \left\{(x_1, x_2, x_3) \in \mathbb{R}^3 \mid -1 < x_1, x_2, x_3 < 1, \quad x_1 + x_2 + x_3 < 0\right\}$$

and

$$\mathcal{Q}^3 = \left\{(x_1, x_2, x_3) \in \mathbb{R}^3 \mid -1 < x_1, x_2, x_3 < 1\right\}.$$

and the vertices, edges and faces of its closure.

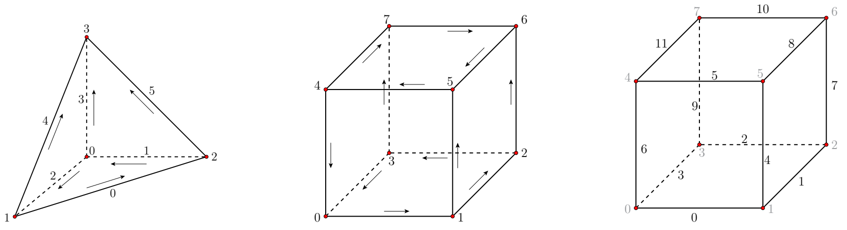
We define similar numbering and orientations as before. First, we provide the decomposition of these reference elements in terms of topological subentities. For the tetrahedron we have

$$\begin{array}{rcl}
\mathcal{E}_0^3 & = & \{0,1,2,3\} \\
\mathcal{E}_1^3 & = & \{0,1,2,3,4,5\} \\
\mathcal{E}_2^3 & = & \{0,1,2,3\} \\
\mathcal{E}_3^3 & = & \{0\}
\end{array}$$

and for the hexahedron

$$\begin{array}{rcl}
\mathcal{E}_0^3 & = & \{0,1,2,3,4,5,6,7\} \\
\mathcal{E}_1^3 & = & \{0,1,2,3,4,5,6,7,8,9,10,11\} \\
\mathcal{E}_2^3 & = & \{0,1,2,3,4,5\} \\
\mathcal{E}_3^3 & = & \{0\}
\end{array}$$

In Figure 1.3 we display the numbering and orientation of the vertices and edges composing the reference elements.



(a) Vertices and edges. (b) Vertices. (c) Edges.

Figure 1.3: Numbering and orientation for vertices and edges in the tetrahedron and hexahedron.

In the three dimensional case, we define the functions `elementToPoint`, `elementToEdge`, `elementToFace`, `faceToPoint`, `faceToEdge` and `edgeToPoint`. The first three functions have the same definition for both reference elements:

$$\begin{array}{rlll}
\texttt{elementToPoint}: & \mathcal{E}_3^3 \times \mathcal{E}_0^3 \longrightarrow \mathcal{E}_0^3, & \texttt{elementToPoint}(0,i) = i, & \forall i \in \mathcal{E}_0^3, \\
\texttt{elementToEdge}: & \mathcal{E}_3^3 \times \mathcal{E}_1^3 \longrightarrow \mathcal{E}_1^3, & \texttt{elementToEdge}(0,i) = i, & \forall i \in \mathcal{E}_1^3, \\
\texttt{elementToFace}: & \mathcal{E}_3^3 \times \mathcal{E}_2^3 \longrightarrow \mathcal{E}_2^3, & \texttt{elementToFace}(0,i) = i, & \forall i \in \mathcal{E}_2^3.
\end{array}$$

The last three need to be defined differently for each one of the reference elements. To simplify their definition, we present these functions in the form of an array. The first row has the indices of the topological entity with bigger dimension and the first left column has the local indices of the underlying sub entity.

In the case of the tetrahedron, these functions are defined as follows:

(i) `faceToPoint` : $\mathcal{E}_2^3 \times \mathcal{E}_0^2 \longrightarrow \mathcal{E}_0^3$

	0	1	2	3
0	1	0	0	0
1	2	2	1	1
2	3	3	2	3

Remark 1.1.1. *The indices 0,1,2,3 in the top row correspond to the indices of the faces in the element (in the case of the tetrahedra, they are four). The indices 0,1,2 in the left column correspond to the local indices of the points in a triangular face.*

(ii) `faceToEdge` : $\mathcal{E}_2^3 \times \mathcal{E}_1^2 \longrightarrow \mathcal{E}_1^3$

	0	1	2	3
0	0	1	2	2
1	5	3	1	4
2	4	5	0	3

(iii) `edgeToPoint` : $\mathcal{E}_2^3 \times \mathcal{E}_0^1 \longrightarrow \mathcal{E}_0^3$

	0	1	2	3	4	5
0	1	2	0	0	1	2
1	2	0	1	3	3	3

The numbering and orientations on vertices, edges and faces for the tetrahedron are displayed in Figures 1.3(a) and 1.4. Each face in Figure 1.4 is then oriented as in section 1.1.2.

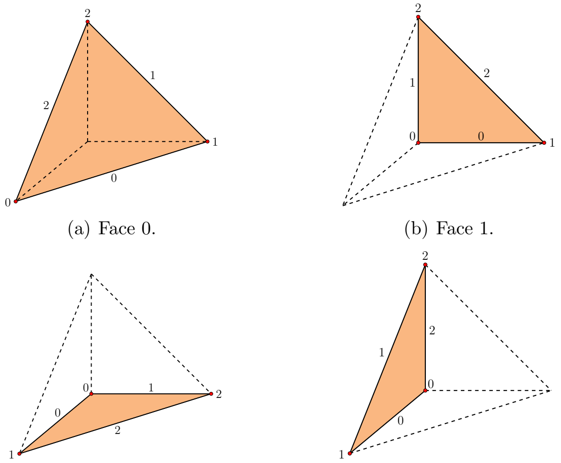
Regarding the hexahedron, we define the functions

(i) `faceToPoint` : $\mathcal{E}_2^3 \times \mathcal{E}_0^2 \longrightarrow \mathcal{E}_0^3$

	0	1	2	3	4	5
0	0	0	1	2	3	4
1	1	1	2	3	0	5
2	2	5	6	7	4	6
3	3	4	5	6	7	7

(ii) `faceToEdge` : $\mathcal{E}_2^3 \times \mathcal{E}_1^2 \longrightarrow \mathcal{E}_1^3$

	0	1	2	3	4	5
0	0	0	1	2	3	5
1	1	4	7	9	6	8
2	2	5	8	10	11	10
3	3	6	4	7	9	11



(a) Face 0. (b) Face 1. (c) Face 2. (d) Face 3.

Figure 1.4: Local numbering for the faces of the tetrahedron.

(iii) `edgeToPoint` : $\mathcal{E}_2^3 \times \mathcal{E}_0^1 \longrightarrow \mathcal{E}_0^3$

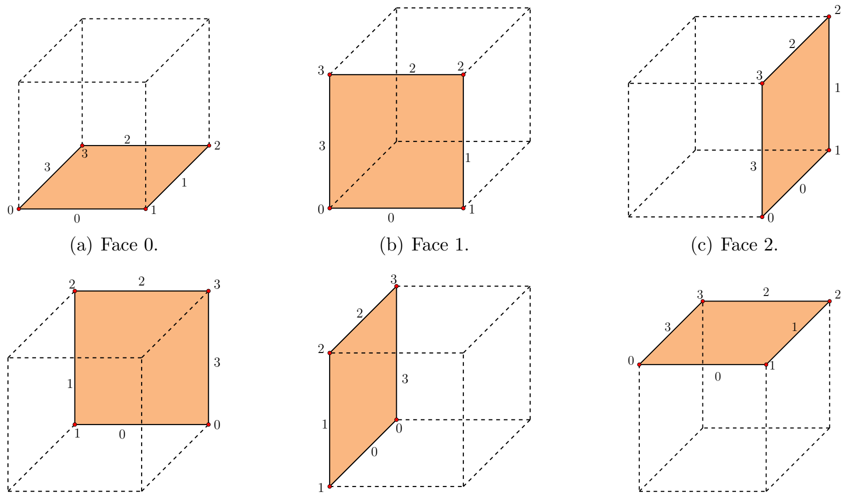
	0	1	2	3	4	5	6	7	8	9	10	11
0	0	1	2	3	1	5	4	2	6	3	7	4
1	1	2	3	0	5	4	0	6	5	7	6	7

The complete numbering and orientation of the vertices, edges and faces for the hexahedron is displayed in Figures 1.3(b), 1.3(c) and 1.5. Again, each face in Figure 1.5 is then oriented as in section 1.1.2.

1.2 Point Sets

In finite element or spectral methods, point sets have an important role in three main applications: (i) numerical integration (ii) nodal bases definition and (iii) polynomial differentiation. Within the context of defining nodal bases, they are used to determine projections into nodal subspaces, construct the table of degrees of freedom (see section 2.1) and visualization of arbitrary degree polynomials (see section 2.3).

We distinguish two main classes (not necessarily exclusive): *interpolation* point sets and *quadrature* point sets.


(a) Face 0. (b) Face 1. (c) Face 2.

(d) Face 3. (e) Face 4. (f) Face 5.

Figure 1.5: Local numbering for the faces of the hexahedron.

1.2.1 Quadrature point sets

Starting again with the one dimensional case, quadrature formulas can be constructed by using zeros of Legendre polynomials defined in $(-1, 1)$ and/or its boundary points, originating the Gauss-Legendre, Gauss-Radau-Legendre and Gauss-Lobatto-Legendre quadrature formulas in 1D.

Let us briefly review these quadrature formulas (see e.g. Canuto, Hussaini, Quarteroni and Zang [12]).

Gauss-Legendre

Let $x_0 < x_1 < \cdots < x_N$ be the roots of the $(N+1)$-th Legendre polynomial, L_{N+1}, and let $w_0, \ldots, w_N$ be the solution of the linear system

$$\sum_{j=0}^{N} (x_j)^k w_j = \int_{-1}^{1} x^k \, dx, \quad 0 \leqslant k \leqslant N. \tag{1.1}$$

Then,

(i) $w_j > 0$, for $j = 0, \ldots, N$ and

$$\sum_{j=0}^{N} p(x_j) w_j = \int_{-1}^{1} p(x) \, dx, \qquad \text{for all } p \in \mathbb{P}_{2N+1}(-1,1)$$

where $\mathbb{P}_{2N+1}(-1,1)$ denotes the polynomials of degree less or equal to $2N+1$ defined in $(-1,1)$.

The positive numbers w_j are called *weights*. They are also given by

$$w_j = \int_{-1}^{1} \ell_j(x)\ dx$$

being ℓ_j the characteristic Lagrange polynomial of degree N such that $\ell_j(x_i) = \delta_{ij}$.

(ii) It is not possible to find x_j, w_j, $j = 0, \dots, N$, such that (1.1) holds for all polynomials $p \in \mathbb{P}_{2N+2}(-1,1)$.

This is the well known Gauss-Legendre quadrature formula. However, the roots, which define a point set, do not include any points in the boundary. The inclusion of such points provides the Gauss-Radau-Legendre and Gauss-Lobatto-Legendre quadratures. Let us start by the first.

Gauss-Radau-Legendre

The Gauss-Radau-Legendre quadrature includes one point in the boundary of the interval $(-1,1)$. We construct it after the following polynomial:

$$q(x) = L_{N+1}(x) + aL_N(x), \tag{1.2}$$

where a is such that $q(-1) = 0$, that is,

$$a = -L_{N+1}(-1)/L_N(-1).$$

Let $x_0 < x_1 < \cdots < x_N$ be the roots of (1.2) and let $w_0, \dots, w_N$ the solution of the linear system

$$\sum_{j=0}^{N} (x_j)^k w_j = \int_{-1}^{1} x^k\ dx, \quad 0 \leqslant k \leqslant N.$$

Then

$$\sum_{j=0}^{N} p(x_j) w_j = \int_{-1}^{1} p(x)\ dx, \quad \text{for all } p \in \mathbb{P}_{2N}(-1,1).$$

In order to have the Gauss-Radau formula to include the point $x = 1$, the variable a is taken in such a way that $q(1) = 0$ and a similar result as the previously presented is valid.

Gauss-Lobatto-Legendre

Finally, the Gauss-Lobatto-Legendre quadrature formula is obtained by considering

$$q(x) = L_{N+1}(x) + aL_N(x) + bL_{N-1}(x), \tag{1.3}$$

where a and b are chosen such that $q(-1) = q(1) = 0$.

Let $x_0 < x_1 < \cdots < x_N$ be the roots of (1.3) and let $w_0, \ldots, w_N$ be the solution of the linear system

$$\sum_{j=0}^{N} (x_j)^k w_j = \int_{-1}^{1} x^k \, dx, \quad 0 \leqslant k \leqslant N.$$

Then

$$\sum_{j=0}^{N} p(x_j) w_j = \int_{-1}^{1} p(x) \, dx, \quad \text{for all } p \in \mathbb{P}_{2N-1}(-1,1).$$

Remark 1.2.1. *For simplicity in the terminology we will refer to the previous quadrature formulas, or just the point sets described, as* Gauss*,* Gauss-Radau *or* Gauss-Lobatto*, respectively.*

The extension of these quadrature formulas to quadrangles and hexahedra is straightforward using tensorization, see Sherwin and Karniadakis [48] or Canuto, Hussaini, Quarteroni and Zang [12].

Quadrature formulas in simplices

For simplices we mention two types of quadrature formulas: (i) the *general formulas* and (ii) the *optimal formulas*. The former are based on Gauss type formulas defined in the simplex products and can be extended to any degree. The latter have to be defined one by one for the simplices but they use less integration points than the general formulas to integrate exactly the same polynomial degree.

General formulas can be constructed by applying a suitable transformation to the quadrature formulas defined in the square/hexahedron. In 2D consider the mappings

$$\mathbf{m} \; : \; \mathcal{T}^2 \to \mathcal{Q}^2, \;\; (x_1, x_2) \mapsto (\xi_1, \xi_2) = \left(2\frac{1 + x_1}{1 - x_2} - 1, x_2 \right) \tag{1.4}$$

$$\mathbf{m}^{-1} \; : \; \mathcal{Q}^2 \to \mathcal{T}^2, \;\; (\xi_1, \xi_2) \mapsto (x_1, x_2) = \left(\frac{1}{2}(1 + \xi_1)(1 - \xi_2) - 1, \xi_2 \right), \tag{1.5}$$

between $\mathcal{T}^2$ and $\mathcal{Q}^2$ and

$$Q_{\mathcal{Q}^2} = \{(x_i, w_i)\}_{i=0}^{(N+1)^2}$$

a quadrature formula in the tensorized domain $\mathcal{Q}^2$. Then

$$Q_{\mathcal{T}^2} = \left\{ (\mathbf{m}^{-1}(x_i), J_{\mathbf{m}}(x_i) \cdot w_i) \right\}_{i=0}^{(N+1)^2}$$

is as a quadrature formula for $\mathcal{T}^2$, where $J_{\mathbf{m}}$ is the Jacobian of (1.4). In Figure 1.6 we display the effect of transformation $\mathbf{m}^{-1}$ to a set of quadrature points in $\mathcal{Q}^2$. The map (1.4) is called *collapsed coordinate system.*

For tetrahedra, this construction can be conducted using transformation (1.15).

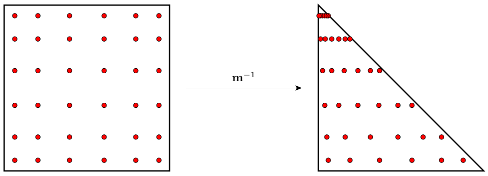


Figure 1.6: Transformation $\mathbf{m}^{-1}$ acting on a tensorized Gauss point set.

Remark 1.2.2. *The singularity in the collapsed coordinate system transformation does not allow this procedure to be applied to all quadrature formulas defined in the quadrilateral. It can only be applied to quadrature rules that do not have points in the top edge of the quadrilateral. Both Gauss and Gauss-Radau quadrature formulas can be used, but not Gauss-Lobatto. Similar considerations are valid for hexahedra and tetrahedra.*

The reason why to consider non optimal quadrature rules in simplices is related with implementation reasons. While arbitrary degree quadrature rules for tensorized domains can be generated automatically through the zeros of specific polynomials, for simplicial domains, this is not the case. Formulas exist, but they cannot be generated in a simple and unified fashion and have to be hard coded. We refer the reader to Solin, Segeth and Dolezel [79]. If one seeks performance, then these later are the choice, but they have to be programmed for each degree. The simulations in this thesis use, in 2D and 3D, whenever possible, such formulas up to $N = 20$. In Table 1.1 we show a comparison between the two types of quadratures formulas in terms of the number of points used.

	2	4	6	8	10	12	14	16	18	20
Gauss	4	9	16	25	36	49	64	81	100	121
Optimal	3	6	12	16	25	33	42	52	70	79

	2	4	6	8	10	12	14	16	18	20
Gauss	8	27	64	125	216	343	512	729	1000	1331
Optimal	4	11	24	43	126	210	330	495	715	1001

Table 1.1: Number of points needed to integrate a polynomial of degree N using general and optimal formulas for $d = 2$ (top) and $d = 3$ (bottom).

1.2.2 Interpolation point sets

Interpolation point sets are points that are used to generate *Lagrange* basis, see section 1.3.3. Let $N > 0$ be an integer and T_N a quantity that depends on the reference element and N and is defined as

$$T_N = \begin{cases} (N+1)^d - 1, & \text{for } \mathcal{Q}^d, \quad d = 1,2,3 \\ \binom{N+d}{N} - 1, & \text{for } \mathcal{T}^d, \quad d = 2,3. \end{cases} \tag{1.6}$$

We denote $\{x_i\}_{i=0}^{T_N}$ a point set and $\{\ell_i\}_{i=0}^{T_N}$ the Lagrange polynomials associated. The Lebesgue constant Λ_{T_N} is defined as

$$\Lambda_{T_N} = \max_{\xi \in \hat{\Omega}} \sum_{i=0}^{T_N} |\ell_i(\xi)|. \tag{1.7}$$

Let $f : \hat{\Omega} \longrightarrow \mathbb{R}$ denote a function and $\Pi f : \hat{\Omega} \longrightarrow \mathbb{R}$ its interpolating polynomial of degree T_N that verifies

$$\Pi f(x_i) = f(x_i), \quad \forall i = 0, \dots, T_N. \tag{1.8}$$

Then, it can be shown that

$$\|f - \Pi f\|_\infty \leqslant (1 + \Lambda_{T_N}) \|p^* - f\|_\infty \tag{1.9}$$

where p^* denotes the best approximating polynomial in the max-norm

$$\|f\|_\infty = \max_{\xi \in \hat{\Omega}} |f(x)|.$$

We are interested in points such that the growth of the Lebesgue constant in T_N is moderate. Due to inequality (1.9), this constant can be considered as a measure of the "good" approximation properties of the Lagrange basis associated. More details can be found in Canuto, Hussaini, Quarteroni and Zang [12] or Sherwin and Karniadakis [48].

In the literature, several point sets have been proposed, for tensorized and simplicial domains. The Gaussian points (Gauss, Gauss-Radau and Gauss-Lobatto) are usually used to construct Lagrange bases in tensorized geometries due to their well behaved Lebesgue constants. For simplicial domains, there is no equivalent of the Gaussian points. The Equidistributed points are a first alternative, but these do not have low Lebesgue constants and, in the context of the Galerkin method, they lead to very ill conditioned linear systems, as will be seen in section 2.4. Other choices in the triangular case, more robust with respect to interpolation, are the Electrostatic [44], Fekete [85], Heinrichs [43] and more recently, Warpblend [90] points. A very interesting property shared by the Electrostatic, Fekete and Warpblend points is that, in the edges of the triangle where they are defined, they coincide with the Gauss-Lobatto points. This feature allows the use of hybrid meshes (composed of quadrangles and triangles) in a continuous Galerkin setting.

In the following, we present these point sets (except the Electrostatic) used to construct Lagrange basis for tensorized as well as simplicial domains.

Point set ordering

The ordering of the point set is done according to the numbering and orientations defined in section 1.1. First, the vertices, ordered by local index; second, the points that lie in the edges of the element, again ordered by local index, and so on. Within each topological subentity, take edges as an example, we order the points in edge 0 according to the orientation of this edge, and do the same to all edges of the reference element (see Figure 1.7). In 3D, the same is done to the points that lie in the faces.

For simplicity of exposure, we do not order the points in the following.

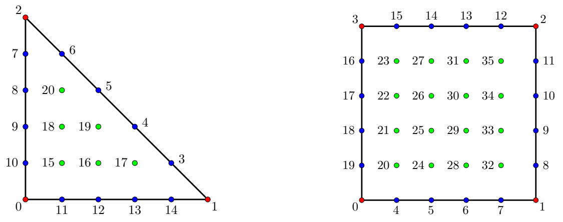


Figure 1.7: Example of the ordering of a point set in the reference triangle and quadrangle.

Equidistributed points

The definition of the Equidistributed points is quite straightforward in the reference domains we are considering. For the $[-1, 1]$ interval, given an integer $N > 0$, they are defined as the set

$$\left\{ x_i^{\text{equi}} : \ x_i^{\text{equi}} = \frac{2i}{N} - 1, \ i = 0, 1, \ldots, N \right\}$$

As we did in previous cases, the extension from the interval $[-1, 1]$ to quadrangles is done by tensorization

$$\left\{ x_{i,j}^{\text{equi}} : \ x_{i,j}^{\text{equi}} = \left(\frac{2i}{N} - 1, \frac{2j}{N} - 1 \right), \ i, j = 0, 1, \ldots, N \right\}$$

Finally, for triangles, the Equidistributed point set reads as

$$\left\{ x_{i,j}^{\text{equi}} : \ x_{i,j}^{\text{equi}} = \left(\frac{2i}{N} - 1, \frac{2j}{N} - 1 \right), \ 0 \leqslant i + j \leqslant N \right\}$$

For hexahedra and tetrahedra, the generalization is straightforward from the previous definitions.

Warpblend points

The Warpblend point set is introduced in Warburton [90] as a point distribution that allows to create arbitrary degree interpolation points based on the Equidistributed and Gauss-Lobatto points mentioned previously. This construction is done in the reference triangle and tetrahedron.

The Warpblend points are constructed by defining a transformation from the Equidistributed to the Gauss-Lobatto points in the interval $[-1, 1]$. and blending this deformation for the Equidistributed points inside an equilateral triangle. These points, as will be seen later in this section, have relatively low Lebesgue constants and have, in the edges of the triangle, the Gauss-Lobatto points. A similar construction can be done for tetrahedra. However, the possibility of having hybrid meshes is no longer available, if hexahedra and Gauss-Lobatto points are to be used in the nodal expansion.

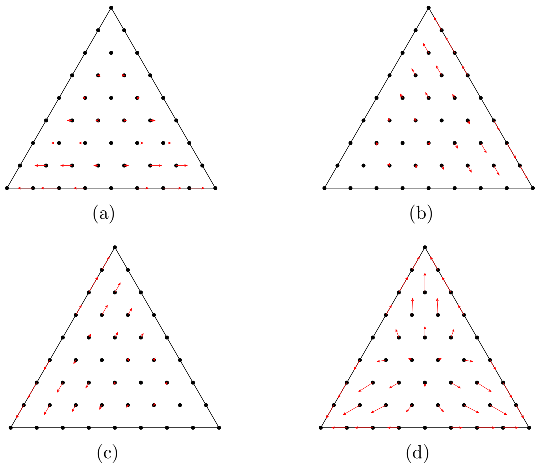


Figure 1.8: (a-c): warp and blend functions for each of the three edges, constructed with warping which reproduces a 8 point Gauss-Lobatto point set. (d): sum of warp and blend functions for the three edges.

We address now the construction of this point set in 2D and refer the reader to Warburton [90] for the details on how to construct the points in the tetrahedron. Given the $N+1$ Gauss-Lobatto points in the interval $[-1,1]$,

$$\left\{x_i^{\mathrm{gl}},\ 0 \leqslant i \leqslant N\right\},$$

we define a deformation function

$$\omega(x) = \sum_{i=0}^{N}\left(x_i^{\mathrm{gl}} - x_i^{\mathrm{equi}}\right)\prod_{j=0, j\neq i}^{N}\left(\frac{x - x_j^{\mathrm{equi}}}{x_i^{\mathrm{equi}} - x_j^{\mathrm{equi}}}\right)$$

where x_i^{equi} are the Equidistributed points defined in the previous section.

We consider now a reference triangle with vertices

$$\mathbf{v}^1 = \left(-1, -\frac{1}{\sqrt{3}}\right), \quad \mathbf{v}^2 = \left(1, -\frac{1}{\sqrt{3}}\right), \quad \mathbf{v}^3 = \left(0, \frac{2}{\sqrt{3}}\right).$$

We extend the edge warp into the triangle by blending in the edge normal direction. Therefore, given the barycentric coordinates of a point in the previous triangle, we define the *warp and blend* transformation that achieves this results as being the product of the following functions

$$\mathbf{w}^1(\lambda^1, \lambda^2, \lambda^3) = \omega(\lambda^3 - \lambda^2)(1, 0)$$

and

$$b^1(\lambda^1,\lambda^2,\lambda^3) = \left(\frac{2\lambda^3}{2\lambda^3+\lambda^1}\right)\left(\frac{2\lambda^2}{2\lambda^2+\lambda^1}\right)$$

This is the transformation for edge one. For the other edges, the warping functions can be expressed by permuting $\lambda^1, \lambda^2, \lambda^3$:

$$\mathbf{w}^2(\lambda^1,\lambda^2,\lambda^3) = \omega(\lambda^1-\lambda^3)\left(-\frac{1}{2}, \sqrt{\frac{3}{2}}\right)$$

$$\mathbf{w}^3(\lambda^1,\lambda^2,\lambda^3) = \omega(\lambda^2-\lambda^1)\left(-\frac{1}{2}, -\sqrt{\frac{3}{2}}\right)$$

and the blending functions are

$$b^2(\lambda^1,\lambda^2,\lambda^3) = \left(\frac{2\lambda^3}{2\lambda^3+\lambda^2}\right)\left(\frac{2\lambda^1}{2\lambda^1+\lambda^2}\right)$$

$$b^3(\lambda^1,\lambda^2,\lambda^3) = \left(\frac{2\lambda^2}{2\lambda^2+\lambda^3}\right)\left(\frac{2\lambda^1}{2\lambda^1+\lambda^3}\right),$$

respectively.

The final coordinate shift can be expressed as

$$\mathbf{g}(\lambda^1,\lambda^2,\lambda^3) = b^1\mathbf{w}^1 + b^2\mathbf{w}^2 + b^3\mathbf{w}^3. \tag{1.10}$$

To further obtain a point set in $\mathcal{T}^2$, we need to transform the equilateral triangle into our reference element $\mathcal{T}^2$. This can be done by a linear transformation, see section 1.4.

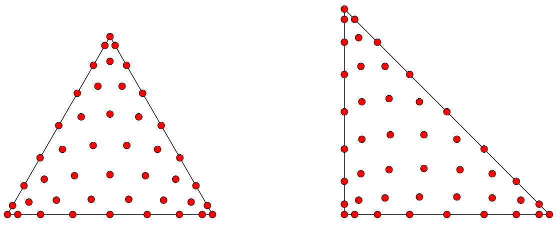


Figure 1.9: Warpblend points corresponding to the 8 point Gauss-Lobatto point set in the equilateral triangle (left) and the reference element (right).

It is possible to improve further the Lebesgue constant associated with the set just defined by introducing a parameter α in transformation (1.10).

$$\mathbf{g}(\lambda^1,\lambda^2,\lambda^3) = \left(1+(\alpha\lambda^1)^2\right)b^1\mathbf{w}^1 + \left(1+(\alpha\lambda^2)^2\right)b^2\mathbf{w}^2 + \left(1+(\alpha\lambda^3)^2\right)b^3\mathbf{w}^3.$$

	triangle	tetrahedron
α_3	1.4152	0
α_4	0.1001	0.1002
α_5	0.2751	1.1332
α_6	0.9808	1.5608
α_7	1.0999	1.3413
α_8	1.2832	1.2577
α_9	1.3648	1.1603
α_{10}	1.4773	1.0153
α_{11}	1.4959	0.6080
α_{12}	1.5743	0.4523
α_{13}	1.5770	0.8856
α_{14}	1.6223	0.8717
α_{15}	1.6258	0.9655

Table 1.2: Values of the optimization parameter α for the Warpblend point set.

A complete study of the growth of the Lebesgue constant for these points will be given in section 1.2.2.

In Warburton [90], the values for parameter α are given up to $N = 15$ for the two and three dimensional point sets. We reproduce these values in Table 1.2. For higher degree, we set $\alpha = \frac{5}{3}$ for $d = 2$ and $\alpha = 1$ for $d = 3$.

Fekete points

Let us consider a basis $\mathcal{B} = \{\phi_i\}_{i=0}^{T_N}$, a point set $X = \{x_i\}_{i=0}^{T_N}$ and define the generalized Vandermonde matrix associated

$$V(\mathcal{B}, X) = \begin{bmatrix} \phi_0(x_0) & \dots & \phi_0(x_{T_N}) \\ \vdots & & \vdots \\ \phi_{T_N}(x_0) & \dots & \phi_{T_N}(x_{T_N}) \end{bmatrix} \tag{1.11}$$

The Fekete points are a set of points $\{\xi_0, \dots, \xi_{T_N}\}$ that maximize, for a fixed basis $\mathcal{B}$, the determinant of (1.11), ie,

$$\max_{\xi_i} |V(\mathcal{B}, \{\xi_0, \dots, \xi_{T_N}\})|.$$

It is known that in the interval $[-1, 1]$, the Fekete points are the Gauss-Lobatto quadrature points. This was proven by Fejer [28] and was extended to the square and cube by Bos, Taylor and Wingate [3]. To evaluate the Fekete points, Taylor, Wingate and Vincent [85] proposed a steepest-descent algorithm to determine the points that provide the maximum determinant of the generalized Vandermonde matrix.

A consequence of this property is that the Lagrange polynomials constructed with these points achieve their maximum at the nodal points. This implies a bound to the Lebesgue constant given by

$$\Lambda_{T_N} = \max_{\xi \in \hat{\Omega}} \sum_{i=0}^{T_N} |\ell_i(\xi)| \leqslant T_N. \tag{1.12}$$

Evaluation of the Lebesgue constants

To calculate the Lebesgue constant Λ_{T_N}, we need to compute the maximum (1.7). For this, we evaluated the sum at sufficiently high number of Equidistributed points, that is, 170 points in $\mathcal{Q}^1$, $\binom{100+d}{100}$ points in $\mathcal{T}^d$, $d = 2, 3$ and 31^d points in $\mathcal{Q}^d$, $d = 2, 3$. We plot

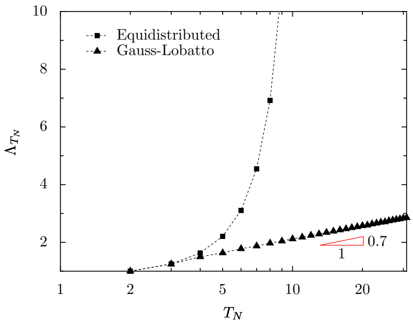


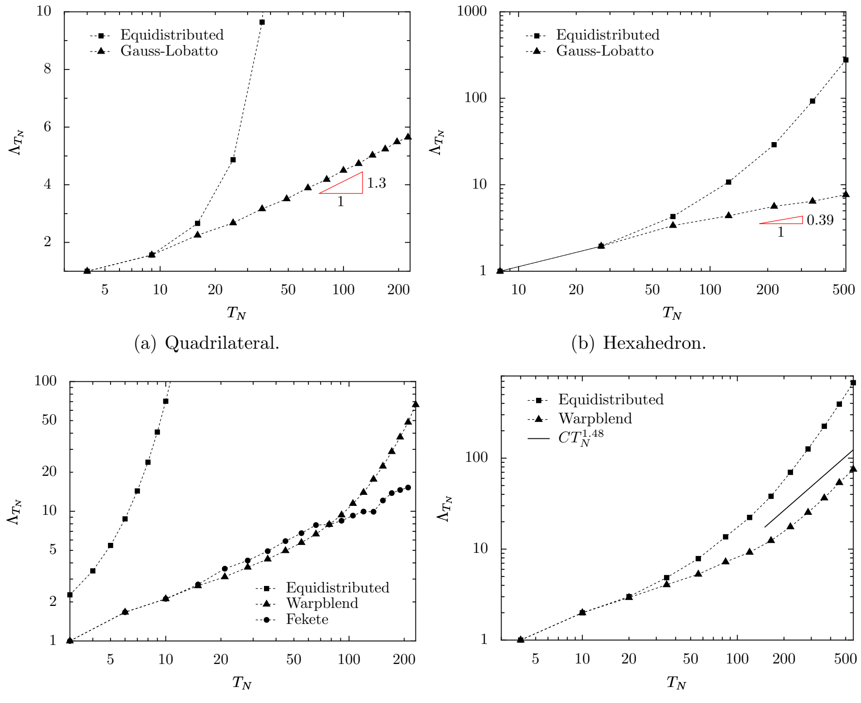
Figure 1.10: Lebesgue constants in the reference line.

in Figures 1.10 and 1.11 the Lebesgue constants that we obtained. We start by observing that the Lebesgue constant associated with Equidistributed point set is too big for $N \geqslant 5$. This is a well known fact and they are not suited as interpolation points, see Trefethen and Weideman [87].

Regarding the one dimensional case, we recover a known result concerning the growth of the Lebesgue constant associated with the Gauss-Lobatto points, namely, that it grows as $\mathcal{O}(\log(T_N))$, see Sherwin and Karniadakis [48].

For the two dimensional case, we start by discussing the triangular points. We notice that the Fekete points have lower Lebesgue constants than the Warpblend ones for $N > 12$. We estimated the Lebesgue constant for the Warpblend points grow as $\mathcal{O}(1.014^{T_N})$ and for the Fekete points as $\mathcal{O}(T_N^{0.62})$. For the Gauss-Lobatto point set defined in the reference square, we estimated the Lebesgue constant to grow as $\mathcal{O}(\log(T_N))$.

Finally, for the three dimensional point sets, we observe a growth of $\mathcal{O}(T_N^{1.48})$ for the Warpblend points and $\mathcal{O}(T_N^{0.39})$ for the Gauss-Lobatto one's. The last result is in agreement with (1.12). We note that the Lebesgue constant associated with the 3D equidistributed points exhibits again an exponential behavior.



(a) Quadrilateral. (b) Hexahedron.

(c) Triangle. (d) Tetrahedron.

Figure 1.11: Lebesgue constants for the reference simplices and tensorized domains.

1.3 Polynomial bases

There are two ways to define polynomial bases in the reference element. One is to calculate formally the expression of the polynomials. These expressions can then be implemented but they are different for each polynomial degree and reference element. The other possibility, and the one followed in this work, is to use algebraic calculus. This approach consists in having a (fixed) basis $\mathcal{B} = \{\hat{\phi}_i\}_{i=0}^{T_N}$, called *prime basis*, defined in the reference element, $\hat{\Omega}$ in which all other polynomials defined in the element are expressed. The prime basis should be a modal basis and provide well conditioned generalized Vandermonde matrices, see section 1.3.2. The requirement of modality for the prime basis is justified by the fact that it allows in a straightforward way to extract subsets of bases functions.

Let $\mathcal{P} = \{\hat{p}_j\}$ be another polynomial family. Then there exist coefficients $\alpha_{ij} \in \mathbb{R}$ such that

$$\hat{p}_j = \sum_{i=0}^{T_N} \alpha_{ij} \hat{\phi}_i, \quad \forall j.$$

Let $C_{\mathcal{P}}$ be a matrix containing the entries α_{ij}. Then, $\mathcal{P}$ is represented by the prime basis $\mathcal{B}$ and the matrix $C_{\mathcal{P}}$. One advantage of this approach is that it reduces operations like evaluation, differentiation and integration of polynomials to matrix-matrix of matrix-vector operations, see section 1.3.1. At this level, BLAS or LAPACK optimized routines can be used to boost performance. From a programming perspective, the prime basis should support evaluation, differentiation and integration on its own. This means that these three operations should be implemented for each polynomial of the basis. We refer the reader to Warburton, Pavarino and Hesthaven [91], Sherwin and Karniadakis [77] and Pasquetti and Rapetti [60] for more details regarding this approach.

On top of this algebraic polynomial construction, we define finite elements in the sense of Ciarlet [16]. Let W be a vector space spanned by $\mathcal{B}$ and Σ a set of linear forms $\{\sigma_0, \dots, \sigma_{T_N}\}$ that form a basis for the space of linear functionals in X. Then, Σ induces a basis

$$\mathcal{B}^* = \{\theta_0, \dots, \theta_{T_N}\}$$

for W that satisfies

$$\sigma_i(\theta_j) = \delta_{ij}, \quad \forall i, j = 0, \dots, T_N. \tag{1.13}$$

By expressing the functions in $\mathcal{B}^*$ in terms of the prime basis $\mathcal{B}$, equation (1.13) allows to calculate $C_{\mathcal{B}^*}$. We highlight that this approach to define finite elements is used in section 1.3.3 to construct the Lagrange polynomials in $\hat{\Omega}$, but the same framework can be used to construct other finite elements: (i) the Raviart-Thomas element or (ii) the Nédélec element. For more details on these finite elements, see Ern and Guermond [26] or Ciarlet [16].

In this section, a few families of polynomials that can be used as prime basis in the reference elements presented in section 1.1 are defined. They are

(i) the Jacobi basis

(ii) the Legendre basis (tensorized domains)

(iii) the Dubiner basis (simplices)

(iv) the Boundary Adapted basis (tensorized domains and simplices).

In section 1.3.1, we show how to evaluate and differentiate polynomials (expressed in terms of a prime basis). We present in section 1.3.2 a numerical analysis on how to choose the prime basis. Finally, in section 1.3.3, the construction of Lagrange polynomials associated with a point set is addressed.

We recall that for $d = 1, 2, 3$, we have

$$\mathcal{T}^d = \left\{(x_1, \dots, x_d) \in \mathbb{R}^d \mid -1 < x_1, \dots, x_d < 1, \ x_1 + \dots + x_d < 0\right\}$$

and

$$\mathcal{Q}^d = \left\{(x_1, \dots, x_d) \in \mathbb{R}^d \mid -1 < x_1, \dots, x_d < 1\right\}.$$

Note that for $d = 1$, these two sets coincide with the interval $(-1, 1)$.

In what follows, we are interested in constructing bases for the space $\mathbb{P}_N(\mathcal{T}^d)$ of polynomials of total degree smaller or equal than N and the space $\mathbb{Q}_N(\mathcal{Q}^d)$ of polynomials of degree smaller or equal than N, for $d = 1, 2, 3$. We remark that the dimension of $\mathbb{P}_N(\mathcal{T}^d)$ and $\mathbb{Q}_N(\mathcal{Q}^d)$ is given by the constant $T_N + 1$.

Remark 1.3.1. *Sometimes we refer to the spaces $\mathbb{P}_N$ and $\mathbb{Q}_N$, rather than $\mathbb{P}_N(\mathcal{T}^d)$ and $\mathbb{Q}_N(\mathcal{Q}^d)$ to simplify notation.*

1.3.1 Algebraic framework for polynomials

We now describe how to evaluate and differentiate several families of high degree polynomials.

Polynomial evaluation

Let us consider a set of polynomials $\mathcal{P} = \{p_i\}_{i=0}^{M}$ expressed in a basis $\mathcal{B} = \{\phi_j\}_{j=0}^{T_N}$. If

$$p_i = \sum_{j=0}^{T_N} \alpha_{ij}\phi_j, \ i = 0, \dots, M \tag{1.14}$$

then the entry (i, j) of $C_{\mathcal{P}}$ is given by α_{ij}.

The evaluation of all the polynomials of $\mathcal{P}$ at a given set of points, say $Y = \{y_k\}_{k=0}^{K}$, is done by calculating

$$p_i(y_k) = \sum_{j=0}^{K} \alpha_{ij}\phi_j(y_k), \ i = 0, \dots, M, \ k = 0, \dots, T_N.$$

Remark 1.3.2. *This evaluation can be recast in matricial notation*

$$V(\mathcal{P}, Y) = C_{\mathcal{P}} V(\mathcal{B}, Y)$$

using the notations from section 1.2. This provides an uniform framework of evaluating polynomials, independent of reference element, basis functions or polynomial degree.

We notice also that if several polynomial sets have to be evaluated at the same point set, then we only need to calculate the matrix $V(\mathcal{B}, Y)$ once and reuse it in each evaluation. Taking into account the construction we presented, this is of particular interest when $\mathcal{B}$ is the prime basis and Y are quadrature points.

Polynomial differentiation

Let us suppose that the polynomials in $\mathcal{P}$ are defined in $\hat{\Omega} \subset \mathbb{R}^d$. For each $r = 1, 2, \dots, d$, by differentiating (1.14) in the r direction we have

$$\frac{\partial p_i}{\partial x_r} = \sum_{j=0}^{K} \alpha_{ij} \frac{\partial \phi_j}{\partial x_r}.$$

In this way, the evaluation of the derivatives of the polynomials p_i is delegated to the evaluation of the derivatives of the prime basis. If we can express the functions $\frac{\partial \phi_j}{\partial x_r}$ in the prime basis $\mathcal{B}$, then the evaluation of the derivatives reduces again to matrix-matrix multiplications.

To accomplish this “simple” calculus, we need to calculate the coefficients of $\frac{\partial \phi_j}{\partial x_r}$ with respect to the basis $\mathcal{B}$. This means that we need to determine the coefficients β_{jm} such that

$$\frac{\partial \phi_j}{\partial x_r} = \sum_{m=0}^{T_N} \beta_{jm} \phi_m.$$

Finding such coefficients is possible by considering a point set, say $Y = \{y_k\}_{k=0}^{T_N}$, and impose the equality at such points. The point set Y must satisfy the following conditions:

- the number of points must be the same as the number of basis functions in $\mathcal{B}$
- $y_k \neq y_j$, if $k \neq j$.

If such a point set is used then

$$\frac{\partial \phi_j}{\partial x_r}(y_k) = \sum_{m=0}^{K} \beta_{jm} \phi_m(y_k).$$

Let $\mathcal{D}_r$ denote the set of the derivatives of the polynomials in $\mathcal{P}$ in the r direction. Then

$$V(\mathcal{D}_r, Y) = C_{\mathcal{D}_r} V(\mathcal{B}, Y)$$

and $C_{\mathcal{D}_r}$, called *differentiation matrix*, can be calculated by solving a linear system with matrix $V(\mathcal{B}, Y)$. Here we notice the importance of the good conditioning of this matrix.

Once $C_{\mathcal{D}_r}$ is calculated, the evaluation of $\frac{\partial p_i}{\partial x_r}$ at a given point set is straightforward. In fact, even higher order derivatives are easy to calculate following this approach. Let n be a positive integer and $X = \{x_k\}_{k=0}^{M}$ a set of points. Denoting $\mathcal{D}_r^n$ as the set of the $n-th$ derivatives of the polynomials in $\mathcal{P}$ then

$$V(\mathcal{D}_r^n, X) = C_{\mathcal{P}} (C_{\mathcal{D}_r})^n V(\mathcal{B}, X).$$

1.3.2 Choices for the prime basis

The basis in which all polynomial families are expressed, the prime basis, needs to be carefully chosen. We highlight three characteristics that it should/could verify: (i) hierarchical, (ii) L^2-orthogonal and (iii) provide good conditioning for $V(\mathcal{B}, Y)$. The first property is related with the easiness to extract basis for subsets and it is relevant in the context of constructing the Raviart-Thomas element, for instance. The second property is relevant for exact integration. The third option is of crucial importance for numerical stability of these algorithms.

In the following, we present some families of polynomials that could be considered as prime basis. Some verify all three properties and some do not. In section 1.3.2 we show a numerical study of the conditioning of the generalized Vandermonde matrix and the different choices of prime basis. We will make our choice using these results as reference.

Jacobi polynomials

The Jacobi polynomials $P_k^{(\alpha,\beta)}$ of indices $\alpha, \beta > 1$ and degree $k \geqslant 0$ are a family of orthogonal polynomials in $(-1, 1)$, see Szego [84]. The orthogonality holds with respect to the inner product

$$(u, v)_{(\alpha,\beta)} = \int_{-1}^{1} u(x)v(x)(1-x)^{\alpha}(1+x)^{\beta}\, dx.$$

These are calculated easily using recurrence relations (see for instance Sherwin and Karniadakis [48]).

Moment basis

The moment basis, which is a modal one, consists in taking the canonical basis for the spaces $\mathbb{P}_N(\mathcal{T}^d)$ or $\mathbb{Q}_N(\mathcal{Q}^d)$. For the first space we consider the functions

$$m_{ij}(x_1, x_2) = x_1^i x_2^j,\ 0 \leqslant i + j \leqslant N$$

in the 2D case and

$$m_{ijk}(x_1, x_2, x_3) = x_1^i x_2^j x_3^k,\ 0 \leqslant i + j + k \leqslant N$$

for $d = 3$.

Regarding quadrilaterals and hexahedra, the spaces are constructed in an analogous way, but with the conditions $0 \leqslant i, j \leqslant N$ for $d = 2$ and $0 \leqslant i, j, k \leqslant N$ for $d = 3$.

Tensorized Legendre basis

By taking $\alpha = \beta = 0$ in the Jacobi polynomials, we define the Legendre polynomials L_k (that form a basis for $\mathbb{Q}_N(\mathcal{Q}^1)$).

The extension of a 1D basis to quadrilaterals or hexahedra is done by tensorization. If we consider d 1D bases $\{\varphi_{k_l}^{(l)}\}_{l=1}^{d}$ defined in $\mathcal{Q}^1$, the family $\{\phi_{\mathbf{k}}(\mathbf{x})\}_{\mathbf{k}}$ given by

$$\phi_{\mathbf{k}}(\mathbf{x}) = \prod_{l=1}^{d} \varphi_{k_l}^{(l)}(x_l),\ \mathbf{k} = (k_1, \ldots, k_d),\ \mathbf{x} = (x_1, \ldots, x_d),$$

is a multidimensional basis for $\mathbb{Q}_N(\mathcal{Q}^d)$. Taking the 1D Legendre basis in the previous construction, we obtain bases in 2D or 3D.

Dubiner basis

The Dubiner basis is an orthogonal modal basis defined in the reference triangle, introduced by Dubiner in [24]. Using the collapsed coordinate system (1.4) introduced in section 1.2.1, for $\mathbf{k} = (k_1, k_2)$, we define the function

$$\Phi_{\mathbf{k}}(\xi_1, \xi_2) = \psi_{k_1}(\xi_1)\psi_{k1,k2}(\xi_2),$$

where

$$\psi_{k_1}(\xi_1) = P_{k_1}^{(0,0)}(\xi_1),\quad \psi_{k1,k2}(\xi_2) = \left(\frac{1-\xi_2}{2}\right)^{k_1} P_{k_2}^{(2k_1+1,0)}(\xi_2),$$

which is a polynomial of degree k_1 in ξ_1 and $k_1 + k_2$ in ξ_2. ψ_p and $\psi_{p,q}$ are called *principal functions* of first and second order. Applying m we obtain a function defined in $\mathcal{T}^2$:

$$\varphi_{k_1,k_2}(x_1, x_2) = P_{k_1}^{(0,0)}\left(2\frac{1+x_1}{1-x_2} - 1\right)\left(\frac{1-x_2}{2}\right)^{k_1} P_{k_2}^{(2k_1+1,0)}(x_2).$$

By taking $k_1, k_2 \geqslant 0$ and $k_1+k_2 \leqslant N$, we obtain the so called 2D Dubiner basis for $\mathbb{P}_N(\mathcal{T}^2)$.

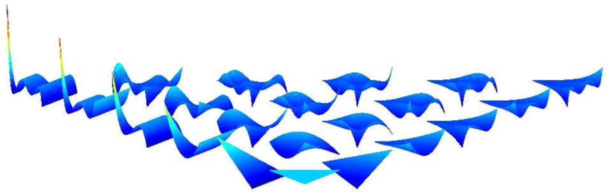


Figure 1.12: 2D Dubiner basis functions for $\mathbb{P}_5(\mathcal{T}^2)$. (Courtesy of G. Steiner)

In 3D, the construction follows the same ideas by using the coordinate mapping from $\mathcal{T}^3$ to $\mathcal{Q}^3$ defined by

$$\xi_1 = \frac{2(1+x_1)}{-x_2-x_3} - 1,\quad \xi_2 = \frac{2(1+x_1)}{1-x_3} - 1,\quad \xi_3 = x_3. \tag{1.15}$$

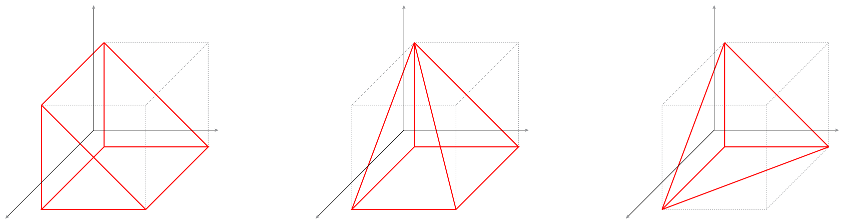


Figure 1.13: The three stages to contract the hexahedron onto the tetrahedron (from left to right).

We can see in Figure 1.13 the several stages by which the coordinate system in the hexahedra is collapsed to obtain the tetrahedron. We define the principal functions of first and second order as before and the principal functions of third order as

$$\psi_{k1,k2,k3}(\xi_3) = \left(\frac{1-\xi_3}{2}\right)^{k_1+k_2} P_{k_3}^{(2k_1+2k_2+2,0)}(\xi_3).$$

Therefore, the 3D Dubiner polynomials can be written as

$$\varphi_{k_1,k_2,k_3}(x_1,x_2,x_3) \quad = \quad \psi_{k1}(\xi_1)\psi_{k1,k2}(\xi_2)\psi_{k1,k2,k3}(\xi_3).$$

that is

$$\begin{aligned}\varphi_{k_1,k_2,k_3}(x_1,x_2,x_3) \quad &= \quad P_{k_1}^{(0,0)}\left(\frac{2(1+x_1)}{-x_2-x_3}-1\right) \\ &\cdot \quad \left(\frac{-x_1-x_3}{1-x_3}\right)^{k_1} P_{k_2}^{(2k_1+1,0)}\left(\frac{2(1+x_1)}{1-x_3}-1\right) \\ &\cdot \quad \left(\frac{1-x_3}{2}\right)^{k_1+k_2} P_{k_3}^{(2k_1+2k_2+2,0)}(x_3).\end{aligned}$$

and the set

$$\{\varphi_{k_1,k_2,k_3} \mid 0 \leqslant k_1, k_2, k_3, \quad k_1+k_2+k_3 \leqslant N\}$$

is a basis for $\mathbb{P}_N(\mathcal{T}^3)$.

Boundary Adapted basis

We define also the Boundary Adapted basis, as presented in Sherwin and Karniadakis [48], which is also a modal basis. This boundary Adapted basis allows to build a C^0 multidomain expansion, see chapter 2.

In 1D, this basis is defined by performing a modification of the Jacobi basis. The basis functions are

$$\varphi_p(x) = \begin{cases} \frac{1-x}{2}, & p = 0, \\ \left(\frac{1-x}{2}\right)\left(\frac{1+x}{2}\right)P_{p-1}^{(1,1)}(x), & 0 < p < N, \\ \frac{1+x}{2}, & p = N. \end{cases}$$

By considering the tensorized construction described in section 1.3.2, we can obtain the Tensorized Boundary Adapted basis.

Concerning the construction of this set of polynomials in the triangle, this is done in the same way as for the Dubiner set except that the principal functions are different.

For $N \in \mathbb{N}$, we define the first, second and third order boundary adapted principal functions as:

$$\psi_i(x) = \begin{cases} \frac{1-x}{2}, & i = 0, \\ \left(\frac{1-x}{2}\right)\left(\frac{1+x}{2}\right)P_{i-1}^{(1,1)}(x), & 0 < i < N, \\ \frac{1+x}{2}, & i = N, \end{cases}$$

$$\psi_{ij}(x) = \begin{cases} \psi_j(x), & i = 0, 0 \leqslant j \leqslant N, \\ \left(\frac{1-x}{2}\right)^{i+1}, & 1 \leqslant i < N, j = 0, \\ \left(\frac{1-x}{2}\right)^{i+1}\left(\frac{1+x}{2}\right)P_{j-1}^{(2i+1,1)}(x) & 1 \leqslant i < N, 1 \leqslant j < N, \\ \psi_j(x), & i = N, 0 \leqslant j \leqslant N, \end{cases}$$

$$\psi_{ijk}(x) = \begin{cases} \psi_{jk}(x), & i = 0, 0 \leqslant j \leqslant N, 0 \leqslant k \leqslant N, \\ \psi_{ik}(x), & 0 \leqslant i \leqslant N, j = 0, 0 \leqslant k \leqslant N, \\ \left(\frac{1-x}{2}\right)^{i+j+1}, & 1 \leqslant i < N, 1 \leqslant j < N, k = 0, \\ \left(\frac{1-x}{2}\right)^{i+j+1}\left(\frac{1+x}{2}\right)P_{k-1}^{(2i+2j+1,1)}(x) & 1 \leqslant i < N, 1 \leqslant j < N, 1 \leqslant k < N \\ \psi_{ik}(x), & 0 \leqslant i \leqslant N, j = N, 0 \leqslant k \leqslant N, \\ \psi_{jk}(x), & i = N, 0 \leqslant j \leqslant N, 0 \leqslant k \leqslant N, \end{cases}$$

for $x \in [-1, 1]$.

The Boundary Adapted basis is then the collapsed product of these principal functions. We list here the functions in the 2D case. For each vertex, we associate a function

$$\begin{aligned} &\text{vertex } 0: && \varphi_{0,0}(x_1, x_2) = \psi_0(\xi_1)\psi_{00}(\xi_2), \\ &\text{vertex } 1: && \varphi_{N,0}(x_1, x_2) = \psi_N(\xi_1)\psi_{N0}(\xi_2), \\ &\text{vertex } 2: && \varphi_{0,N}(x_1, x_2) = \psi_0(\xi_1)\psi_{0N}(\xi_2) + \psi_N(\xi_1)\psi_{NN}(\xi_2). \end{aligned}$$

For each edge, $(N-1)$ basis functions are associated:

$$\begin{aligned} &\text{edge } 2: && \varphi_{i,0}(x_1, x_2) = \psi_i(\xi_1)\psi_{i0}(\xi_2), && 0 < i < N, \\ &\text{edge } 1: && \varphi_{0,j}(x_1, x_2) = \psi_0(\xi_1)\psi_{0j}(\xi_2), && 0 < j < N, \\ &\text{edge } 0: && \varphi_{N,j}(x_1, x_2) = \psi_N(\xi_1)\psi_{Nj}(\xi_2), && 0 < j < N. \end{aligned}$$

Finally, the $(N-1)(N-2)/2$ interior functions are defined by

$$\varphi_{i,j}(x_1, x_2) = \psi_i(\xi_1)\psi_{ij}(\xi_2), \quad 0 < i, j, \ i + j < N.$$

The set of all the $\varphi_{i,j}$ functions just defined is a basis for $\mathbb{P}_N(\mathcal{T}^2)$.

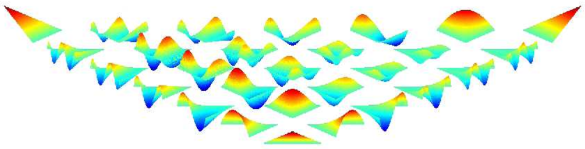


Figure 1.14: Boundary Adapted function basis in 2D for $\mathbb{P}_5(\mathcal{T}^2)$. (Courtesy of G. Steiner)

For the 3D basis, the reader can consult Sherwin and Karniadakis [48] for the vertex, edge, face and volume functions. We highlight that our numbering of vertices/edges/faces is different.

Condition number of the generalized Vandermonde matrix

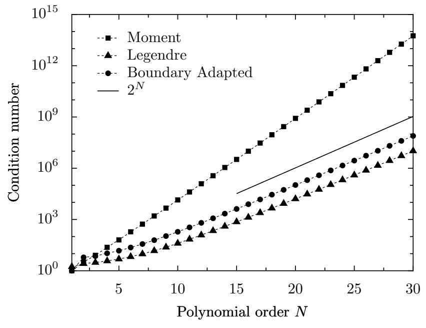
Typically, the prime basis taken to represent polynomials is the moment basis. However, other choices are obviously available: the Dubiner, Legendre or Boundary Adapted bases, depending on the reference element we are considering. We discuss now the choice of basis and point set to have the lowest condition number for the generalized Vandermonde matrix $V(\mathcal{B}, X)$. Let us start with the bases and point sets defined in the reference interval. We evaluate the Moment, Legendre and Boundary Adapted basis at the Equidistributed and Gauss-Lobatto point sets. The condition number[1] of the generalized Vandermonde matrix is plotted in Figures 1.15(a) and 1.15(b). We observe that the lowest condition numbers are obtained with the Gauss-Lobatto points. Indeed, the sub-linear growth associated with the Legendre basis and these points lead us to conclude that this is the best possible choice among those we present.

Regarding bases and point sets defined in $\mathcal{Q}^2$ and $\mathcal{Q}^3$, similar results are obtained. In Figure 1.15 we plot the condition number of $V(\mathcal{B}, X)$, for the same bases (except the Moment basis) and point sets.

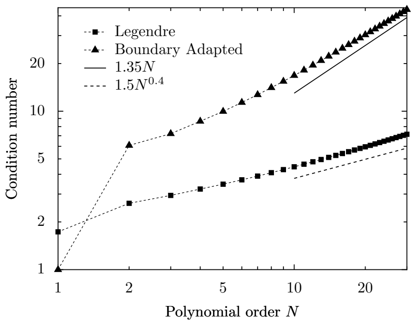
It is interesting to notice that the growth rate associated with the Boundary Adapted (and the Legendre basis) and the Gauss-Lobatto points does not differ too much between the two and three dimensional setting.

The previous results justify our choice of using the Gauss-Lobatto points to construct the differentiation matrix.

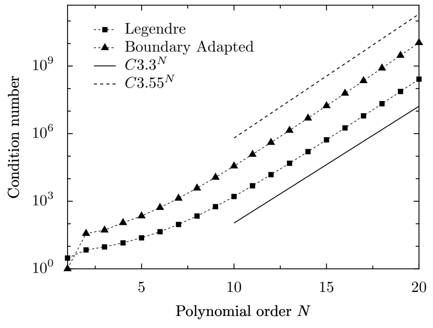
[1]The condition number was calculated with the algorithm provided by the GMM++ 1.6 library.



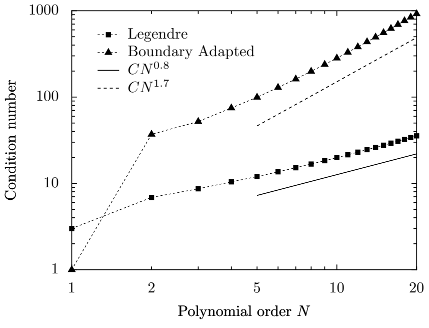
(a) Equidistributed point set.



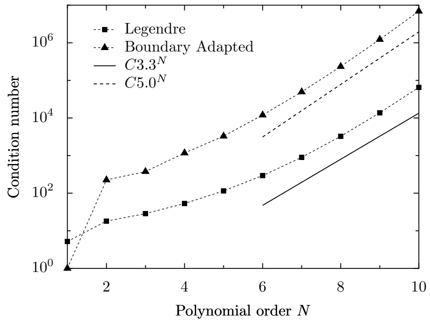
(b) Gauss-Lobatto point set.



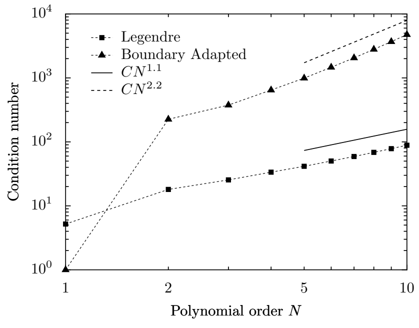
(c) Equidistributed point set.



(d) Gauss-Lobatto point set.



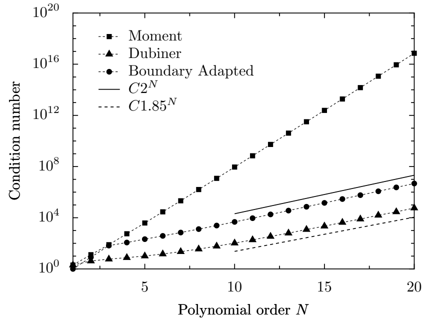
(e) Equidistributed point set.



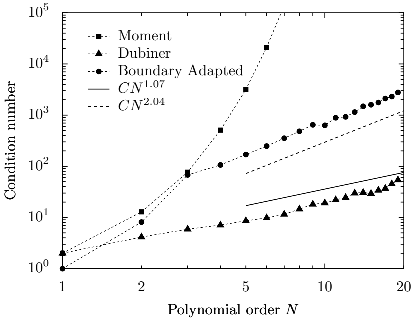
(f) Gauss-Lobatto point set.

Figure 1.15: Condition number of $V(\mathcal{B}, X)$ for $\mathcal{Q}^1$ (top), $\mathcal{Q}^2$ (middle) and $\mathcal{Q}^3$ (bottom).

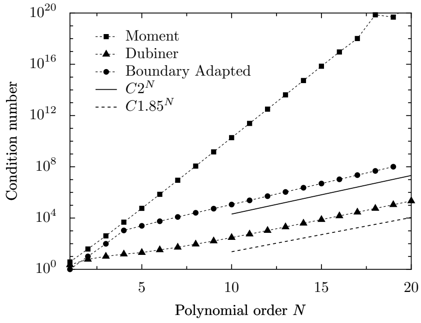
We turn now to the simplicial reference elements $\mathcal{T}^2$ and $\mathcal{T}^3$. It is also clear from Figure 1.16 that from the choices available, the Fekete points (in 2D) and the Warpblend (in 3D) should be used in the construction of the differentiation matrix. The growth of the condition number of $V(\mathcal{B}, X)$ is linear using Fekete points either with the Dubiner or Boundary Adapted two dimensional bases. In the case of the Dubiner basis, the results are comparable with the ones obtained with the Legendre basis and the Gauss-Lobatto points. We do not present a figure with the results obtained using the Warpblend points in 2D, but the growth of $\mathcal{K}(V(\mathcal{B}, X))$ using those points is exponential. On the other hand, for



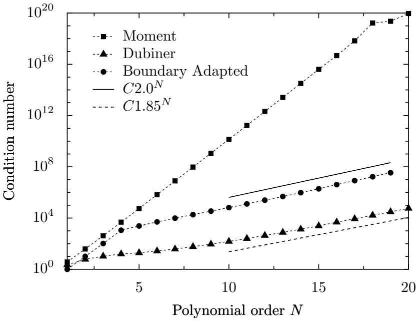
(a) Equidistributed point set.



(b) Fekete point set.



(c) Equidistributed point set.



(d) Warpblend point set.

Figure 1.16: Condition number of $V(\mathcal{B}, X)$ associated with $\mathcal{T}^2$ (top) and $\mathcal{T}^3$ (bottom).

the three dimensional case, the condition number of $V(\mathcal{B}, X)$ is lower using the Dubiner basis calculated at the Warpblend points. The growth is exponential but is about four times smaller than in the case of the Equidistributed points.

We present shortly, in Table 1.3, the choices we make for the prime basis in dD, $d = 1, 2, 3$, and for the different reference elements, and in Table 1.4, the choices for

	$\mathcal{T}^d$	$\mathcal{Q}^d$
$d = 1$	Legendre	
$d = 2$ $d = 3$	Dubiner	Tensorized Legendre

Table 1.3: Prime basis that provide the lowest condition number for $V(\mathcal{B}, X)$.

	$\mathcal{T}^d$	$\mathcal{Q}^d$
$d = 1$	Gauss-Lobatto	
$d = 2$	Fekete	Gauss-Lobatto
$d = 3$	Warpblend	Gauss-Lobatto

Table 1.4: Point set that provide the lowest condition number for the generalized Vandermonde matrix.

the point set to be used for differentiation.

1.3.3 Lagrange basis

Let us fix a polynomial degree $N > 0$. To define a Lagrange basis for $\mathbb{P}_N$ (or $\mathbb{Q}_N$) in the reference elements, we need a set of distinct points $X = \{x_i\}_{i=0}^{T_N}$. The Lagrange finite element $(\hat{\Omega}, \mathbb{P}_N, \Sigma)$ is defined by the set of linear functionals $\Sigma = \{\sigma_0, \dots, \sigma_{T_N}\}$ given by

$$\sigma_i : \mathbb{P}_N \longrightarrow \mathbb{R}, \quad \sigma_i(p) = p(x_i), \ \ \forall p \in \mathbb{P}_N.$$

Using equation (1.13) and the definition of σ_i, the Lagrange basis functions $\mathcal{L} = \{\ell_i\}_{i=0}^{T_N}$ satisfy

$$\ell_i(x_k) = \delta_{ik}, \quad \forall i, k = 0, \dots, T_N \tag{1.16}$$

If we want to express the Lagrange polynomials in terms of a prime basis $\mathcal{B} = \{\phi_j\}_{j=0}^{T_N}$, we need to calculate, for each i, the coefficients of ℓ_i in this basis. By writing

$$\ell_i(x) = \sum_{j=0}^{T_N} \alpha_{ij}\phi_j(x), \tag{1.17}$$

relations (1.16) and expansion (1.17) imply that the coefficients α_{ij} must satisfy

$$\sum_{j=0}^{T_N} \alpha_{ij}\phi_j(x_k) = \delta_{ik}, \quad i = 0, \dots, T_N$$

Therefore by evaluating the prime basis in the point set and solving the linear system

$$C_{\mathcal{L}} V(\mathcal{B}, X) = I_{T_N+1}$$

where I_{T_N+1} denotes the $(T_N+1)\times(T_N+1)$ identity matrix, the coefficient matrix $C_{\mathcal{L}}$ can be calculated.

We remark that the matrix $V(\mathcal{B}, X)$ should be well conditioned so that the coefficients of the Lagrange basis are calculated accurately. Since the points are fixed a priori, we can only decide which basis $\mathcal{B}$ to choose to represent the Lagrange polynomials in order fulfill this property. The conclusions from section 1.3.2 are still valid in this context.

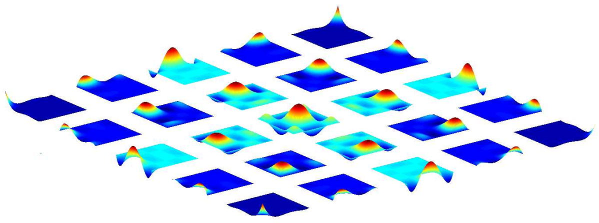


Figure 1.17: Lagrange basis of degree 4 in $\mathcal{Q}^2$.

Remark 1.3.3. *For the Lagrange basis presented, the numbering associated with the basis functions is the same as for the point set associated, see section 1.2.*

1.4 Geometrical Mapping

The connection between the reference element, which from now on we generically denote by $\hat{\Omega}$, and another domain Ω (that can be, for instance, an element of a mesh or a single domain, as is the case of the spectral method), is done by means of a mapping, $\boldsymbol{\varphi} : \hat{\Omega} \longrightarrow \Omega$, which is usually called *geometrical mapping*. Depending on the choice of $\boldsymbol{\varphi}$, this map can account for elements with curved edges/faces.

We make some assumptions on this transformation: (i) the Jacobian of $\boldsymbol{\varphi}$ is not singular, (ii) $\boldsymbol{\varphi}$ is a bijection between the $\hat{\Omega}$ and Ω and (iii) $\boldsymbol{\varphi}$ is sufficiently regular.

Our construction of the geometrical mapping is done by considering a fixed positive integer N_{geo}, two sets of nodes $\hat{X} = \{\hat{x}_i\}_{i=0}^{T_{N_{\text{geo}}}} \subset \hat{\Omega}$ and $X = \{x_i\}_{i=0}^{T_{N_{\text{geo}}}} \subset \Omega$ and a Lagrange interpolant associated with the points in $\hat{X}$. The points in X are called *geometrical nodes.* Let $\mathcal{L} = \{\ell_i\}_{i=0}^{T_{N_{\text{geo}}}}$ denote the Lagrange basis functions associated with $\hat{X}$ and $G = [g_{ij}]$ the $d \times (T_{N_{\text{geo}}} + 1)$ matrix that has the coordinates of the points of X in its columns. Then the $i-$th component of $\boldsymbol{\varphi}$ is expressed as

$$\sum_{j=0}^{T_{N_{\text{geo}}}} g_{ij}\ell_j(\hat{\mathbf{x}}), \quad \hat{\mathbf{x}} \in \hat{\Omega}. \tag{1.18}$$

for $i = 1, \ldots, d$. By expressing the geometric mapping in terms of the Lagrange polynomials defined in section 1.3.3, it inherits all the potentialities of the framework developed so far, namely, how to evaluate and differentiate $\boldsymbol{\varphi}$.

Let us denote by $\mathbf{C}_{\mathcal{L}}$ the block diagonal matrix with blocks $C_{\mathcal{L}}$ and $\mathbf{V}(\mathcal{L}, X)$ the analogous matrix but now built with blocks $V(\mathcal{L}, X)$. Then

$$\boldsymbol{\varphi}(\hat{\mathbf{x}}) = G\mathbf{V}(\mathcal{L}, \{\hat{\mathbf{x}}\}), \qquad \hat{\mathbf{x}} \in \hat{\Omega}. \tag{1.19}$$

The derivative of $\boldsymbol{\varphi}$ can now be easily calculated and we denote it by the matrix

$$K(\hat{\mathbf{x}}) = \boldsymbol{\nabla}\boldsymbol{\varphi}(\hat{\mathbf{x}}) = G\boldsymbol{\nabla}\mathbf{V}(\mathcal{L}, \{\hat{\mathbf{x}}\})$$

which can be expressed with the formulas given in section 1.3.1 in terms of matrix-matrix operations.

We define also the *pseudo-inverse* of $\boldsymbol{\nabla}\boldsymbol{\varphi}$ as

$$B(\hat{\mathbf{x}}) = K(\hat{\mathbf{x}})^{-T}.$$

In our construction, we take $\hat{X}$ as the Equidistributed point set. The mapping $\boldsymbol{\varphi}$ is then completely defined by providing the point set X in the domain Ω. There are two ways of obtaining X. The first is to use a high order mesh generator, like for instance Gmsh, see [35]. Given the geometry of the problem, it provides the geometrical nodes and $\boldsymbol{\varphi}$ is automatically defined. However, we point out that in the case of Gmsh, the Jacobian of this transformation might be singular, see section 4.3.1.

The second possibility is to use Gordon-Hall type transformations, see Gordon and Hall [36, 37]. This is the strategy we adopt to generate the coordinates of the points of X. Given a description of the boundary of Ω, we construct the Gordon-Hall transformation associated and apply it to the points of $\hat{X}$, thus obtaining the point set that we take as X. $\boldsymbol{\varphi}$ can then be seen as a Lagrange interpolant of this Gordon-Hall transformation and we remark that it conserves the orientation defined in section 1.1. In the following we present the expressions of these Gordon-Hall type transformations.

Finally, we observe that the spaces $\mathbb{P}_N(\hat{\Omega})$ and $\mathbb{Q}_N(\hat{\Omega})$ can be naturally extended to Ω by considering the image of all polynomials by $\boldsymbol{\varphi}$:

$$\mathbb{P}_N(\Omega) = \left\{ p : \; p = \hat{p} \circ \boldsymbol{\varphi}^{-1}, \; \hat{p} \in \mathbb{P}_N(\hat{\Omega}) \right\} \tag{1.20}$$

Remark 1.4.1. *When using an affine geometrical transformation, (1.20) coincides with the space of polynomials of total degree N in Ω. However, if the map is a polynomial of higher degree, this is not the case. Indeed, by looking at relation*

$$p(\boldsymbol{\varphi}(\hat{x})) = \hat{p}(\hat{x}), \quad \forall \hat{x} \in \hat{\Omega}$$

we notice that if $\boldsymbol{\varphi}$ is a polynomial of degree K, to represent polynomials of degree N exactly in $\mathbb{P}_N(\Omega)$, we need to consider, in our construction, polynomials from $\mathbb{P}_{N \cdot K}(\hat{\Omega})$.

1.4.1 One dimensional reference element

The geometrical mapping to transform $\mathcal{Q}^1$ in any interval $\Omega = [a, b]$ is given by

$$\boldsymbol{\varphi}(\xi) = \frac{(b-a)\xi + b + a}{2}, \quad \forall \xi \in \mathcal{Q}^1.$$

This map is a clear C^∞ bijection between $\mathcal{Q}^1$ and Ω.

1.4.2 Triangular reference element

Let us consider a triangular region in $\mathbb{R}^2$ with curved boundaries, Ω, as in Figure 1.18. A simple procedure exists to map a straight sided triangle into one with curved boundaries based in Gordon-Hall transformations, see Gordon and Hall [36, 37]. Let us admit that the

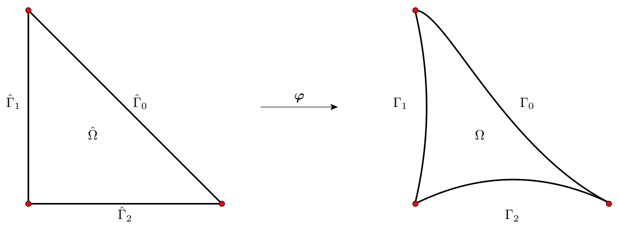


Figure 1.18: Transformation of the reference triangle.

edges of Ω, which we denote by Γ_i, are parameterized by functions $\pi_i : [-1, 1] \longrightarrow \Gamma_i$. We assume that the parameterizations are consistent with the orientation defined in section 1.2, meaning $\pi_0(-1) = \pi_2(1)$, $\pi_1(-1) = \pi_0(1)$ and $\pi_2(-1) = \pi_1(1)$.

A transformation that extends smoothly the boundary mappings to the interior of Ω can be found in Canuto, Hussaini, Quarteroni and Zang [12]. Here, we adapt it to be consistent with our orientation of the edges. The geometrical transformation has the form

$$\boldsymbol{\varphi}(\xi, \eta) = F_{aff}(\xi, \eta) + F_0(\xi, \eta) + F_1(\xi, \eta) + F_2(\xi, \eta), \quad \forall (\xi, \eta) \in \hat{\Omega}$$

where $F_{aff}(\xi, \eta)$ is the affine mapping given by

$$F_{aff}(\xi, \eta) = \frac{1+\xi}{2}\pi_2(1) + \frac{1+\eta}{2}\pi_0(1) - \frac{\xi+\eta}{2}\pi_1(1)$$

and the functions F_i are such that they vanish on the sides $\hat{\Gamma}_k$, for $k \neq i$ and are equal to π_i in the interior of $\hat{\Gamma}_i$.

For example, if $\hat{\pi}_2$ is the bubble function over $(-1, 1)$ defined as

$$\hat{\pi}_2(\xi) = \pi_2(\xi) - \frac{1-\xi}{2}\pi_2(-1) - \frac{1+\xi}{2}\pi_2(1)$$

a possible expression for F_2 is

$$F_2(\xi,\eta) = \frac{1-\eta}{2}\hat{\pi}_2(\xi) - \frac{1+\xi}{2}\hat{\pi}_2(-\eta).$$

One can easily check that $F_2(\xi,-1) = \hat{\pi}_2(\xi)$, $F_2(-1,\eta) = 0$ and $F_2(\xi,-\xi) = 0$.

Proceeding in a similar way for F_0 and F_1, we arrive at the final expression

$$\begin{aligned}\boldsymbol{\varphi}(\xi,\eta) &= \frac{1-\eta}{2}\pi_2(\xi) - \frac{1+\xi}{2}\pi_2(-\eta) + \frac{1-\xi}{2}\pi_1(-\eta) - \frac{1+\eta}{2}\pi_1(\xi) \\ &\quad + \left(1 + \frac{\xi+\eta}{2}\right)\pi_0(-\xi) - \frac{1+\xi}{2}\pi_0(-1-\xi-\eta) \\ &\quad + \frac{1+\xi}{2}\pi_2(1) + \frac{\xi+\eta}{2}\pi_1(1),\end{aligned}$$

for all $(\xi,\eta) \in \hat{\Omega}$.

1.4.3 Quadrangular reference element

Regarding the quadrilateral reference element, a transformation of the same kind can be derived using the ideas presented in the previous subsection. If we consider parameter-

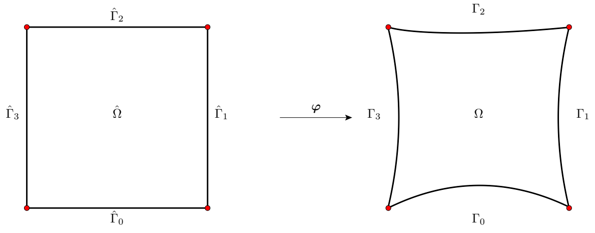


Figure 1.19: Transformation of the reference square.

izations $\pi_i : [-1,1] \longrightarrow \Gamma_i$ as in Figure 1.19 with the orientations described in section 1.2, then the following transformation can be derived

$$\begin{aligned}\boldsymbol{\varphi}(\xi,\eta) &= \frac{1-\eta}{2}\pi_0(\xi) + \frac{1+\eta}{2}\pi_2(-\xi) \\ &\quad + \frac{1-\xi}{2}\left[\pi_3(-\eta) - \frac{1+\eta}{2}\pi_2(1) - \frac{1-\eta}{2}\pi_3(1)\right] \\ &\quad + \frac{1+\xi}{2}\left[\pi_1(\eta) - \frac{1+\eta}{2}\pi_1(1) - \frac{1-\eta}{2}\pi_0(1)\right]\end{aligned}$$

for all $(\xi,\eta) \in \hat{\Omega}$.

1.4.4 Tetrahedral and hexahedral reference element

The geometrical mappings for the three dimensional reference elements can also be built using the same techniques. We refer the reader to Sherwin and Karniadakis [48] or Solin, Segeth and Dolezel [79].

1.5 Integration on a single domain

We now address how to calculate integrals of functions defined in Ω, or a part of it's boundary. The tools that we use are the geometrical mapping, see section 1.4, and a quadrature rule, see section 1.2.1.

Let $f : \Omega \longrightarrow \mathbb{R}$ be a function and let us denote again $\boldsymbol{\varphi} : \hat{\Omega} \longrightarrow \Omega$. We wish to calculate

$$\int_{\Omega} f(x)\ dx. \tag{1.21}$$

By performing a change of variable using the geometrical mapping $\boldsymbol{\varphi}$ in the previous integral, we obtain

$$\int_{\hat{\Omega}} f(\boldsymbol{\varphi}(\hat{x})) J_{\boldsymbol{\varphi}}(\hat{x})\ d\hat{x}. \tag{1.22}$$

Let $Q_{\hat{\Omega}} = \{(\hat{x}_i, w_i)\}_{i=0}^{K}$ denote a quadrature formula in $\hat{\Omega}$. Then (1.22) can be approximated by

$$\sum_{i=0}^{K} f(\boldsymbol{\varphi}(\hat{x_i})) J_{\boldsymbol{\varphi}}(\hat{x_i}) w_i.$$

We highlight that in the case f is a polynomial of $\mathbb{P}_N(\Omega)$ or $\mathbb{Q}_N(\Omega)$ (see 1.4), then (1.21) can be calculated exactly, either by exact or numerical integration. The first approach takes advantage of the fact that the polynomials are expressed in terms of the prime basis, as well as it's Jacobian. The second uses a quadrature formula with enough points so that the integral is calculated exactly.

In a similar way we calculate integrals over subsets $\Gamma \subset \Omega$ such that $\Gamma = \boldsymbol{\varphi}(\hat{\Gamma})$ where $\hat{\Gamma}$ is a face of the reference element. Let $f : \Gamma \longrightarrow \mathbb{R}$ denote now a function defined in the face Γ. Then

$$\int_{\Gamma} f\ ds = \int_{\hat{\Gamma}} f(\boldsymbol{\varphi}(\hat{s}))\, \|\mathbf{n}_{\Gamma}\|\, J_{\boldsymbol{\varphi}}(\hat{s})\ d\hat{s} = \int_{\hat{\Gamma}} f(\boldsymbol{\varphi}(\hat{s}))\, \|B(\hat{s})\mathbf{n}_{\hat{\Gamma}}\|\, J_{\boldsymbol{\varphi}}(\hat{s})\ d\hat{s}$$

where $\mathbf{n}_{\Gamma}$ and $\mathbf{n}_{\hat{\Gamma}}$ denote the outward normals to Γ and $\hat{\Gamma}$, respectively. The matrix B is defined as in section 1.4. The evaluation of this integral is done with an appropriate quadrature formula, see section 1.2.1.

In the case f is a vectorial function, say $f : \Gamma \longrightarrow \mathbb{R}^d$, then

$$\int_{\Gamma} f \cdot \mathbf{n}_{\Gamma}\ ds = \int_{\hat{\Gamma}} f(\boldsymbol{\varphi}(\hat{s})) \cdot (B(\hat{s})\mathbf{n}_{\hat{\Gamma}})\, J_{\boldsymbol{\varphi}}(\hat{s})\ d\hat{s}.$$

1.6 Conclusion

In this chapter, a general framework is proposed to evaluate, differentiate and integrate polynomial bases for the spectral method in dD, $d = 1, 2, 3$. By construction, we use a basis to represent all polynomial families and delegate the latter operations into the prime basis.

In the framework presented, the Legendre/Dubiner bases are the most well suited among the shown families, to be prime bases.

The choice of a point set to construct Lagrange basis, the Fekete and Warpblend points revealed to have the lowest Lebesgue constants, in 2D and 3D (note that using a hexahedron as reference geometry, the Fekete points are the ones to choose in this matter). This property makes them suitable as interpolation point sets. The same conclusions are valid for chosing points to numerically differantiate the polynomial bases.

Regarding the two dimensional framework, it can be further extended by using the Fekete points up to order 30. These are calculated in Roth [73] and should be considered when using higher degree polynomials.

The work presented in this chapter can and should be extended also to other reference elements such as prisms and pyramids, in the 3D case. The prime bases functions for these reference elements can be found in Sherwin and Karniadakis [48].

In the matter of interpolation and differentiation in 3D simplicial geometries, a better point set should be used. We considered the Warpblend point set, but the growth of the Lebesgue constant is not satisfactory. Ideally, Fekete points should be used, but they are not known in 3D.

Finally, the high order geometrical mappings in 2D should be extended to 3D, see Solin [79].

Chapter 2

The Spectral Element Method

So far, we have seen how to construct, evaluate, differentiate and integrate polynomial sets defined in a single domain. The goal of this chapter is to define polynomial sets of arbitrary order defined in an domain partitioned in several elements. The construction is based on the fact that each element is the image by a geometrical transformation of the type described in section 1.4. Using this transformation, the polynomial bases defined in the previous chapter are then constructed in this element.

We analyze the construction of two types of global expansions: continuous and discontinuous. The first produces basis functions that are globally continuous, while the second produces an expansion where the basis functions are continuous within each element of the partition and discontinuous across the faces. These basis functions are suitable for the continuous and discontinuous Galerkin (dG) methods, respectively. To build the continuous expansion, special attention is given to glue the basis functions between neighbor elements. This constraint is not present in the construction of the discontinuous expansion. However, when integrating a function (that might take two values) over a face, we have the problem of matching quadrature points coming from the different elements that share that face. This issue is circumvented in 2D using the same construction done in the continuous setting.

This chapter is organized as follows: in the first section we describe how to construct basis functions in a domain partitioned in several elements, in the continuous and discontinuous setting just presented; the second section is dedicated to the assembly of vectors and matrices arising from a Galerkin formulation and integration over faces in the dG framework; section 2.3 introduces an operator which provides a tool for visualization of arbitrary order polynomials, in dD. In the last section, we analyze the properties of the method in the context of a Poisson problem, namely the spectral accuracy of the method, the eigenvalues and condition number of the matrix arising from the Galerkin approximation and the mass matrix associated with the bases.

2.1 Multidomain expansions

Let us consider a domain $\Omega \subset \mathbb{R}^d$, $d = 1, 2, 3$, and a partition $\mathcal{T}_{h,N_{\text{geo}}}$ (h is the maximum diameter of all elements in the partition) of Ω in the union of N_{el} subdomains Ω_e satisfying the following assumptions:

- $\overline{\Omega} = \bigcup_{e=1}^{N_{el}} \overline{\Omega_e}$
- $\Omega_e \cap \Omega_i$ is empty whenever $e \neq i$
- two neighbor subdomains can only share vertices, edges or faces[1]
- Ω_e is the image of one of the reference elements, see section 1.1, by a geometrical mapping, say $\boldsymbol{\varphi}_e : \hat{\Omega} \longrightarrow \Omega_e$, of degree N_{geo}, see section 1.4. We assume that all geometrical transformations have the same polynomial degree.

Depending on the topological dimension, each subdomain is either an interval, a triangle, a quadrangle, a tetrahedron or a hexahedron with possibly curved boundaries.

The extensions to the multidomain case of the polynomial space $\mathbb{P}_N(\Omega_e)$, defined in (1.20), (and $\mathbb{Q}_N(\Omega_e)$) can be stated as

$$\mathcal{F}_N(\mathcal{T}_{h,N_{\text{geo}}}) = \left\{ v \in C^0(\overline{\Omega}) : \; v_{|\Omega_e} \in \mathbb{P}_N(\Omega_e), \; \forall \Omega_e \in \mathcal{T}_{h,N_{\text{geo}}} \right\} \tag{2.1}$$

and

$$\mathcal{F}_N^{\text{disc}}(\mathcal{T}_{h,N_{\text{geo}}}) = \left\{ v \in L^2(\Omega) : \; v_{|\Omega_e} \in \mathbb{P}_N(\Omega_e), \; \forall \Omega_e \in \mathcal{T}_{h,N_{\text{geo}}} \right\}. \tag{2.2}$$

To simplify the notation, we define the parameter $\delta = (h, N_{\text{geo}})$ and write $\mathcal{T}_\delta$ whenever we want to designate $\mathcal{T}_{h,N_{\text{geo}}}$. If δ is replaced only by h, then it is subjacent that the geometrical transformation is of degree 1, that is $N_{\text{geo}} = 1$.

Remark 2.1.1. *Regarding the notation for the triangulations, we will not make a difference between using triangulations with simplices of simplex products. This difference will be however pointed out each time there's ambiguity of what type of geometrical elements are being used.*

We describe how to construct a basis for $\mathcal{F}_N^{\text{disc}}(\mathcal{T}_\delta)$ and $\mathcal{F}_N(\mathcal{T}_\delta)$. Let $\{\hat{\phi}_i\}_{i=0}^{T_N}$ be the basis functions associated with $\hat{\Omega}$ and $\{\hat{\phi}_i \circ \boldsymbol{\varphi}_e^{-1}\}_{i=0}^{T_N}$ the corresponding functions in Ω_e. We start by extending each function $\hat{\phi}_i \circ \boldsymbol{\varphi}_e^{-1}$ to the whole domain Ω

$$\phi_{e,i}(\mathbf{x}) = \begin{cases} \hat{\phi}_i\left(\boldsymbol{\varphi}_e^{-1}(\mathbf{x})\right), & \mathbf{x} \in \Omega_e \\ 0, & \text{otherwise} \end{cases} \tag{2.3}$$

[1]The geometrical entities shared by subdomains depends on the topological dimension of Ω: in 1D, only vertices are shared, in 2D, vertices and edges can belong to neighbor elements, and in 3D all three types of geometrical entities are allowed to be shared.

Discontinuous space $\mathcal{F}_N^{\mathbf{disc}}(\mathcal{T}_\delta)$. Using the functions given by (2.3), we can develop a polynomial $u_N \in \mathcal{F}_N^{\mathrm{disc}}(\mathcal{T}_\delta)$ as

$$u_N(\mathbf{x}) = \sum_{e=1}^{N_{el}} \sum_{i=0}^{T_N} \hat{u}_{e,i}^{\mathrm{loc}} \phi_{e,i}(\mathbf{x}),$$

for $\mathbf{x} \in \Omega$. The coefficients $\hat{u}_{e,i}^{\mathrm{loc}}$ are called *local degrees of freedom.* If we consider global functions that are discontinuous across the element's boundary, then a basis for $\mathcal{F}_N^{\mathrm{disc}}(\mathcal{T}_\delta)$ is directly given by the set of functions $\{\phi_{e,i}\}$.

We remark that in the discontinuous Galerkin framework, continuity is not enforced at the level of the basis functions, but is done weakly at the level of the variational formulation.

Continuous space $\mathcal{F}_N(\mathcal{T}_\delta)$. It is clear that the boundary modes included in (2.3) associated with each element (functions that do not have compact support in Ω_e) are not continuous in Ω and that the interior modes, which are zero on $\partial\Omega_e$ are naturally extended to global continuous functions.

To build a continuous global expansion, the matching of similar boundary functions that are defined in neighbor elements is required. This matching depends on the orientation of the subentities that compose the reference element. Notice that a boundary adapted or nodal basis should be used to construct multidomain expansions, otherwise it is impossible to ensure the continuity in the global basis functions.

From the implementation point of view, the matching between neighbor basis functions can be described with a mapping array $n(e,i)$ that contains the global index of the ith local index in the element Ω_e. We have to identify the neighbor basis functions and assign them the same global index. This means that if $\hat{u}_{e,i}^{\mathrm{loc}}$ and $\hat{u}_{k,j}^{\mathrm{loc}}$ are coefficients of basis functions that should be glued together, then $n(e,i) = n(k,j)$. The polynomial u_N now reads as

$$u_N(\mathbf{x}) = \sum_{e=1}^{N_{el}} \sum_{i=0}^{T_N} \hat{u}_{n(e,i)}^{\mathrm{glo}} \phi_{e,i}(\mathbf{x}),$$

for $\mathbf{x} \in \Omega$. The coefficients $\hat{u}_{n(e,i)}^{\mathrm{glo}}$ are called *global degrees of freedom.*

Remark 2.1.2. *In the discontinuous setting, the sets of local and global degrees of freedom have the same cardinality.*

In the nodal case, only the map $n(\cdot,\cdot)$ is necessary in the assembly process. However, when using a modal expansion, to properly match neighbor degrees of freedom, we need to negate some of the modes in the expansion. For now, we will just introduce a map, similar to $n(\cdot,\cdot)$, say $\mathrm{sign}(\cdot,\cdot)$, that given the element and the local to the element index of the degree of freedom, returns the sign with which the mode should be multiplied, therefore, $+1$ or -1.

We refer the reader to Sherwin and Karniadakis [48] for a complete description on how to build the maps $n(\cdot,\cdot)$ and $\mathrm{sign}(\cdot,\cdot)$. In the following, we explain how to identify neighbor functions as well as provide the necessary algorithms to determine the permutations to be applied to the local degrees of freedom.

2.1.1 2D connectivity

To match the degrees of freedom (degree of freedom) associated with the edges of the element, the key ideas are the orientation and numbering of the reference element and the geometrical transformation $\boldsymbol{\varphi}_e$, see chapter 1. We point out that $\boldsymbol{\varphi}_e$ preserves the numbering and orientation of the subentities in the elements.

To make the exposure of the matching easier, we suppose that the elements are oriented in an anticlockwise fashion, as in Figure 2.1. When this is not the case, the algorithms to determine the orientation of the edges of the triangulation presented in section 2.1.2 are valid and can be applied in the 2D case.

Nodal basis functions. The ingredients of this construction are the maps that allow to pass between the global/local indices of the points in the edges of the element, as well as the permutation (in this case a simple reversion) in the local indices of the nodes. In this case, we only need to reverse the order of the local degrees of freedom in one of the edges, since the local degrees of freedom are ordered according to the edge's orientation. Let us take Figure 2.2 as example: in order to match the functions defined in both elements, degree of freedom number 3 in the upper element must correspond to degree of freedom number 10 in the lower and so on. Using the decomposition into subentities of the reference element

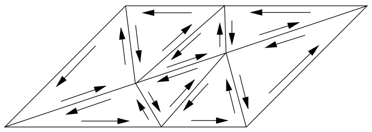


Figure 2.1: Orientation of the edges in a two dimensional mesh of triangles.

and the point set, we know, for a given edge and point in that edge, how to relate the local (w.r.t. the edge) index of the point with its index in the set of points of the whole element, i.e., the indices of the local degree of freedom . If we denote by e_1 and e_2 the element's indices and edge_1 and edge_2 the corresponding local to the element indices of the edges, then

$$n(e_1, \texttt{edgeToPoint}(\text{edge}_1, i-1)) = n(e_2, \texttt{edgeToPoint}(\text{edge}_2, P(i-1))),$$

for $i = 1, \ldots, N-1$, where P is a permutation such that $P(i) = N - 2 - i$.

Modal basis functions. If a modal basis is used in the construction, the situation is slightly different. Since the local numbering follows the polynomial degree of the functions, there is no need to invert the order of the local modes. However, due to the possible inverse edge orientation, the odd degree modes associated with the edge will have opposite signs in neighbor elements. Therefore, we negate these modes in one of the edges, that is, we

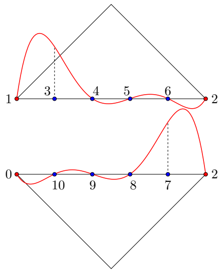


Figure 2.2: Permutation needed in the local degrees of freedom to match neighbor edge associated functions.

set the sign function equal to +1 in all the modes in both edges and −1 in only the odd modes of one edge. Regarding the interior degrees of freedom, the bubble functions have compact support in the elements and they do not need to be taken care of.

2.1.2 3D connectivity

The problem in matching functions defined in edges or faces in 3D is more complex than in the 2D case. For edges, we need to determine the permutation to be used in the local indices of the points since it is impossible to have always opposite orientations by default. An algorithm to determine which permutation to apply to the edges is used, which we describe briefly. Since the same algorithm is used for determining the permutation for the faces, we will describe it in terms of entity which can either be edge or face.

Algorithm 2.1 Edge permutation determination.

```
if indices match then
    assign identity permutation
else
    assign reverse permutation
end if
```

Let us assume that the set of all possible permutations is known for each entity[2]. We start by creating two empty vectors: v_{index} which stores the global indices (in the mesh) of the entities whose permutations have been assigned and v_{tuple} that stores the global indices of the vertices that define the entity (ordered entity local indices). The full procedure is then given by Algorithm 2.2.

We remark that point (*) is what makes the difference between dealing with faces or edges in 3D. For edges, this line should be replaced by Algorithm 2.1.

[2]This set will be specified later on for quadrangular and triangular faces.

Algorithm 2.2 Procedure to determine the permutation for edges and faces in 3D.

```
for $e = 1 \dots N_{el}$ do
  for $p = 1 \dots N_{\text{entity}}$ do
    determine global index of the entity in the mesh, $g_p$
    if $g_p$ not found in $v_{index}$ then
      add $g_p$ to $v_{index}$
      add entity global indices to $v_{tuple}$ (ordered by local index)
      assign identity permutation
    else
      determine the global indices of the vertices that define the entity
      compare the previous indices with the ones in $v_{tuple}$
      depending on the previous comparison, assign a permutation to the entity (*)
    end if
  end for
end for
```

For faces, due to the possible local coordinate systems, there are more possible permutations. We note that this number is different between triangular and quadrangular faces. We shall specify now those permutations for both cases.

Triangular face match. For triangular faces there are six possible permutations. In this context, point (*) is replaced by Algorithm 2.3.

Algorithm 2.3 Face permutation algorithm.

```
if all indices match then
  assign identity permutation
else
  if only one index matches (**) then
    assign swap permutation
  else
    assign rotation permutation
  end if
end if
```

It is clear that depending on the index that matches in Point (**) of Algorithm 2.3, we choose a different permutation: using the notation of Figure 2.3, if the first index matches, then σ_1 is assigned; if the second index matches, assign σ_3 and finally, if the third index matches, assign σ_2. A similar procedure is applied for the rotation permutations. The six possible permutations are shown in Figure 2.3 for a particular case. We remark that the set of these permutations is a group $\mathcal{S}_{\mathcal{T}}$ that can be generated by base and height swap, that is, σ_2 and σ_1:

$$\sigma_4 = \sigma_1\sigma_2, \quad \sigma_3 = \sigma_4\sigma_1, \quad \sigma_5 = \sigma_2\sigma_1$$

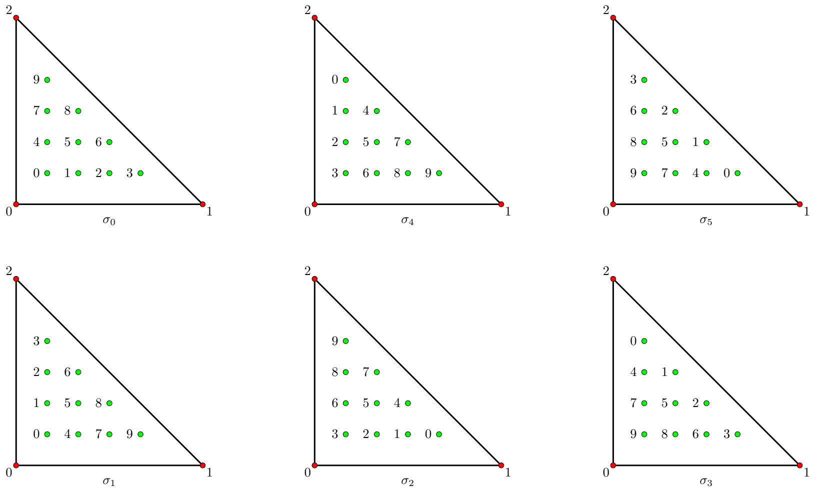


Figure 2.3: Example of the permutations for the local degrees of freedom. The indices 0, 1 and 2 denote the local face vertex indices.

With these permutations, we now know exactly how to order the local degrees of freedom in a face for a nodal basis to properly match the functions.

When analyzing the same problem for a modal basis, we notice that a renumbering of the tetrahedron vertices needs to be done. Because a collapsed coordinate system is used and the basis may not be rotationally symmetric, we must enforce that all tetrahedra are oriented like Figure 2.4 a) and not like 2.4 b). To accomplish this, we follow an

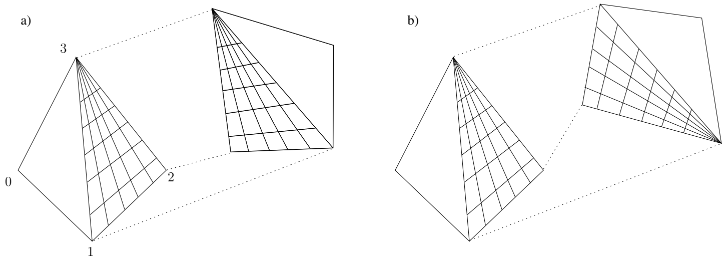


Figure 2.4: Orientation for neighbor tetrahedra for a modal basis.

idea of Warburton [89]. Referring to ($\xi_1 = -1, \xi_2 = -1, \xi_3 = 1$) as local top vertex and ($\xi_1 = -1, \xi_2 = 1, \xi_3 = -1$) as local base vertex, we reorder the vertex local indices according to Algorithm 2.4. Although it is not necessary for matching nodal basis, this reordering

Algorithm 2.4 Local renumbering algorithm.

```
place the local top vertex at the global vertex with the lowest global index
place the local base vertex at the global vertex with the second lowest global index
orient the last two vertices in such a way that the anticlockwise orientation of the indices
is preserved
```

reduces the number of possible non-identity permutations for the faces, and thus simplifies the construction of the degree of freedom table construction as well as face integration.

Quadrangular face match

For quadrangular faces, the algorithm for determining the permutations for each face is similar to 2.3. The group of all possible permutations of the vertices of the quadrilateral

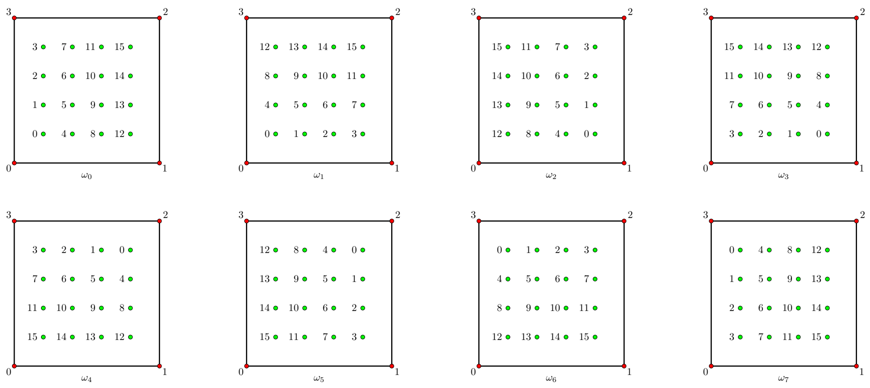


Figure 2.5: Example of the permutations for the local degrees of freedom in a hexahedron.

is of order eight and, using notation from Figure 2.5, it can be generated with ω_2 and ω_3:

$$\omega_1 = \omega_2\omega_3, \quad \omega_7 = \omega_1\omega_3, \quad \omega_5 = \omega_7\omega_2, \quad \omega_4 = \omega_7\omega_3, \quad \omega_6 = \omega_3\omega_5$$

2.2 Global assembly of matrices and face integration

Let us consider a bilinear form $a(\cdot,\cdot)$ and a linear form $F(\cdot)$ arising from the weak formulation of some differential equation. The assembly process of the linear system matrix and right hand side vector corresponding to $a(\cdot,\cdot)$ and $F(\cdot)$ are described in algorithm 2.5.

Algorithm 2.5 Assembly procedure.

```
$A(i,j) = 0, \ \forall i,j$
$F(i) = 0, \ \forall i$
for $e = 1 \dots N_{el}$ do
  for $p = 1 \dots N_{\text{entity}}$ do
    for $q = 1 \dots N_{\text{entity}}$ do
      $A(n(e,p), n(e,q)) = A(n(e,p), n(e,q)) + \text{sign}(e,p)\text{sign}(e,q)a(\phi_{e,p}, \phi_{e,q})$
    end for
    $F(n(e,p)) = F(n(e,p)) + \text{sign}(e,p)F(\phi_{e,p})$
  end for
end for
```

The integrals $a(\phi_{e,p}, \phi_{e,q})$ and $F(\phi_{e,p})$ are defined in the element Ω_e and are calculated as in section 1.5.

We describe now how to evaluate a certain type of integrals that appear in the discontinuous Galerkin method or whenever we need to integrate discontinuous functions that have a jump across the interfaces between elements. Let $u, v \in \mathcal{F}_N^{\text{disc}}(\mathcal{T}_{h,N_{\text{geo}}})$ and Γ be an internal face of the triangulation shared by two elements, say Ω_R and Ω_L. We denote by u_R and u_L the trace of u in element Ω_R and Ω_L, respectively. Similar definitions are valid for v. We denote by $\boldsymbol{\varphi}_R$ and $\boldsymbol{\varphi}_L$ the respective geometrical mappings associated with each element.

We are interested in calculating integrals of the form

$$\int_\Gamma u_R v_L \, ds \tag{2.4}$$

where the functions in the integral are defined in different elements that share Γ.

Let us fix an element a priori, say Ω_R. We extend the function v_L defined in Ω_L to Ω_R by

$$\tilde{v}_L = v_L \circ \boldsymbol{\varphi}_L \circ \boldsymbol{\varphi}_R^{-1}.$$

Notice that $\tilde{v}_L$ is defined in Ω_R and it coincides with v_L in Γ. Now, integral (2.4) is rewritten as

$$\int_\Gamma u_R \tilde{v}_L \, ds.$$

To calculate this integral, we now apply the procedures from section 1.5.

If $Q_{\hat{\Gamma}_R} = \{(\hat{s}_i, w_i)\}_{i=0}^K$ is a quadrature formula defined on the face $\hat{\Gamma}_R$ then,

$$\sum_{i=0}^K u_R(\boldsymbol{\varphi}_R(\hat{s}_i)) \tilde{v}_L(\boldsymbol{\varphi}_R(\hat{s}_i)) \left\| B_R(\hat{s}_i) \mathbf{n}_{\hat{\Gamma}_R} \right\| J_{\boldsymbol{\varphi}_R}(\hat{s}_i) w_i. \tag{2.5}$$

Our goal now is to rewrite $\tilde{v}_L(\boldsymbol{\varphi}_R(\hat{s}_i))$ using the geometrical mapping $\boldsymbol{\varphi}_L$ and v_L. We introduce another quadrature formula, now in $\hat{\Gamma}_L$, $Q_{\hat{\Gamma}_L} = \{(\hat{q}_i, z_i)\}_{i=0}^{K}$. We notice that if $x_i = \boldsymbol{\varphi}_R(\hat{s}_i)$ then

$$\tilde{v}_L(\boldsymbol{\varphi}_R(\hat{s}_i)) = \tilde{v}_L(x_i) = v_L(x_i).$$

Therefore, if the quadrature points in $Q_{\hat{\Gamma}_L}$ are such that

$$x_i = \boldsymbol{\varphi}_L(\hat{q}_i) = \boldsymbol{\varphi}_R(\hat{s}_i) \tag{2.6}$$

then (2.5) is equivalent to

$$\sum_{i=0}^{K} u_R(\boldsymbol{\varphi}_R(\hat{s}_i)) v_L(\boldsymbol{\varphi}_L(\hat{q}_i)) \left\| B_R(\hat{s}_i)\mathbf{n}_{\hat{\Gamma}_R} \right\| J_{\boldsymbol{\varphi}_R}(\hat{s}_i) w_i. \tag{2.7}$$

Once the quadrature points are generated in the face of the reference elements and according to the face's orientation, it is not given that condition (2.6) is satisfied. In fact, this condition is equivalent to the problem of glueing nodal basis functions between neighbor elements. Therefore, we can apply here the algorithm described in section 2.1 to have the proper matching of the points in Γ, thus satisfying (2.6). We remark that in 3D, we should be careful with the permutations to be applied to the quadrature points in order to do the matching. Whether the quadrature points are obtained using Gauss type or optimal quadratures, the permutations to be applied have to be devised for each case. Another, more expensive alternative, is to generate quadrature formulas directly in the face of the mesh.

2.3 Visualization of polynomials of arbitrary degree

We now describe an operator which is useful in visualizing meshes with curved elements and functions of (2.1) without losing too much information. We will present this construction in the context of nodal basis and for simplicial elements. The extension to modal basis or quadrangular elements is straightforward. We introduce the following interpolation op-

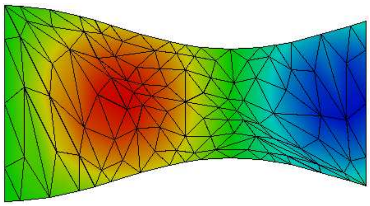


Figure 2.6: Visualization of a function using the operator $\Pi^{\mathbb{P}_1}$. The mesh $\mathcal{T}_{\tilde{h},1}$ is represented in black.

erator

$$\Pi^{\mathbb{P}_1} : \mathcal{F}_N(\mathcal{T}_{h,N_{\text{geo}}}) \mapsto \mathcal{F}_N(\mathcal{T}_{\tilde{h},1}) \tag{2.8}$$

where the triangulation associated to $\mathcal{F}_N(\mathcal{T}_{\tilde{h},1})$, a space spanned by a first degree Lagrange polynomial basis using a first degree geometric approximation, is constructed from the points associated to the degrees of freedom of $\mathcal{F}_N(\mathcal{T}_{h,N_{\text{geo}}})$. The cornerstone in the construction of $\Pi^{\mathbb{P}_1}$ is the triangulation $\mathcal{T}_{\tilde{h},1}$. This is done in two steps: first we create a triangulation, $\hat{\mathcal{T}}$, in $\hat{\Omega}$ whose vertices are the points associated with the nodal basis in the reference element; second, for each element of $\mathcal{T}_{h,1}$, we apply the corresponding geometrical mapping to $\hat{\mathcal{T}}$. This induces a triangulation in Ω that we take as $\mathcal{T}_{\tilde{h},1}$. Notice that the construction is valid in dD, $d = 1, 2, 3$. We observe that the points on $\partial\mathcal{T}_{h,N_{\text{geo}}}$ are located,

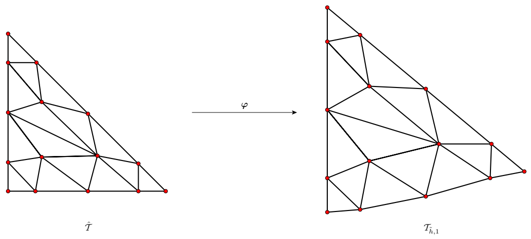


Figure 2.7: On the left, the triangulation $\hat{\mathcal{T}}$ in the reference element; on the right, the image of $\hat{\mathcal{T}}$ using the geometrical transformation, $\mathcal{T}_{\tilde{h},1}$.

thanks to the geometrical transformation $\boldsymbol{\varphi}$, exactly on $\partial\mathcal{T}_{\tilde{h},1}$ thus retaining a good approximation of the boundary. Also, this operator retains the continuity of the original space, as well as the dimension, ie,

$$\dim\left(\mathcal{F}_N(\mathcal{T}_{\tilde{h},1})\right) = \dim\left(\mathcal{F}_N(\mathcal{T}_{h,N_{\text{geo}}})\right).$$

We point out that this operator has been analysed in literature for preconditioning purposes. This stems from the fact that if $v \in \mathcal{F}_N(\mathcal{T}_{h,N_{\text{geo}}})$, $\Pi^{\mathbb{P}_1}(v)$ has the same nodal values on the mesh associated to $\mathcal{F}_N(\mathcal{T}_{\tilde{h},1})$ as v. For more details, see Canuto, Hussaini, Quarteroni and Zang [12, 11] and Deville, Fischer and Mund [20]. We will also consider this kind of preconditioner in section 3.4.

2.4 A Poisson test problem

We now apply the Galerkin spectral element method to a simple Poisson test case. This analysis will be carried out for one, two and three dimensional domains.

Let Ω be one of the reference elements, that is, $\Omega = \mathcal{T}^d$ or $\Omega = \mathcal{Q}^d$, for some $d \in \{1, 2, 3\}$. We consider the following Poisson problem

$$\begin{aligned} -\Delta u &= f, \quad \text{in } \Omega \\ u &= g, \quad \text{on } \partial\Omega \end{aligned} \tag{2.9}$$

where f and g are determined so that the exact solution of the problem is $u(x) = \exp(x)$ if $d = 1$, $u(x, y) = \exp(x + y)$, if $d = 2$ and $u(x, y, z) = \exp(x + y + z)$, if $d = 3$.

We define our functional setting

$$H^1_g = \left\{ v \in H^1(\Omega) : \; v_{|\partial\Omega} = g \right\}$$

and denote $H^1_0(\Omega)$ by taking in the previous space $g = 0$.

The weak formulation of problem (2.9) reads as

Problem 2.4.1. *Find $u \in H^1_g(\Omega)$ such that*

$$\int_\Omega \nabla u \cdot \nabla v \; dx = \int_\Omega fv \; dx, \;\; \forall v \in H^1_0(\Omega)$$

Let us present the numerical method used to approximate the solution u of (2.9). Let $N > 0$ be an integer and $h > 0$ be a real. We consider $\mathcal{T}_h$, a triangulation where the maximum diameter of all elements is smaller than h, having N_{el} elements and the space $\mathcal{F}_N(\mathcal{T}_h)$, as defined in section 2.1. If $\{\phi_i\}_{i=0}^K$ denotes a basis for $\mathcal{F}_N(\mathcal{T}_h)$, then, by putting

$$u_N = \sum_{i=0}^K \hat{u}_i \phi_i$$

the numerical method reads:

Problem 2.4.2. *Find $u_N \in \mathcal{F}_N(\mathcal{T}_h)$ such that*

$$\sum_{i=0}^K \hat{u}_i a(\phi_i, \phi_j) = F(\phi_j), \;\; \forall j = 0, \ldots, K. \tag{2.10}$$

which is equivalent to solve the linear system

$$A^{exp}\mathbf{u} = \mathbf{f}$$

where $A^{exp}_{ij} = a(\phi_i, \phi_j)$ and $\mathbf{u}$ and $\mathbf{f}$ contain the components $\hat{u}_i$ and $F(\phi_j)$, respectively.

Remark 2.4.1. *We deal with the imposition of the boundary equations at an algebraic level, that is, for each degree of freedom whose corresponding function has nonzero trace on the boundary of Ω, we replace the line of A^{exp} corresponding to such degree of freedom with zeros and put the value 1 in the diagonal entry; also, we replace the corresponding entry in vector $\mathbf{f}$ by the proper value given by the boundary condition. In practice, this means that we discretize the original differential problem in $H^1(\Omega)$ and then fix the degrees of freedom on the boundary at the algebraic level. This is the reason why the space $\mathcal{F}_N(\mathcal{T}_h)$ (that accounts for functions with nonzero trace on the boundary) is used to solve the discrete problem* (2.10).

Let $\Lambda = \{0, \dots, K\}$ and $\Lambda^* \subset \Lambda$ denote the set of indices corresponding to degrees of freedom where Dirichlet boundary conditions are imposed. Then, Problem 2.4.2 is recast as

Problem 2.4.3. *Find $u_N \in \mathcal{F}_N(\mathcal{T}_h)$ such that*

$$\sum_{i=0}^{K} \hat{u}_i a(\phi_i, \phi_j) = F(\phi_j), \ \forall j \in \Lambda \backslash \Lambda^* \tag{2.11}$$

$$\hat{u}_i = g_i, \forall i \in \Lambda^* \tag{2.12}$$

where g_i corresponds to the value of the function g calculated at the node i.

This is the problem that we are going to solve in this section. However, Problem 2.4.3 induces a linear system similar to $A^{exp}\mathbf{u} = \mathbf{f}$, say $A\mathbf{u} = \widetilde{\mathbf{f}}$, where A and $\widetilde{\mathbf{f}}$ are calculated respectively from A^{exp} and $\mathbf{f}$ following remark 2.4.1. We shall call matrix A, the *stiffness matrix.*

We define the *iterative condition number* of matrix A as the ratio

$$\mathcal{K}(A) = \frac{\lambda_{\max}(A)}{\lambda_{\min}(A)}$$

where $\lambda_{\max}(A)$ and $\lambda_{\min}(A)$ are the largest and smallest eigenvalues of A, if A has real and positive eigenvalues.

In literature, we find estimates for the condition number of the matrix A^{exp} where the rows and columns of the boundary degrees of freedom are eliminated from the system. We highlight that the condition number of this matrix is the same as the iterative condition number of A, since their eigenvalues are the same (apart from the extra eigenvalues $\lambda = 1$ introduced by the boundary condition enforcement). This is why, in the analysis performed in this section, we compare $\mathcal{K}(A)$ with the results existing in literature.

In the following sections, we will also analyze the eigenvalues/condition number of the *mass matrix* associated with the basis of $\mathcal{F}_N(\mathcal{T}_h)$, that is, the matrix M with entries

$$M_{ij} = \int_{\Omega} \phi_i \phi_j \, dx.$$

Remark 2.4.2. *Whenever integrals should be calculated to assemble the matrices or to measure the error in a particular norm, e.g., $\|u - u_N\|_{H^1(\Omega)}$, we use a Gauss quadrature formula that integrates exactly all polynomials.*

2.4.1 1D Poisson problem

We considered in our numerical test two meshes. The first corresponds to take the whole domain as an element and the second corresponds to take 10 equally spaced subdomains. Regarding the bases, we compare the Lagrange bases with Equidistributed and Gauss-Lobatto points.

Error analysis. From Canuto, Hussaini, Quarteroni and Zang [11] we know that if $u \in H^s(\Omega)$ then

$$\|u - u_N\|_{H^1(\Omega)} \leqslant C_1(s) h^{\min(N+1,s)-1} N^{1-s} |u|_{H^s(\Omega)}$$

where C_1 is a constant that only depends on the regularity index s of u, u_N is the approximation obtained from the discrete problem (2.10) and $|u|_{H^s(\Omega)}$ is the standard semi-norm of $H^s(\Omega)$. From Figure 2.9 we can observe the spectral convergence of the method using

Figure 2.8: From left to right, meshes with one element ($h = 2.0$) and ten elements ($h = 0.1875$) for the 1D Poisson problem.

one or ten elements. We observe also, as to be expected, the faster convergence rate using more than one element due to the h^N factor in the error estimate. There is no substantial difference between the Lagrange bases associated with Gauss-Lobatto and Equidistributed points up to degree $N = 11$. For $N > 11$, better results are obtained using Gauss-Lobatto points as interpolation points and this is related with the conditioning of the stiffness matrix.

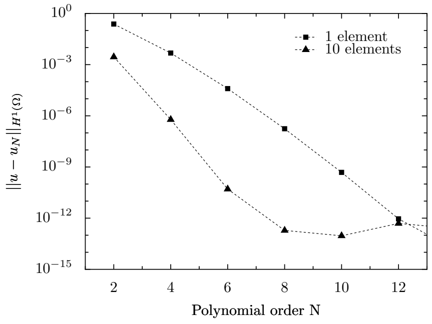
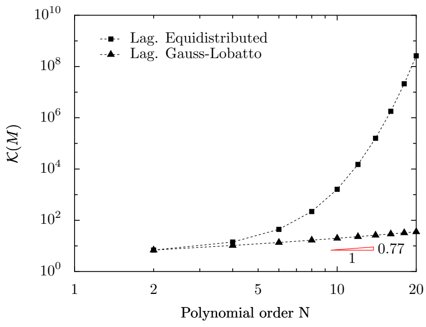
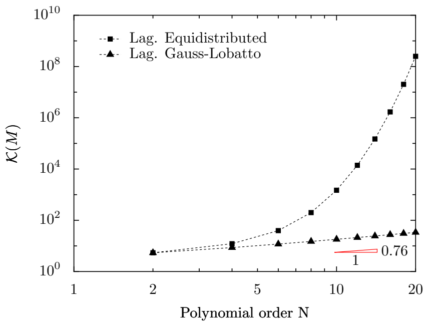


Figure 2.9: Error measured in the $H^1(\Omega)$-norm for the 1D Poisson problem.

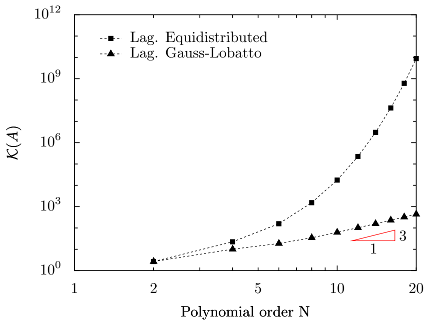
Iterative condition number of A. Regarding the matrix associated with the problem, we plot in Figures 2.10(c) and 2.10(d) the iterative condition number of matrix A for the previous choices of h. In this context, both bases show very different behaviors: the iterative condition number of the matrix obtained using the Lagrange basis associated with Gauss-Lobatto points grows algebraically as $\mathcal{O}(N^3)$, while for the other one, they grow exponentially. The behavior of the condition number for the Lagrange Gauss-Lobatto basis is comparable to the one associated with a Legendre approximation in one domain,



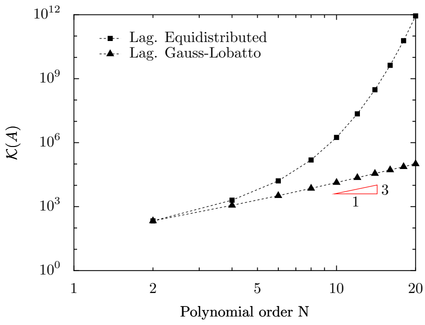
(a) $h = 2.0$.



(b) $h = 0.1875$



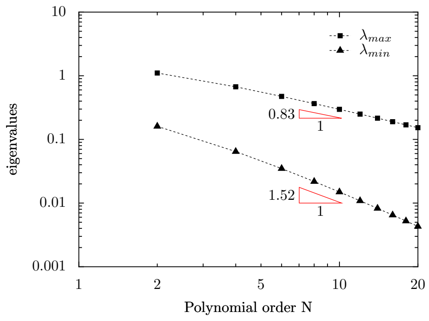
(c) $h = 2.0$.



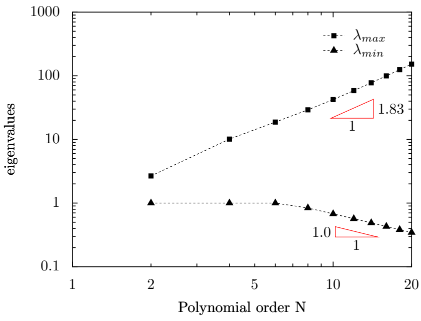
(d) $h = 0.1875$

Figure 2.10: Iterative condition number for the mass (top) and stiffness (bottom) matrices for the 1D Poisson problem.

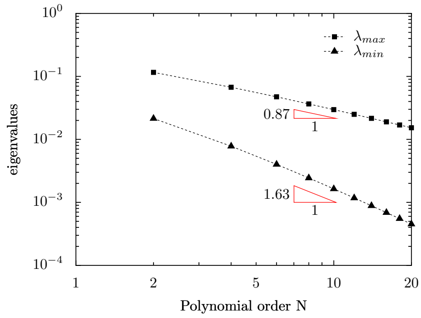
see Canuto, Hussaini, Quarteroni and Zang [12]), and is consistent with the results in Melenk [55]. The exponential growth of $\mathcal{K}(A)$ using a Lagrange basis with Equidistributed points had already been proved in Olsen and Douglas [58].



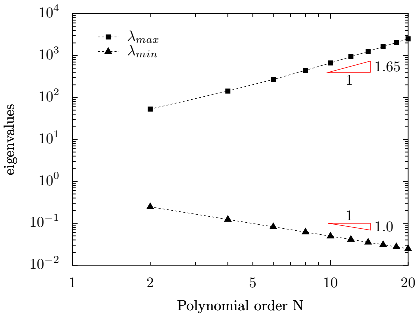
(a) Mass matrix.



(b) Stiffness matrix.



(c) Mass matrix.



(d) Stiffness matrix.

Figure 2.11: Maximum and minimum eigenvalues for the 1D Poisson problem ($h = 1$ on top and $h = 0.1875$ on bottom).

Condition number of M. In Figures 2.10(a) and 2.10(b) we notice that the behavior of $\mathcal{K}(M)$ is exponential for the Equidistributed point set basis and algebraic for the Gauss-Lobatto basis. In the latter basis, we estimate that this quantity grows as $\mathcal{O}(N^{0.7})$. As far as we know, these are new results concerning the iterative condition number of the mass matrix when using the Lagrange basis with Gauss-Lobatto points. Usually, the matrix M is *lumped*, that is, a Gauss-Lobatto quadrature formula is used to evaluate the entries $\int_\Omega \phi_i\phi_j \, dx$ and, in this case, the matrix turns out to be diagonal, and it is known that in this context, $\mathcal{K}(M) = \mathcal{O}(N)$. We highlight that in the one element case, M is full.

Eigenvalues of A and M. Regarding the largest and lowest eigenvalues of A and M, like for the condition number, they grow algebraically for the Gauss-Lobatto and exponentially for the Equidistributed bases. For the stiffness matrix, our results agree with the theoretical predictions for the Gauss-Lobatto Lagrange basis, see Canuto, Hussaini, Quarteroni and Zang [12]. Theory predicts that the smallest eigenvalue decreases with $\mathcal{O}(N^{-1})$ and this is confirmed from Figures 2.11(b) and 2.11(d). For the highest eigenvalue, we have smaller growth rates: the theoretical estimation for the growth is $\mathcal{O}(N^2)$ but we obtain $\mathcal{O}(N^{1.83})$ in the monodomain case and $\mathcal{O}(N^{1.65})$ in the multidomain example.

We display in Figures 2.11(a) and 2.11(c) the extreme eigenvalues for the mass matrix.

2.4.2 2D Poisson problem with triangular mesh

We now consider the domain $\Omega = \mathcal{T}^2$ and discretize it with two different meshes: one having only 1 element (this correspondes to take $h = 1.5$) and the other having 125 elements (which corresponds to take $h = 0.375$), see Figure 2.12. Regarding the bases

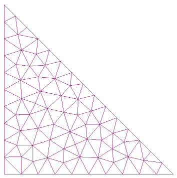


Figure 2.12: Mesh with 125 elements for the 2D Poisson problem defined in $\mathcal{T}^2$.

tested for this case, we only use Lagrange nodal bases associated with Fekete, Warpblend and Equidistributed point sets.

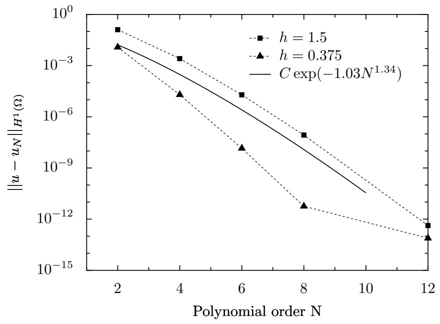
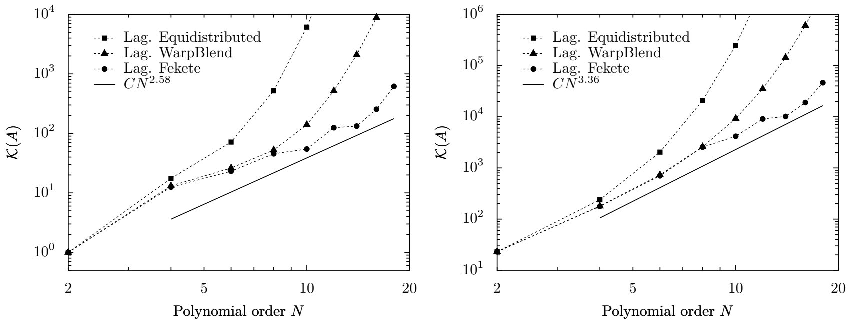


Figure 2.13: H^1 error plots for Poisson problem using a triangular mesh.

Error analysis. Again, spectral convergence is observed for both (single element) spectral method and the spectral element methods, see Figure 2.13. As in the 1D test problem, the error associated with each basis is the same, and a small difference is only noticed for large N, in the refined mesh.



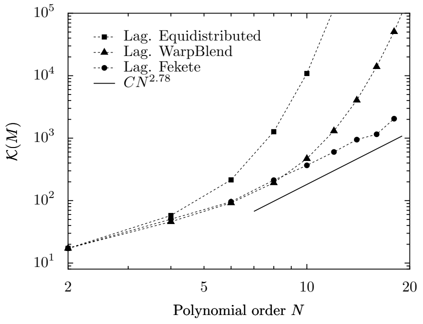
(a) $h = 1.5$.

(b) $h = 0.375$.

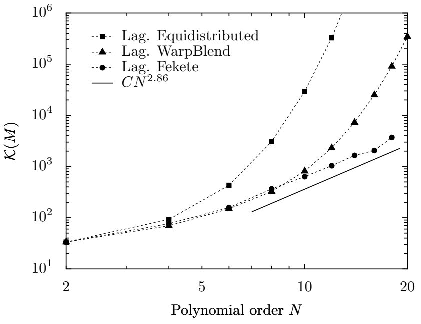
Figure 2.14: Iterative condition number $\mathcal{K}(A)$ for the Poisson problem discretized with a triangular mesh.

Iterative condition number of A and M. Unlike for the Gauss-Lobatto Lagrange basis, there are no known estimates regarding the conditioning number of the stiffness matrix, nor its maximum or minimum eigenvalues, in the case of simplicial domains. However, we provide some insight regarding the behavior of these quantities. In Figures 2.14 and 2.15 we can see that the condition number associated with the Lagrange Warpblend basis grows exponentially as $\mathcal{O}(2^N)$, both for the stiffness and mass matrices. Regarding the Lagrange basis associated with Fekete points, we estimated that $\mathcal{K}(A)$ grows like $\mathcal{O}(N^{3.36})$ and $\mathcal{K}(M)$ as $\mathcal{O}(N^{2.86})$ (in the spectral element case). The former result is consistent with the observations in Pasquetti and Rapetti [60]. We remark that up to degree 8, the iterative condition number associated with the Warpblend and Fekete points is almost the same. However, the Lagrange basis associated with Warpblend points produces linear systems that can be ill conditioned, for $N > 10$. We remark here that good preconditioning strategies should be developed to tackle the ill conditioned system that needs to be solved. A possibility, already found in Warburton, Pavarino and Hesthaven [91] for Fekete and Electrostatic based Lagrange bases, is to use the finite element preconditioner. Although there is no FEM-SEM equivalence (see Canuto, Hussaini, Quarteroni and Zang [12]) for these points, it can still ease the solving of such systems.

Eigenvalues of A and M. Finally, we show in Figure 2.17 the largest and smallest eigenvalue of the stiffness matrix. We denote by λ^F (resp. λ^W), the eigenvalue of the

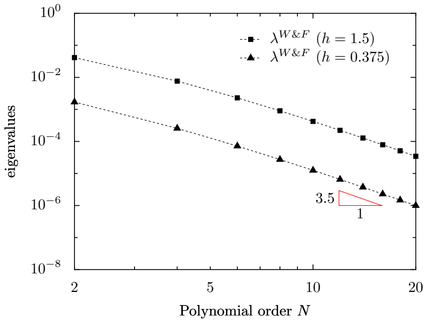


(a) $h = 1.5$.

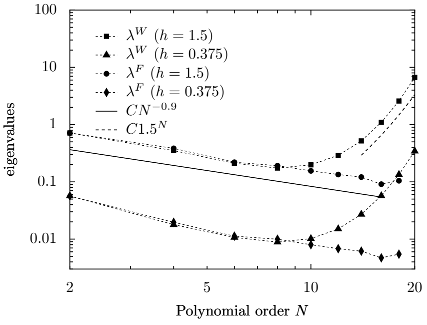


(b) $h = 0.375$.

Figure 2.15: Iterative condition number $\mathcal{K}(M)$ for Poisson problem discretized with a triangular mesh.



(a) Minimum eigenvalue.

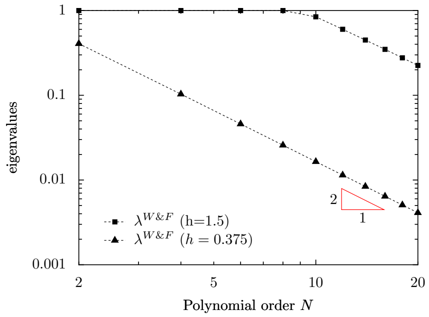


(b) Maximum eigenvalue.

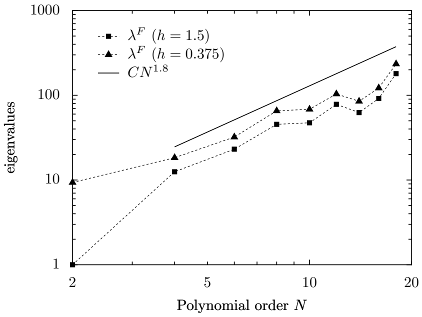
Figure 2.16: Eigenvalues for the mass matrix for the 2D Poisson problem using a triangular mesh.

underlying matrix using Fekete (resp. Warpblend) points.

We observe that the growth of the smallest eigenvalue of this matrix is the same as the theory predicts for a Gauss-Lobatto Lagrange basis. However, the same does not happen to the largest eigenvalue. We can see from Figures 2.14(a) and 2.14(b) an exponential growth for this quantity. Similar considerations are valid for the eigenvalues of the mass



(a) Minimum eigenvalue.



(b) Maximum eigenvalue.

Figure 2.17: Eigenvalues for the stiffness matrix for the 2D Poisson problem using a triangular mesh.

matrix in Figure 2.16.

2.4.3 2D Poisson problem with quadrangular mesh

Let us consider now problem (2.9) defined in $\Omega = (-1, 1)^2$. We define a single element mesh and also a uniform mesh with sixteen quadrangular elements.

In Figure 2.18 we show that in this case too, for monodomain or multidomain expansions, spectral convergence is achieved, for both Lagrange bases considered.

Error analysis. In this case, again we recover spectral convergence, see Figure 2.18.

Iterative condition number of A and M. It can be seen from Figures 2.19(a) and 2.19(b) that the iterative condition number of matrix A associated with Lagrange Gauss-Lobatto basis exhibits an algebraic growth rate. This result agrees with the estimation provided in Melenk [55] that states that the conditioning of this matrix scales as $h^{-2}N^3$.

The iterative condition number of M grows like $\mathcal{O}(N^{1.5})$ for the Lagrange basis associated with Gauss-Lobatto points. $\mathcal{K}(M)$ shows exponential growth when using the Lagrange basis with Equidistrubuted points.

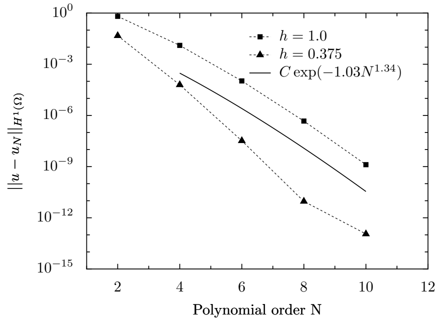
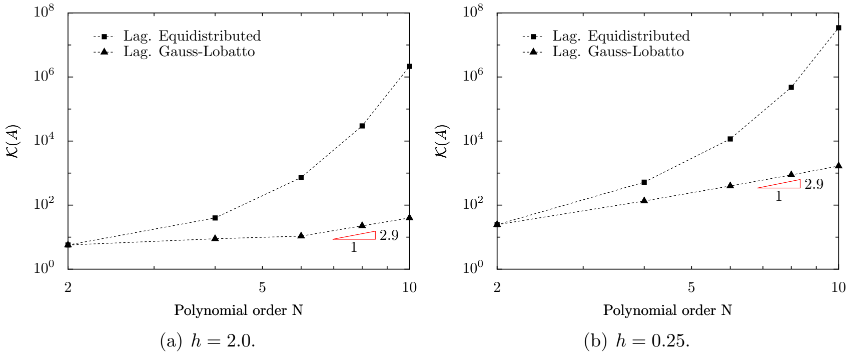


Figure 2.18: H^1 error for the Poisson problem with a quadrangular mesh.



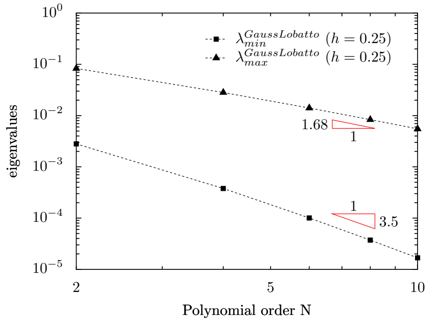
(a) $h = 2.0$.

(b) $h = 0.25$.

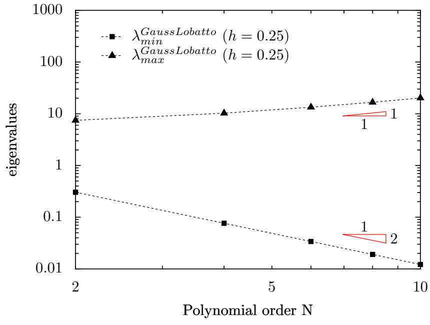
Figure 2.19: Condition number of the stiffness matrix for quadrangular meshes.

Eigenvalues of A and M. We start by introducing some notation. The symbol $\lambda_{min}^{\text{GaussLobatto}}$ (resp. $\lambda_{max}^{\text{GaussLobatto}}$) denotes the smallest (resp. largest) eigenvalue of the underlying matrix using Gauss-Lobatto points.

Regarding the eigenvalues of the stiffness matrix, it is interesting to notice that the smallest eigenvalue behaves exactly the same way as for the triangular case, that is, $\mathcal{O}(N^{-2})$. As for the largest eigenvalue, we estimate that is grows like $\mathcal{O}(N)$.



(a) Mass matrix.



(b) Stiffness matrix.

Figure 2.20: Minimum and maximum eigenvalues for A and M for the Poisson problem using quadrangular meshes.

Finally, we plot 2.20(b) the minimum and maximum eigenvalues of the mass matrix. We estimate that the minimum eigenvalue decreases as $\mathcal{O}(N^{-3.5})$ and the maximum as $\mathcal{O}(N^{-1.68})$. Again, the behavior of the minimum eigenvalue is the same as for the triangular case.

2.4.4 3D Poisson problem with tetrahedral mesh

We now consider the domain $\Omega = \mathcal{T}^3$ and discretize it with two different meshes: one having only 1 element and the other having 6 elements. Regarding the bases tested for this case, we only use Lagrange nodal bases associated with Warpblend and Equidistributed point sets.

Error analysis. Again, spectral convergence is observed for both monodomain and multidomain cases, see Figure 2.21. The error associated with each basis is the same, and a small difference is only noticed for large N, in the refined mesh.

Iterative condition number of A and M. The condition number of matrix A is plotted in Figures 2.23(a) and 2.23(b), for the cases when the case has one and six elements, respectively. We notice that using the Lagrange basis associated with Warpblend points

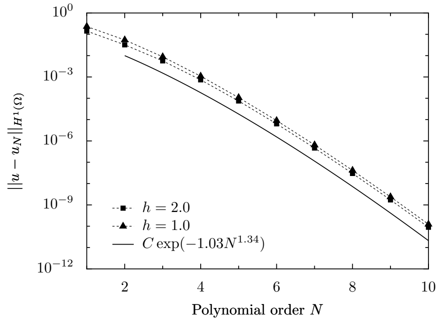
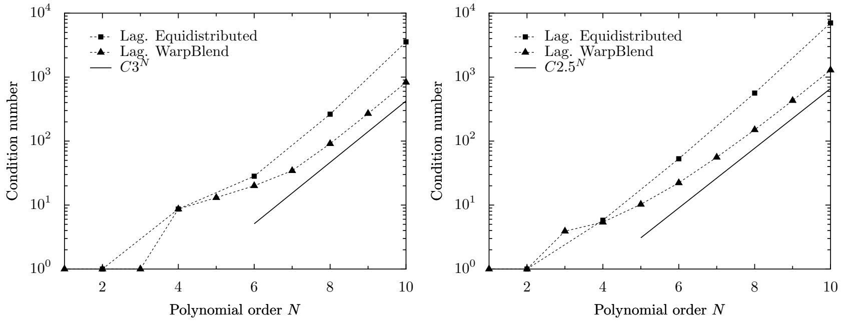


Figure 2.21: H^1 error plots for Poisson problem using a tetrahedral mesh.



(a) $h = 2.0$. (b) $h = 1.0$.

Figure 2.22: Iterative condition number $\mathcal{K}(A)$ for the Poisson problem discretized with a tetrahedral mesh.

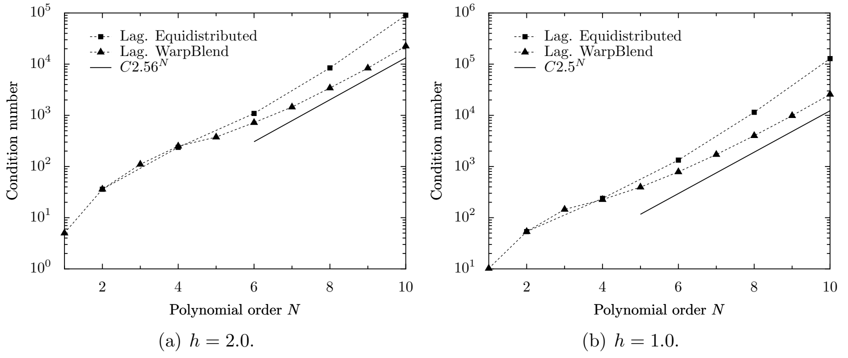


Figure 2.23: Iterative condition number $\mathcal{K}(M)$ for Poisson problem discretized with a tetrahedral mesh.

we get smaller condition numbers for A than using the Equispaced points. However, the growth in both cases is exponential ($\mathcal{O}(3^N)$ in the monodomain case and $\mathcal{O}(2.5^N)$ in the multidomain case). Again, good preconditioning strategies should be developed to tackle the ill conditioned system that needs to be solved. Similar considerations are valid for the mass matrix.

2.5 Conclusion

In this chapter, we extended the framework developed in chapter 1 to domains decomposed in several elements. The algorithms to construct a continuous expansion in a multidomain setting were addressed, as well as the discontinuous case. In the matter of matrix assembly, we provided an algorithm for assembly of linear and bilinear forms using the local to global table of degrees of freedom $n(\cdot,\cdot)$. In 2D, face integration was addressed and we provide the ideas and tools to calculate integrals in this context.

Regarding the Poisson test case presented in the last section, we obtained several results that were known in the literature: the condition number of the stiffness matrix in 1D and 2D, using Gauss-Lobatto points; spectral convergence using Fekete, Warpblend and Gauss-Lobatto based Lagrange bases. We also obtain some new numerical results regarding the point sets used, namely, the condition number and eigenvalues of the mass matrix associated with each discrete function space. We highlight that $\mathcal{K}(M)$ grows like $\mathcal{O}(N^{0.7})$ and $\mathcal{O}(N^{1.5})$ using Gauss-Lobatto points in 1D and 2D, respectively, and like $\mathcal{O}(N^{2.86})$ using Fekete points (in 2D). The condition number of A and M obtained using Warpblend points have always a growth that is exponential in N. All the other results regarding condition numbers are consistent with what can be found in the literature.

In the present work, we do not conduct a test for meshes composed of hexahedra.

This should be done to confirm the spectral convergence of the Gauss-Lobatto points and gather information on the eigenvalues and condition number of the stiffness and mass matrix associated with this problem.

Chapter 3

Algorithms for the incompressible Stokes and Navier-Stokes equations

In this chapter, we discuss algorithms to solve the incompressible Stokes and Navier--Stokes equations, written in the primitive variable formulation using the spectral element method. It is structured as follows. In section 3.1 we present the steady Stokes equations. After introducing its spectral element approximation, we provide some numerical tests to confirm the expected orders of convergence for several known inf-sup stable spaces for velocity and pressure, varying the mesh size and the polynomial degree. Spectral convergence is shown also for the Kovasznay test problem when using geometrical elements of degree one up to four. In the second section we introduce the unsteady Navier-Stokes equations and propose a spectral element discretization in space. Section 3.3 is dedicated to preconditioning strategies for the steady/unsteady Stokes/Navier-Stokes linear system arising from time/space discretization and linearization of the convective term. We present a block preconditioner and show the dependence of the number of iterations with respect to viscosity, mesh size, time step and polynomial degree. We also compare this preconditioner with two other strategies: a LU direct solver and a incomplete LU preconditioner. These strategies were compared in terms of time to calculate the preconditioners, time to solve the linear problem and number of iterations taken by an iterative solver. The last section presents a class of algebraic factorization methods to solve the unsteady Stokes/Navier--Stokes equations, called the *Yosida-q* schemes. We show that in our framework, these schemes maintain their expected order of convergence in time for the velocity and pressure fields. We address also the question of how to precondition efficiently the approximate Schur complement that appears in these methods.

3.1 The steady Stokes equations

Let Ω be a bounded open polygonal domain in $\mathbb{R}^d$ with Lipschitz continuous boundary $\partial\Omega$, $d = 2, 3$. The Stokes equations

$$\begin{aligned} -\nu\boldsymbol{\Delta}\mathbf{u} + \nabla p &= \mathbf{f}, \quad \text{in } \Omega \\ \operatorname{div}(\mathbf{u}) &= 0, \quad \text{in } \Omega \end{aligned} \tag{3.1}$$

are a fundamental model of viscous flow. Here ν represents the kinematic viscosity, the velocity vector field $\mathbf{u}$ represents the components of the velocity of the fluid and the scalar function p represents the pressure. The first equation arises from the *conservation of momentum* of the fluid and the second from the *conservation of mass*, see Streeter, Wylie and Bedford [82]. The fact that the second equation, called *incompressibility constraint*, does not involve the pressure makes the construction of finite/spectral element methods problematic.

To complete system (3.1), boundary conditions must be supplied to ensure its well-posedness. For simplicity of the presentation, we consider only homogeneous Dirichlet boundary conditions in the whole boundary $\partial\Omega$.

3.1.1 Variational formulation

The weak form of system (3.1) is obtained by multiplying both momentum and mass conservation equations by test functions and integrating by parts. We start by choosing suitable functional spaces for the trial and test functions, for velocity and pressure. Let us introduce the following spaces

$$\mathbf{H}_0^1(\Omega) = \left\{ \mathbf{v} \in H^1(\Omega)^d : \ \mathbf{v}_{|\partial\Omega} = \mathbf{0} \right\}$$

and

$$L_0^2(\Omega) = \left\{ q \in L^2(\Omega) : \ \int_\Omega q \, dx = 0 \right\}.$$

Remark 3.1.1. *The reason why the pressure is sought in $L_0^2(\Omega)$ in the case of Dirichlet boundary conditions is related to the fact that in $L^2(\Omega)$, it exists up to a constant. The extra zero average condition in $L_0^2(\Omega)$ helps to uniquely determine the pressure.*

The variational formulation reads as

Problem 3.1.1. *Find $\mathbf{u} \in \mathbf{H}_0^1(\Omega)$ and $p \in L_0^2(\Omega)$ such that*

$$\begin{aligned} \nu \int_\Omega \boldsymbol{\nabla}\mathbf{u} : \boldsymbol{\nabla}\mathbf{v} \, dx - \int_\Omega p \operatorname{div}(\mathbf{v}) \, dx &= \int_\Omega \mathbf{f} \cdot \mathbf{v} \, dx, \qquad \forall \mathbf{v} \in \mathbf{H}_0^1(\Omega) \\ \int_\Omega q \operatorname{div}(\mathbf{u}) \, dx &= 0, \qquad \forall q \in L_0^2(\Omega) \end{aligned} \tag{3.2}$$

where the notation : stands for the scalar product of two tensor fields of rank 2, $\mathbf{F}$ and $\mathbf{G}$, as

$$\int_\Omega \mathbf{F} : \mathbf{G}\ dx = \sum_{i=1}^{N}\sum_{j=1}^{N}\int_\Omega \mathbf{F}_{ij}\mathbf{G}_{ij}\ dx.$$

To simplify the notation, we define the bilinear forms, $a : \mathbf{H}_0^1(\Omega) \times \mathbf{H}_0^1(\Omega) \longrightarrow \mathbb{R}$ and $b : \mathbf{H}_0^1(\Omega) \times L_0^2(\Omega) \longrightarrow \mathbb{R}$ by

$$a(\mathbf{u},\mathbf{v}) = \nu \int_\Omega \boldsymbol{\nabla}\mathbf{u} : \boldsymbol{\nabla}\mathbf{v}\ dx, \qquad b(\mathbf{v},p) = \int_\Omega p\ \mathrm{div}(\mathbf{v})\ dx \tag{3.3}$$

and the linear bounded functional $F : \mathbf{H}_0^1(\Omega) \longrightarrow \mathbb{R}$ by

$$F(\mathbf{v}) = \int_\Omega \mathbf{f}\cdot\mathbf{v}\ dx.$$

where $\mathbf{f} \in L^2(\Omega)^d$. This functional will take a different form in the case of non homogeneous Dirichlet conditions or Neumann conditions, see Remark 3.1.3. Denoting $\mathbf{V} = \mathbf{H}_0^1(\Omega)$ and $Q = L_0^2(\Omega)$ we can rewrite problem (3.2) in the more compact form

Problem 3.1.2. *Find $\mathbf{u} \in \mathbf{V}$ and $p \in Q$ such that*

$$\begin{array}{rcl} a(\mathbf{u},\mathbf{v}) - b(\mathbf{v},p) & = & F(\mathbf{v}), \quad \forall \mathbf{v} \in \mathbf{V} \\ b(\mathbf{u},q) & = & 0, \quad \forall q \in Q. \end{array}$$

The uniqueness and existence of solution of Problem 3.1.2 depends on the regularity of the data and the functional spaces involved. In the case of pure Dirichlet boundary conditions, a solution exists (see Brezzi [5]) if

$$\exists \beta > 0 : \ \forall p \in Q : \sup_{0\neq\mathbf{v}\in\mathbf{V}} \frac{b(\mathbf{v},p)}{\|\mathbf{v}\|_{\mathbf{V}}} \geqslant \beta\,\|p\|_Q \tag{3.4}$$

$$\exists C > 0 : \ a(\mathbf{v},\mathbf{v}) \geqslant C\,\|\mathbf{v}\|_{\mathbf{V}}^2, \quad \forall \mathbf{v} \in \mathbf{V} \tag{3.5}$$

Condition (3.4) is called the *continuous inf-sup condition* and (3.5) refers to the *coercivity* of the bilinear form $a(\cdot,\cdot)$.

Remark 3.1.2. *In fact, it is sufficient for a solution to exist that $a(\cdot,\cdot)$ be coercive on the subspace*

$$\tilde{\mathbf{V}} = \{\mathbf{v} \in \mathbf{V} : \ b(\mathbf{v},q) = 0, \ \forall q \in Q\}$$

It is easily seen that $a(\cdot,\cdot)$ is continuous and coercive and therefore, the existence and uniqueness of solution depends only on condition (3.4) as well as the regularity of the given data.

Remark 3.1.3. *If there is a part of $\partial\Omega$ where Neumann boundary conditions are imposed, say $\partial\Omega_N$, then the natural space to look for the velocity $\mathbf{u}$ is*

$$\mathbf{H}^1_{\Gamma^D}(\Omega) = \left\{\mathbf{v} \in \mathbf{H}^1(\Omega) : \ \mathbf{v}_{|_{\Gamma^D}} = \mathbf{0}\right\}. \tag{3.6}$$

being Γ^D that part of the boundary on which homogeneous Dirichlet conditions are imposed on the velocity field. The Neumann condition has also an impact on choice of the functional space for the pressure field. Actually, the fact that Neumann boundary conditions are imposed in a nonzero measure set implies that the average of the pressure field is fixed. Therefore the pressure should be sought in $L^2(\Omega)$. In this case, conditions (3.4) *and* (3.5) *still hold.*

3.1.2 The discrete problem

Let $\mathcal{T}_h$ be a triangulation of the domain Ω, see section 2.1. For simplicity we consider a partition with straight edge geometrical elements only, i.e., $N_{\text{geo}} = 1$.

The spectral element spaces used to approximate the velocity and pressure fields are $\mathbf{V}_N = (\mathcal{F}_N(\mathcal{T}_h))^2$ and $Q_M = \mathcal{F}_M(\mathcal{T}_h)$, where $M = N-1$ or $M = N-2$ and $\mathcal{F}_N(\mathcal{T}_h)$ is the space introduced in section 2.1. We remark that the definition in the case of a quadrangular partition is straightforward. When we use these discretization spaces, we will refer to the $\mathbb{P}_N$ - $\mathbb{P}_M$ method ($\mathbb{Q}_N$ - $\mathbb{Q}_M$ if the partition is made out of quadrangles) to be consistent with the nomenclature in the literature. We notice that in this case, we define the pressures continuous across the elements of the mesh. The discontinuous pressure version of this method is also considered and denoted by $\mathbb{P}_N$ - $\mathbb{P}_M^{disc}$.

The discrete problem reads as follows

Problem 3.1.3. *Find $(\mathbf{u}_N, p_N) \in \mathbf{V}_N \times Q_M$ such that*

$$\begin{array}{rcl} a(\mathbf{u}_N, \mathbf{v}) - b(\mathbf{v}, p_N) & = & F(\mathbf{v}) \ \ \forall \mathbf{v} \in \mathbf{V}_N \\ b(\mathbf{u}_N, q) & = & 0, \ \ \forall q \in Q_M. \end{array} \tag{3.7}$$

In order to reduce (3.7) to an algebraic problem, we consider basis functions for the spaces associated with the Galerkin method, say $\mathbf{V}_N = \text{span}\{\boldsymbol{\phi}_i\}$ and $Q_M = \text{span}\{\psi_i\}$. Denoting $\mathbf{U}_N$ and $\mathbf{P}_N$ as the representation of $\mathbf{u}_N$ and p_N w.r.t. these basis, Problem 3.1.3 is equivalent to find the solution of a linear system of the form

$$\begin{bmatrix} F_N & G_N \\ D_N & 0 \end{bmatrix} \begin{bmatrix} \mathbf{U}_N \\ \mathbf{P}_N \end{bmatrix} = \begin{bmatrix} \mathbf{F} \\ 0 \end{bmatrix}. \tag{3.8}$$

where $F_N = [a(\boldsymbol{\phi}_i, \boldsymbol{\phi}_j)]_{i,j}$, $G_N = [-b(\boldsymbol{\phi}_i, \psi_j)]_{i,j}$ and $D_N = -G_N^T$.

From the theoretical point of view, the pressure is sought in $L^2_0(\Omega)$ due to the fact that it exists up to a constant in $L^2(\Omega)$. In practice, to construct a finite dimensional space whose basis functions have zero mean is undesirable. This issue can be circumvented in three ways:

- using the space Q_M that approximates $L^2(\Omega)$ and determine the pressure field up to a constant. The linear system (3.8) will be singular and the linear solver should take that into account. A possibility is to use for instance a Krylov subspace solver with an incomplete factorization.
- fix the pressure field algebraically in one point. This fixes the pressure to have a certain average and makes system (3.8) non singular. The solution $\mathbf{P}_N$ of (3.8) must then be shifted in order for the pressure average to be zero.
- using a Lagrange multiplier approach to fix the average of the pressure.

The latter strategy is actually the one used whenever we solve Stokes problems with only Dirichlet boundary conditions. The idea consists in adding a new equation to (3.1)

$$\int_\Omega p\,dx = 0$$

and use a Lagrange multiplier to enforce it. The weak formulation for the new problem reads as

Problem 3.1.4. *Find $(\mathbf{u}_N, p_N, \lambda_N) \in \mathbf{V}_N \times Q_M \times \mathbb{R}$ such that*

$$\begin{aligned} a(\mathbf{u}_N,\mathbf{v}) - b(\mathbf{v},p_N) &= F(\mathbf{v}) \quad \forall \mathbf{v} \in \mathbf{V}_N && (3.9)\\ b(\mathbf{u}_N,q) + \int_\Omega \lambda_N q\,dx &= 0, \quad \forall q \in Q_M && (3.10)\\ \int_\Omega \mu p_N\,dx &= 0, \quad \forall \mu \in \mathbb{R}. && (3.11) \end{aligned}$$

This problem is now reduced to solving a linear system with one more row and column that is non singular.

Remark 3.1.4. *When dealing with non homogeneous Dirichlet boundary conditions, say*

$$\mathbf{u}_{|\partial\Omega} = \mathbf{g},$$

we write the full variational formulation for the Stokes equations, meaning that we consider any $\mathbf{H}^1(\Omega)$ function as test function for the velocity. At the discrete level, we consider an approximation space whose functions are nonzero at the boundary and impose the Dirichlet boundary conditions algebraically, as mentioned in Remark 2.4.1. The solution obtained with this procedure is the same as the one obtained by deriving the variational formulation with $\mathbf{v} \in \mathbf{H}^1_0(\Omega)$ and lifting the Dirichlet boundary condition with an appropriate interpolant that belongs to $\mathbf{V}_N$. This interpolant is defined as the nodal projection of $\mathbf{g}$ in $\partial\Omega$ and extended to the interior of Ω by setting the respective degrees of freedom to zero. For more details, consult Sherwin and Karniadakis [48].

Stability and convergence of the discrete problem

We first observe that the bilinear and linear forms $a(\cdot,\cdot)$ and $b(\cdot,\cdot)$ are continuous, that is, there exist constants γ, δ such that

$$\begin{aligned} |a(\mathbf{u},\mathbf{v})| &\leqslant \gamma \|\mathbf{u}\|_{\mathbf{V}} \|\mathbf{v}\|_{\mathbf{V}}, \quad \forall \mathbf{u},\mathbf{v} \in \mathbf{V} \\ |b(\mathbf{v},q)| &\leqslant \delta \|\mathbf{v}\|_{\mathbf{V}} \|q\|_{Q}, \quad \forall \mathbf{v} \in \mathbf{V},\ \forall q \in Q. \end{aligned}$$

On the other hand, concerning the right hand side of the momentum equation, due to the Cauchy-Schwarz inequality, there exists $C(f)$ such that

$$|(f,\mathbf{v})| \leqslant C(f) \|\mathbf{v}\|_{\mathbf{V}}, \quad \forall \mathbf{v} \in \mathbf{V} \tag{3.12}$$

The existence and uniqueness of solution of this problem is then connected with the following result, due to Brezzi [5],

(i) Setting

$$\mathbf{Z}_N = \{\mathbf{v} \in \mathbf{V}_N :\ b(\mathbf{v},q) = 0,\ \forall q \in Q_M\} \tag{3.13}$$

there exists a constant $\alpha_N > 0$ such that

$$a(\mathbf{v},\mathbf{v}) \geqslant \alpha_N \|\mathbf{v}\|_{\mathbf{V}}, \quad \forall \mathbf{v} \in \mathbf{Z}_N \tag{3.14}$$

(ii) there exists a constant $\beta_N > 0$ such that

$$\sup_{\mathbf{v}\in\mathbf{V}_N} \frac{b(\mathbf{v},q)}{\|\mathbf{v}\|_{\mathbf{V}}} \geqslant \beta_N \|q\|_Q,\ \forall q \in Q_M. \tag{3.15}$$

The constant β_N is known as *inf-sup constant.*

If these conditions are verified then the following error estimates are valid:

$$\|\mathbf{u}-\mathbf{u}_N\|_{\mathbf{V}} \leqslant \left(1+\frac{\gamma}{\alpha_N}\right) \inf_{\mathbf{w}\in Z_N} \|\mathbf{u}-\mathbf{w}\|_{\mathbf{V}} + \frac{\delta}{\alpha_N} \inf_{q\in Q_M} \|p-q\|_Q \tag{3.16}$$

and

$$\|p-p_N\|_Q \leqslant \frac{\gamma}{\beta_N}\left(1+\frac{\gamma}{\alpha_N}\right) \inf_{\mathbf{w}\in\mathbf{Z}_N} \|\mathbf{u}-\mathbf{w}\|_{\mathbf{V}} + \left(1+\frac{\delta}{\beta_N}\left(1+\frac{\gamma}{\alpha_N}\right)\right) \inf_{q\in Q_M} \|p-q\|_Q \tag{3.17}$$

The condition (3.15) is called *discrete inf-sup condition.*

In our context, we suppose that there exists $\overline{\alpha} > 0$ not depending in N such that

$$0 < \overline{\alpha} \leqslant \alpha_N, \quad \forall N.$$

It can be shown (see Canuto, Hussaini, Quarteroni and Zang [11]) that

$$\begin{aligned} \|\mathbf{u}-\mathbf{u}_N\|_{\mathbf{V}} + \beta_N \|p-p_N\|_Q \quad \leqslant \quad & C_1 \beta_N^{-1} h^{\min\{N,s-1\}} N^{1-s} \|\mathbf{u}\|_{\mathbf{H}^s(\Omega)} \\ & + C_2 h^{\min\{M+1,s-1\}} M^{1-s} \|p\|_{\mathbf{H}^{s-1}(\Omega)} \end{aligned} \tag{3.18}$$

for $\mathbf{u} \in \mathbf{H}^s(\Omega)$ and $p \in H^{s-1}(\Omega)$, $s \geqslant 1$.

For more details concerning these error estimates or estimates that account for numerical integration, we refer again the reader to Canuto, Hussaini, Quarteroni and Zang [11].

Compatibility condition between velocity and pressure: the discrete inf-sup condition

The inequality (3.15) requires a compatibility between the velocity and pressure spaces involved in the discretization of the Stokes problem. The violation of (3.15) produces a set of *spurious pressure modes*, that is, discrete pressures q, non constant in Ω, that satisfy

$$b(\mathbf{v}, q) = 0, \quad \forall \mathbf{v} \in \mathbf{V}_N. \tag{3.19}$$

Since the pressure only enters the Stokes equations through the gradient and no boundary conditions are associated with it, any pair $(\mathbf{u}, p + q)$ is a solution as long as $(\mathbf{u}, p)$ solves the Stokes problem.

The spurious modes can also be characterized at an algebraic level. Using the notation as in section 3.1.2, condition (3.19) is written as

$$G_N \overline{q} = 0 \tag{3.20}$$

for a discrete pressure $\overline{q}$ corresponding to the chosen basis. We remark that this condition is equivalent to the matrix $G_N^T G_N$ having a zero eigenvalue with multiplicity greater than 1.

Another important concept that is related with the discrete inf-sup condition is the notion of *pseudo-spurious pressure modes*. To introduce this notion, let us consider the following generalized eigenvalue problem

$$G_N^T F_N^{-1} G_N \overline{q} = \mu M_p \overline{q} \tag{3.21}$$

where M_p denotes the pressure mass matrix. Denoting β_N^* as the square root of the smallest eigenvalue of (3.21), we say that the numerical method has pseudo-spurious modes if

$$\lim_{N \longrightarrow \infty} \beta_N^* = 0.$$

It can be shown (see the details in Canuto, Hussaini, Quarteroni and Zang [11]) that the quantity β_N^* behaves asymptotically as β_N, the inf-sup constant.

3.1.3 Choices of the pressure space

By fixing the space that approximates the velocities as $\mathbb{P}_N$ or $\mathbb{Q}_N$, we consider different possibilities to approximate the pressure field. As already mentioned, there are several possible choices, but we will restrict our analysis to $\mathbb{P}_M$ (or $\mathbb{Q}_M$), for $M \in \{N-1, N-2\}$, continuous or discontinuous across the elements of the underlying mesh.

We present now a numerical study of the convergence properties of such spaces. As a model problem, we consider the Kovasznay solution of the steady Stokes equations, see

Kovasznay [52]. The exact solution is

$$\begin{aligned} \mathbf{u}(x,y) &= \left[1 - e^{\lambda x}\cos(2\pi y), \frac{\lambda}{2\pi}e^{\lambda x}\sin(2\pi y)\right]^T \\ p(x,y) &= -\frac{e^{2\lambda x}}{2} \\ \lambda &= \frac{1}{2\nu} - \sqrt{\frac{1}{4\nu^2} + 4\pi^2}. \end{aligned} \tag{3.22}$$

The domain is defined as $\Omega = (-0.5, 1) \times (-0.5, 1.5)$ and $\nu = 0.035$. The forcing term for the momentum equation is obtained from the solution and is

$$\mathbf{f} = \left(e^{\lambda x}\left(\left(\lambda^2 - 4\pi^2\right)\nu\cos(2\pi y) - \lambda e^{\lambda x}\right), e^{\lambda x}\nu\sin(2\pi y)(-\lambda^2 + 4\pi^2)\right)^T \tag{3.23}$$

Dirichlet boundary conditions are derived from the exact solution.

The norms in which the error for velocity and pressure are computed are the $\mathbf{H}^1(\Omega)$-norm and $L^2(\Omega)$-norm, for velocity and pressure, respectively.

Space discretization, assembly and linear solver

We have to choose bases for the spaces $\mathbf{V}_N$ and Q_M. In both cases, we use a Lagrange basis, associated with Fekete points in the case the mesh is made up of triangles, and associated with Gauss-Lobatto points in the quadrangular mesh case. In Figure 3.1 we see some of the meshes used in the simulation. The choice of the basis for the pressure in the discontinuous setting is nonstandard in the case of quadrangular meshes. Typically, the bases chosen for this kind of approach are built upon Gauss points or interior Gauss-Lobatto points. We consider the full set of Gauss-Lobatto points to create the Lagrange basis, including the ones in the boundary of the elements. The reason of this choice is to retain the similarity with the equivalent method defined in a triangular mesh and using the Fekete points.

Whenever integrals over the domain have to be calculated, we use a quadrature rule with a sufficiently high number of points that integrates all bilinear/linear forms exactly. In the triangular and quadrangular mesh cases, a quadrature rule is used.

At the linear algebra level, the system is solved with a LU factorization.

The $\mathbb{Q}_N$ - $\mathbb{Q}_{N-1}$ and $\mathbb{P}_N$ - $\mathbb{P}_{N-1}$ methods

One of the most known elements in the finite element community is $\mathbb{Q}_2$ - $\mathbb{Q}_1$ or $\mathbb{P}_2$ - $\mathbb{P}_1$. These are called Taylor-Hood elements. The generalization of these spaces to high degree was studied by Brezzi, Falk [6] in the case of meshes composed by squares and triangles. It was concluded that $\mathbb{Q}_N$ - $\mathbb{Q}_{N-1}$ is inf-sup stable and for meshes satisfying certain properties, so is $\mathbb{P}_N$ - $\mathbb{P}_{N-1}$. However, in the work of Ainsworth and Coggins [1], numerical evidence is shown that the inf-sup constant decays to zero with the increasing polynomial degree.

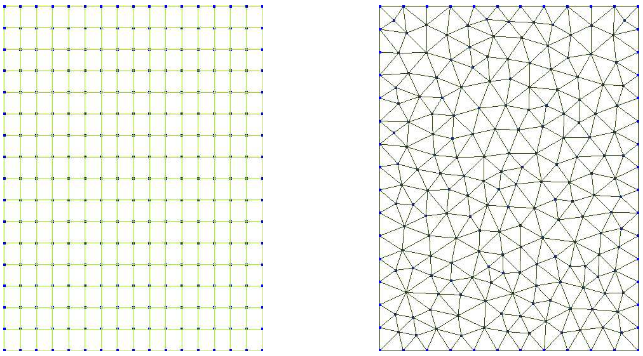


Figure 3.1: Meshes used in the simulations ($h = 0.125$). On the left an uniform quadrangular mesh, on the right a nonuniform triangular mesh.

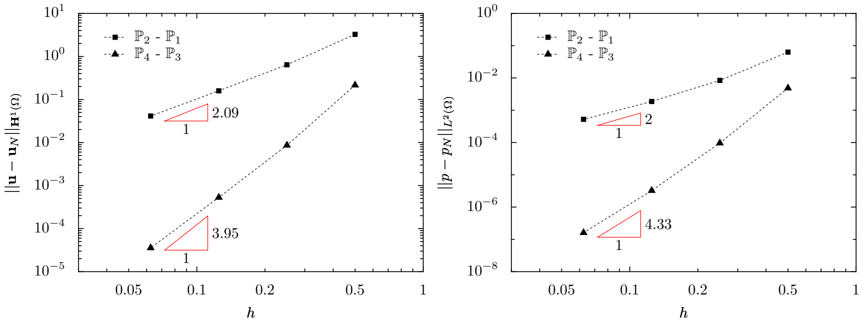


Figure 3.2: Error plot for the velocity and pressure for the generalized Taylor-Hood element $\mathbb{P}_N$ - $\mathbb{P}_{N-1}$.

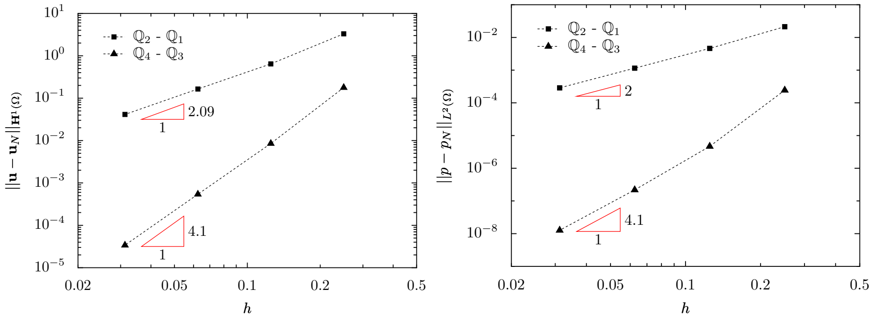


Figure 3.3: Error plot for the velocity and pressure for the generalized Taylor-Hood element $\mathbb{Q}_N$ - $\mathbb{Q}_{N-1}$.

In Figures 3.2 and 3.3 we plot the errors for the velocity and pressure fields. From Figure 3.2, we remark that the $\mathbf{H}^1(\Omega)$ error for the velocity is of order two for $\mathbb{P}_2$ - $\mathbb{P}_1$ and approximately order four for $\mathbb{P}_4$ - $\mathbb{P}_3$. On the other hand, the pressure error is of order two for $\mathbb{P}_2$ - $\mathbb{P}_1$ and of order approximately 4.3 for $\mathbb{P}_4$ - $\mathbb{P}_3$. The convergence order for the velocity and pressure fields appears to be optimal regarding the approximation space. Similar considerations are valid for the $\mathbb{Q}_N$ - $\mathbb{Q}_{N-1}$ method and Figure 3.3.

The $\mathbb{Q}_N$ - $\mathbb{Q}^{disc}_{N-2}$ and $\mathbb{P}_N$ - $\mathbb{P}^{disc}_{N-2}$ methods

The $\mathbb{Q}_N$ - $\mathbb{Q}^{disc}_{N-2}$ method was proposed by Bernardi and Maday in [2] and is one of the most widely used discretization spaces. It is known that the inf-sup constant is proportional to $N^{\frac{1-d}{2}}$. A quasi-optimal error estimate is obtained in this case from the estimate (3.18)

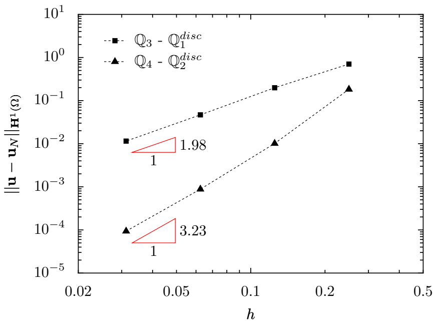
$$\begin{aligned}\|\mathbf{u}-\mathbf{u}_N\|_V + N^{\frac{1-d}{2}}\|p-p_N\|_Q \quad &\leqslant \quad C_1 h^{\min(N,s-1)} N^{1-s}\|\mathbf{u}\|_{\mathbf{H}^s(\Omega)} \\ &\quad\quad +C_2 h^{\min(N-1,s-1)} N^{1-s}\|p\|_{\mathbf{H}^{s-1}(\Omega)}\end{aligned}$$

In Figure 3.4 we can observe the quasi-optimality of the previous estimate regarding the velocity. Though we approximate the velocity with N degree velocities, the error decays as h^{N-1}, for $N=3,4$. As for the pressure, the error behaves as h^{N-1}, thus showing optimality regarding the approximation space.

The same kind of quasi-optimality is present in the $\mathbb{P}_N$ - $\mathbb{P}^{disc}_{N-2}$ method. This is documented in Figure 3.5.

The $\mathbb{Q}_N$ - $\mathbb{Q}_{N-2}$ and $\mathbb{P}_N$ - $\mathbb{P}_{N-2}$ methods

Regarding this choice of spaces, since their counterparts with discontinuous pressures are inf-sup stable, so will these be. With respect to error estimates, we can refer to the work



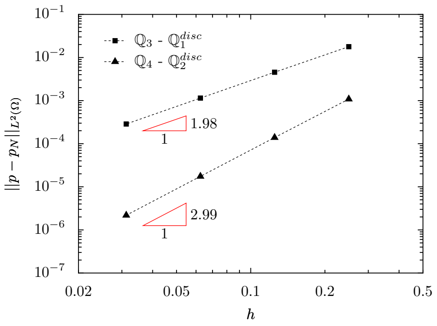
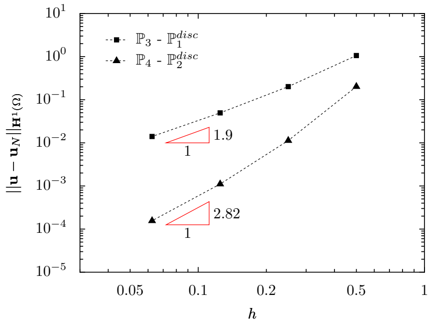


Figure 3.4: Error plot for the velocity and pressure for the element $\mathbb{Q}_N$ - $\mathbb{Q}_{N-2}^{disc}$.



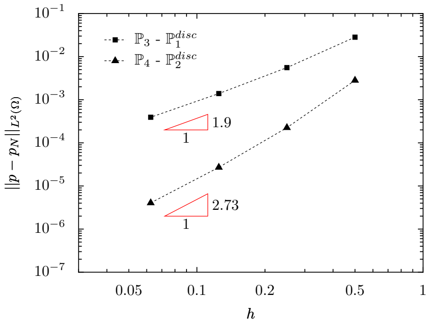


Figure 3.5: Error plot for the velocity and pressure for the element $\mathbb{P}_N$ - $\mathbb{P}_{N-2}^{disc}$.

of Schwab and Suri [76] for the $\mathbb{P}_N$ - $\mathbb{P}_{N-2}$ method. In this paper, exponential convergence is proven for the Stokes problem. The numerical results in terms of convergence order are

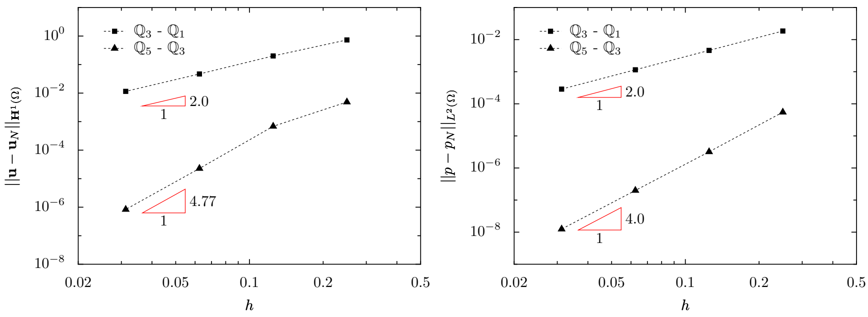


Figure 3.6: Error plot for the velocity and pressure for the element $\mathbb{Q}_N$ - $\mathbb{Q}_{N-2}$.

very similar to the ones obtained in the previous section for the $\mathbb{P}_N$ - $\mathbb{P}^{disc}_{N-2}$ and $\mathbb{Q}_N$ - $\mathbb{Q}^{disc}_{N-2}$ methods, and here at the cost of a reduced pressure space.

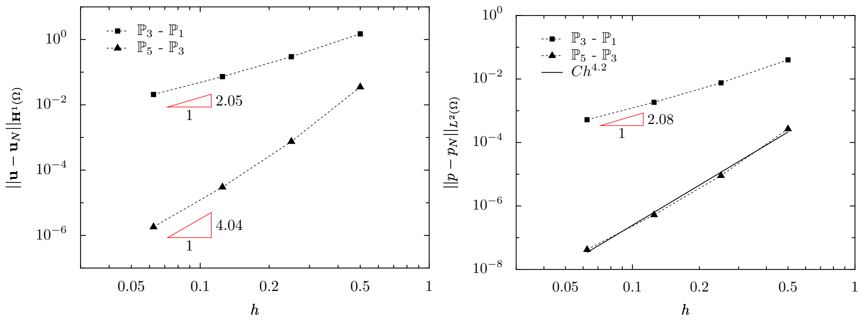


Figure 3.7: Error plot for the velocity and pressure for the element $\mathbb{P}_N$ - $\mathbb{P}_{N-2}$.

Comparison between the different methods and conclusion

If a h refinement is seeked, which of the previous methods should we choose? By analyzing the results presented before, the answer is the generalized Taylor-Hood method. This method, for meshes with triangles or quadrangles, is optimal regarding the approximation spaces used, for velocity and pressure, whereas the other is one order less from optimality.

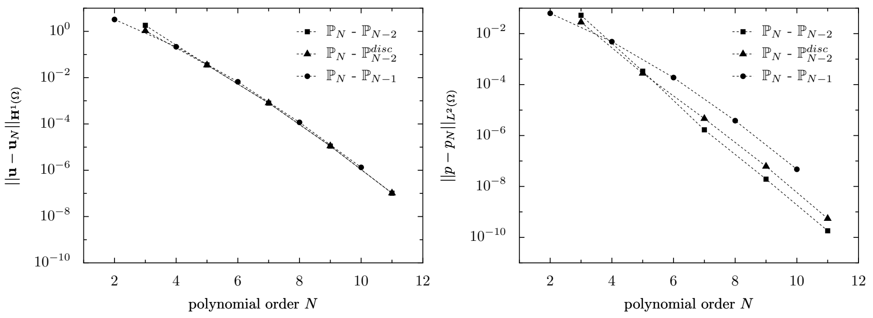


Figure 3.8: Error plot for the several methods associated with triangular meshes.

In the case of a N refinement, then the situation is different. First, we remark that the inf-sup constant associated with the Taylor-Hood methods decays to zero probably at a higher rate than $N^{\frac{1-d}{2}}$. Numerical evidence by Ainsworth and Coggins [1] showed that this should behave as N^{-1}. We did not experience any degradation in the pressure field as result of the increased polynomial degree, and for the range of N that is used in practice, this does not seem to be a problem. However, the generalized Taylor-Hood methods perform worse in terms of polynomial degree refinement. In Figures 3.8 and 3.9 we plot for a fixed mesh size h, the errors for the velocity and pressure fields depending on the polynomial degree. From these figures, we conclude that if we refine in N then one should use the $\mathbb{P}_N$ - $\mathbb{P}_{N-2}$

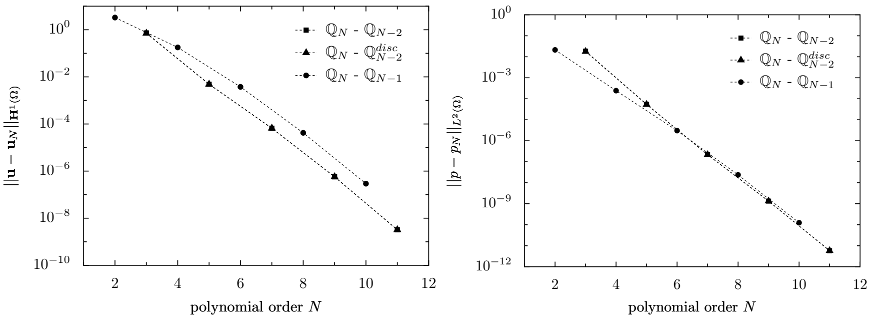


Figure 3.9: Error plot for the several methods associated with quadrangular meshes.

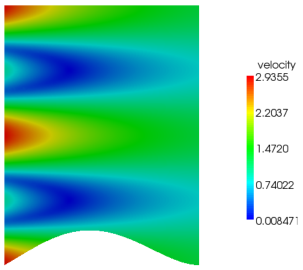
or the $\mathbb{Q}_N$ - $\mathbb{Q}_{N-2}$ methods. Comparing with the generalized Taylor-Hood method, the latter show better accuracy properties and if we compare it with it's discontinuous version, though there is no substantial difference in the quadrangular case, the pressure error is slightly smaller in the triangular case.

3.1.4 Using high order geometrical elements

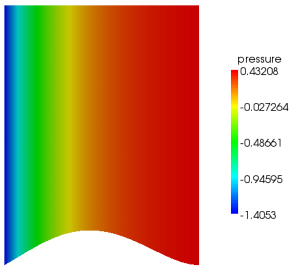
We also checked for spectral convergence of our method w.r.t. using high order triangular geometrical elements. For this benchmark, we consider again the Kovasznay solution (3.22) defined in the new domain Ω obtained from the rectangle $(-0.5, 1) \times (-0.5, 1.5)$ by replacing the lower edge with the image of

$$\boldsymbol{\varphi}(x) = \left[x, 0.08(x+0.5)(x-1)(x^2-1)\right]^T, \quad x \in (-0.5, 1).$$

Again, we impose the Kovasznay solution as Dirichlet boundary condition. A plot of the domain's shape and pressure profile is depicted in Figure 3.10.



(a) Magnitude of velocity



(b) Pressure

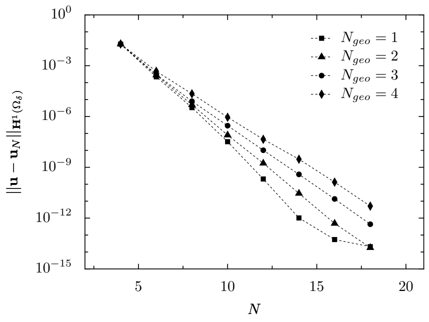
Figure 3.10: Plot of the solution of the Kovasznay problem.

Let us fix $h > 0$ and $N_{\text{geo}} = 1, 2, 3, 4$. We consider a triangulation $\mathcal{T}_\delta$ of Ω, where $\delta = (h, N_{\text{geo}})$ and the respective domain Ω_δ it induces, that is,

$$\overline{\Omega_\delta} = \bigcup_{T \in \mathcal{T}_\delta} T.$$

Remark 3.1.5. *Notice that if geometrical elements of order four are used, the boundary of Ω can be described exactly for all h and no error is produced from approximating the geometry of the domain.*

In Figure 3.11 we plot the error in $\mathbf{H}^1(\Omega_\delta)$-norm for the velocity and $L^2(\Omega_\delta)$-norm for the pressure. We observe that spectral convergence is achieved using all four types of



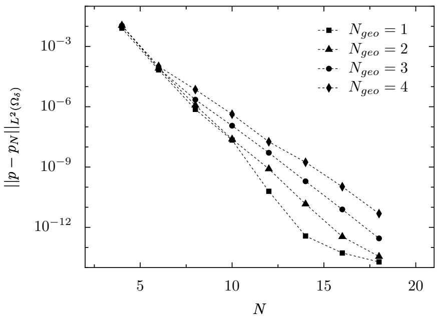


Figure 3.11: Convergence plots for the modified Kovasznay example using several high order geometrical elements and the $\mathbb{P}_N - \mathbb{P}_{N-2}$ method.

geometrical elements. We stress that the error coming from approximation of the domain is not taken into account in the error quantities we calculated. With this test case, we merely want to show that spectral convergence is still achieved, even using high order geometrical elements. We remark from Figure 3.11 the appearance of a shift in the errors, both for velocity and pressure, when we vary the polynomial degree of the geometrical transformation N_{geo}. This is due to what we already mentioned in Remark 1.4.1.

If we calculate the error of the approximation in the $\mathbf{H}^1(\Omega)$-norm then we obtain the plots shown in Figure 3.12. The solution $\mathbf{u}_N$ of the problem is nodally projected onto the

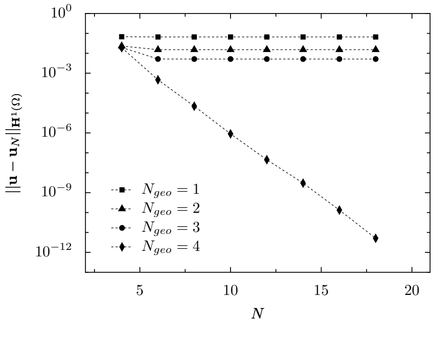


Figure 3.12: Convergence plot in the $\mathbf{H}^1(\Omega)$-norm for the modified Kovasznay example using high order geometrical elements and the $\mathbb{P}_N - \mathbb{P}_{N-2}$ method.

velocity space of the same polynomial degree associated with the fourth order mesh (that describes the domain exactly). By taking into account the approximation of the domain in the error, we observe that for $N_{\text{geo}} = 1, 2, 3$, the error stagnates. There is only convergence when the geometry is exactly described.

Another way to assess the accuracy of a method using high order geometrical elements, is to calculate the *drag* and *lift* forces on the curved boundary. Let us define $\partial\Omega_{bottom}$ as the part of the boundary of Ω that is curved. Then, these coefficients are respectively the first and second component of the vector force

$$\int_{\partial\Omega_{bottom}} (-p\mathbf{I} + \nu\boldsymbol{\nabla}\mathbf{u})\mathbf{n}\; ds.$$

We can use as reference drag and lift the values obtained in the fourth order mesh (that describes the domain exactly) and with an approximation space of high degree for velocity and pressure. In Figure 3.13 we plot the error between several approximations for the domain/fluid variables and exact drag/lift values (obtained using the fourth order mesh and the $\mathbb{P}_8 - \mathbb{P}_6$ method.

In respect of the geometrical elements, we get similar results as in Figure 3.12: the error stagnates, unless we approximate the geometry of the domain exactly. We also observe the convergence in h for the drag and lift coefficients.

Remark 3.1.6. *We highlight that this latter method does not involve any interpolation procedure to the high order mesh and should be preferred to the first one presented.*

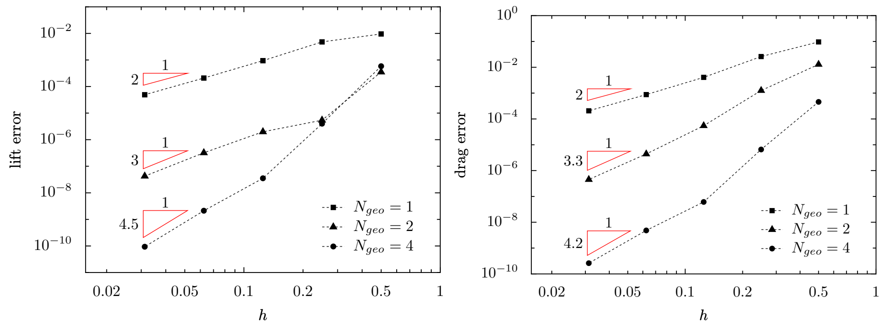
3.2 The unsteady Stokes and Navier-Stokes equations

Let us consider now the unsteady incompressible Stokes and Navier-Stokes equations written in the primitive variable formulation. Let $\Omega \subset \mathbb{R}^d$, $d = 2, 3$, be an open and bounded domain with Lipschitz continuous boundary $\partial\Omega$. We will split the boundary $\partial\Omega$ in two parts: Γ^D and Γ^N where we impose homogeneous Dirichlet and Neumann boundary conditions, respectively. We introduce the space

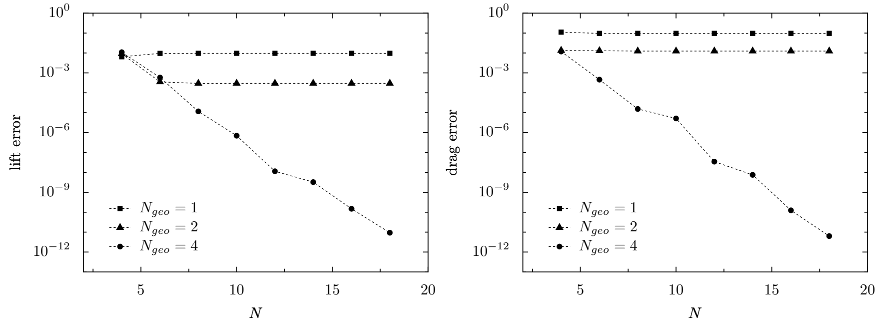
$$\mathbf{H}^1_{\Gamma^D}(\Omega) = \left\{\mathbf{v} \in \mathbf{H}^1(\Omega) : \; \mathbf{v}_{|_{\Gamma^D}} = \mathbf{0}\right\}$$

of all $\mathbf{H}^1(\Omega)$ functions with zero trace in Γ^D. Let T be a positive and fixed real. Given a divergence free datum $\mathbf{u}_0 \in \mathbf{H}^1_{\Gamma^D}(\Omega)$ and an external field $\mathbf{f} \in [L^2(0, T; H^{-1}(\Omega))]^d$, we look for the velocity field $\mathbf{u} \in \left[L^2\left(0, T; \mathbf{H}^1_{\Gamma^D}(\Omega)\right) \cap L^\infty(0, t; \mathbf{L}^2(\Omega)\right]^d$ and pressure field $p \in L^2(0, T; L^2(\Omega))$ solutions of

$$\begin{aligned}
\frac{\partial \mathbf{u}}{\partial t} - \nu\boldsymbol{\Delta}\mathbf{u} + (\mathbf{u}\cdot\boldsymbol{\nabla})\mathbf{u} + \nabla p &= \mathbf{f}, \quad \text{in } \Omega \times (0, T) && (3.24)\\
\operatorname{div}(\mathbf{u}) &= 0, \quad \text{in } \Omega \times (0, T) && (3.25)\\
\mathbf{u} &= \mathbf{0}, \quad \text{on } \Gamma^D \times (0, T) && (3.26)\\
(-p\mathbf{I} + \nu\boldsymbol{\nabla}\mathbf{u})\,\mathbf{n} &= \mathbf{0}, \quad \text{on } \Gamma^N \times (0, T) && (3.27)\\
\mathbf{u} &= \mathbf{u}_0, \quad \text{in } \Omega \times \{0\} && (3.28)
\end{aligned}$$



(a) Error using $\mathbb{P}_6 - \mathbb{P}_4$ method



(b) Error using high order geometrical elements

Figure 3.13: Convergence plot in the $\mathbf{H}^1(\Omega)$-norm for the modified Kovasznay example using high order geometrical elements and the $\mathbb{P}_N - \mathbb{P}_{N-2}$ method.

q	β_{-1}	β_0	β_1	β_2	β_3
1	1	1			
2	$3/2$	2	$-1/2$		
3	$11/6$	3	$-3/2$	$1/3$	
4	$25/12$	4	-3	$4/3$	$-1/4$

Table 3.1: Coefficients of the BDFq schemes up to order 4 of accuracy.

where ν is the viscosity parameter and $\mathbf{I}$ is the identity tensor. This problem admits a unique solution in the case $d = 2$, see Temam [86].

Remark 3.2.1. *In the case $\Gamma^N = \emptyset$ then the pressure must be sought in $L^2(0, T; L^2_0(\Omega))$ since it will be unique up to a constant.*

To obtain the weak formulation of system (3.24)-(3.28) we proceed as follows: first, we discretize the equations in time; then, concerning the space discretization, we proceed as in section 3.1 for the Stokes problem.

3.2.1 Time discretization

We start by approximating the time derivative by a *backward differentiation formula* of order q (BDFq) and linearize the nonlinear convective term with an extrapolation of order q. Given $\Delta t \in (0, T)$, we set $t_0 = 0$, $t_n = t_0 + n\Delta t$ (for any $n \geqslant 1$) and $N_T = \left[\frac{T}{\Delta t}\right]$; then

Problem 3.2.1. *For each $n \geqslant q-1$, we look for the solution $(\mathbf{u}^{n+1}, p^{n+1}) \in \mathbf{H}^1_{\Gamma^D}(\Omega) \times L^2(\Omega)$ of*

$$\begin{aligned} &\frac{\beta_{-1}}{\Delta t}\mathbf{u}^{n+1} - \nu\boldsymbol{\Delta}\mathbf{u}^{n+1} + (\boldsymbol{\beta}^n \cdot \boldsymbol{\nabla})\mathbf{u}^{n+1} + \nabla p^{n+1} = \mathbf{f}^{n+1} + \sum_{j=0}^{q-1} \frac{\beta_j}{\Delta t}\mathbf{u}^{n-j} \quad && \textit{in } \Omega \\ &\operatorname{div}(\mathbf{u}^{n+1}) = 0 && \textit{in } \Omega \end{aligned} \tag{3.29}$$

that satisfies the boundary conditions (3.26)-(3.28).

The vector $\boldsymbol{\beta}^n$ that appears in the previous formulation is the result of the linearization of the convective term and it depends on the BDF scheme used.

The coefficients β_j, $j = -1, \ldots, q-1$ are the well known coefficients associated with the BDF schemes which we briefly remind in Table 3.2.1.

3.2.2 Space discretization

Often the fluid flows are dominated by the convection and the velocity in the variational formulation needs to be stabilized. In our approach, we consider the *interior penalty* (IP) stabilization technique. One of the main advantages of this approach is that the stabilization term is independent of time derivatives and source terms, at the cost however of increasing the stencil of the finite/spectral element method. More details in the IP method can be found in Burman and Fernandez [8] to stabilize the Navier-Stokes equations and Burman and Ern [7] to stabilize steady convection-diffusion problem with an arbitrary degree space discretization.

To properly introduce the IP stabilization, let us first introduce some notation. Let $v \in H^1(\Omega)$ and $\mathbf{v} \in \mathbf{H}^1(\Omega)$ and $\mathcal{F}_I$ the set of internal faces of a triangulation $\mathcal{T}_{h,N_{\text{geo}}}$. Given a face $F \in \mathcal{F}_i$, let T_1 and T_2 the elements of $\mathcal{T}_{h,N_{\text{geo}}}$ that share F, that is, $F = T_1 \cup T_2$. We denote by v_1, v_2, respectively, $\mathbf{v}_1, \mathbf{v}_2$ the restriction of $v, \mathbf{v}$ to the element T_1 and T_2. Let $\mathbf{n}_1$ and $\mathbf{n}_2$ be the exterior normal of T_1 and T_2. Then, the *jump* of $v, \mathbf{v}$ across F is defined as

$$[\![v]\!]_F = v_1\mathbf{n}_1 + v_2\mathbf{n}_2 \tag{3.30}$$

$$[\![\mathbf{v}]\!]_F = \mathbf{v}_1 \cdot \mathbf{n}_1 + \mathbf{v}_2 \cdot \mathbf{n}_2. \tag{3.31}$$

In the case of a matricial function, like for instance $\boldsymbol{\nabla}\mathbf{v}$, we define the jump as

$$[\![\boldsymbol{\nabla}\mathbf{v}]\!]_F = \boldsymbol{\nabla}\mathbf{v}_1\mathbf{n}_1 + \boldsymbol{\nabla}\mathbf{v}_2\mathbf{n}_2.$$

The stabilization term to be added to the variational formulation reads

$$j(\boldsymbol{\beta}; \mathbf{u}, \mathbf{v}) = \sum_{F \in \mathcal{F}_I} \int_F |\boldsymbol{\beta} \cdot \mathbf{n}| \frac{h_F^2}{N^{3.5}} [\![\boldsymbol{\nabla}\mathbf{u}]\!]_F \cdot [\![\boldsymbol{\nabla}\mathbf{v}]\!]_F \, ds \tag{3.32}$$

where h_F denotes the length of the face F and N the degree of the velocity approximation. The presence of the term (3.32) increases the stencil of the standard continuous Galerkin

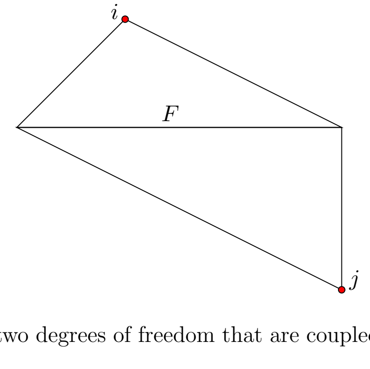


Figure 3.14: Example of two degrees of freedom that are coupled with the IP stabilization term.

formulation. This means that once we assemble the matrix with this term (and all the others from the weak formulation of the Navier-Stokes equations), this matrix has more nonzero entries than the one without the contribution of the IP term. In Figure 3.14 we observe two degrees of freedom, that are not usually coupled. However, if $\boldsymbol{\phi}_i$ and $\boldsymbol{\phi}_j$ are the functions corresponding to the degrees of freedom in the figure, the term

$$\int_F [\![\boldsymbol{\nabla}\boldsymbol{\phi}_i]\!]_F \cdot [\![\boldsymbol{\nabla}\boldsymbol{\phi}_j]\!]_F \; ds$$

is nonzero.

Before we introduce the complete weak formulation associated with system (3.29), we introduce the following notations

$$(\mathbf{u},\mathbf{v}) = \int_\Omega \mathbf{u}\cdot\mathbf{v} \; dx,$$

$$c\left(\boldsymbol{\beta};\mathbf{u},\mathbf{v}\right) = \int_\Omega \left(\boldsymbol{\beta}\cdot\boldsymbol{\nabla}\right)\mathbf{u}\cdot\mathbf{v} \; dx.$$

Using the notation for the bilinear forms (3.3), the approximation for system (3.29) by the IP method reads as

Problem 3.2.2. *For each $n \geqslant q-1$, find $(\mathbf{u}_N^{n+1}, p_N^{n+1}) \in \mathbf{V}_N \times Q_M$ such that*

$$\begin{aligned} \frac{\beta_{-1}}{\Delta t}\left(\mathbf{u}_N^{n+1},\mathbf{v}\right) + a(\mathbf{u}_N^{n+1},\mathbf{v}) + c(\boldsymbol{\beta}_N^n;\mathbf{u}_N^{n+1},\mathbf{v}) + \gamma j(\boldsymbol{\beta}_N^n;\mathbf{u}_N^{n+1},\mathbf{v}) - b(\mathbf{v},p_N^{n+1}) &= (\tilde{\mathbf{f}}^{n+1},\mathbf{v}) \\ b(\mathbf{u}_N^{n+1},q) &= 0 \end{aligned} \tag{3.33}$$

for all $\mathbf{v}\in\mathbf{V}_N$ and $q \in Q_M$,

where $\mathbf{V}_N$ and Q_M are any of the inf-sup stable spaces discussed in section 3.1 associated with $\mathcal{T}_{h,N_{\text{geo}}}$, $\tilde{\mathbf{f}}^{n+1}$ accounts for the whole right hand side the first equations of (3.29) and γ is a parameter. The vector $\boldsymbol{\beta}_N^n$ is now equal to some linear combination of previous time step known solutions

$$\boldsymbol{\beta}_N^n = \begin{cases} \mathbf{u}_N^n, & q=1 \\ 2\mathbf{u}_N^n - \mathbf{u}_N^{n-1}, & q=2 \\ 3\mathbf{u}_N^n - 3\mathbf{u}_N^{n-1} + \mathbf{u}_N^{n-2}, & q=3 \\ 4\mathbf{u}_N^n - 6\mathbf{u}_N^{n-1} + 4\mathbf{u}_N^{n-2} - \mathbf{u}_N^{n-3}, & q=4 \end{cases} \tag{3.34}$$

Remark 3.2.2. *In the case that non homogeneous Neumann boundary conditions are prescribed in Γ^N, say by a function $\mathbf{g}_N$, the variational formulation changes slightly by adding to the right hand side of the first equation of* (3.33) *the term*

$$\int_{\Gamma^N} \mathbf{g}_N\cdot\mathbf{v} \; ds$$

As we did in section 3.1.2, we consider basis function for the spaces $\mathbf{V}_N$ and Q_M, say $\mathbf{V}_N = \operatorname{span}\{\boldsymbol{\phi}_i\}$ and $Q_M = \operatorname{span}\{\psi_i\}$. Then equation (3.33) is equivalent to solve, for each $n \geqslant 1$ a system of the form

$$\begin{bmatrix} F_N & G_N \\ D_N & 0 \end{bmatrix}\begin{bmatrix} \mathbf{U}_N^{n+1} \\ \mathbf{P}_N^{n+1}\end{bmatrix} = \begin{bmatrix} \mathbf{F}_N^{n+1} \\ 0 \end{bmatrix}, \tag{3.35}$$

where $\mathbf{U}_N^{n+1}$ and $\mathbf{P}_N^{n+1}$ are the representations of $\mathbf{u}_N^{n+1}$ and p_N^{n+1} in the bases of $\mathbf{V}_N$ and Q_M, respectively. The matrices G_N and D_N are the discrete representatives of the gradient and divergence operators, just like in system (3.8). However, in this context, matrix F_N accounts for the discretization of not only the diffusive terms of the Navier-Stokes equations, but also the linearized convection, C_N, mass matrix, M_N, and possibly, the matrix arising from the IP stabilization term (which we include in C_N)

$$F_N = \frac{\beta_{-1}}{\Delta t} M_N + \nu H_N + C_N \tag{3.36}$$

Remark 3.2.3. *As already mentioned before, the imposition of Dirichlet boundary conditions is done at the algebraic level. This implies the elimination of the rows in matrix F_N but also of matrix G_N.*

3.3 Efficient preconditioning techniques

Let us recall the linear system (3.35)

$$\underbrace{\begin{bmatrix} F_N & G_N \\ D_N & 0 \end{bmatrix}}_{A_N} \begin{bmatrix} \mathbf{U}_N \\ \mathbf{P}_N \end{bmatrix} = \begin{bmatrix} \bar{\mathbf{f}} \\ 0 \end{bmatrix}. \tag{3.37}$$

where F_N is given in (3.36). We remark that system (3.37) is quite general and encompasses the discretization of the unsteady/steady Stokes/Navier-Stokes equations.

For the solution of (3.37), iterative methods are usually preferred. In the case A_N is symmetric, the conjugate gradient method is usually chosen. In the more general case, *generalized minimum residual* GMRES or Bi-CGSTAB are suitable alternatives. In these circumstances, a good preconditioner is the cornerstone to solve the linear system in a fast way.

Several preconditioners for A_N can be found in literature. Among them, we refer to multigrid [30, 21], overlapping Schwarz [62, 63, 13, **?**, 59] or block type [56, 25] preconditioners as possible choices. In the following, we analyse a family of block preconditioners for A_N and compare it to two other strategies: a direct solver using a LU factorization and a preconditioner performing an incomplete LU factorization.

3.3.1 A block type preconditioner

We start by noticing that A_N can be factorized as follows

$$A_N = \underbrace{\begin{bmatrix} I_N & 0 \\ D_N F_N^{-1} & I_N \end{bmatrix}}_{L} \underbrace{\begin{bmatrix} F_N & 0 \\ 0 & S_N \end{bmatrix}}_{D} \underbrace{\begin{bmatrix} I_N & F_N^{-1} G_N \\ 0 & I_N \end{bmatrix}}_{U}. \tag{3.38}$$

where we denote by $S_N = -D_N F_N^{-1} G_N$ the *pressure Schur complement.*

If we use the matrix $P_L = LD$ as preconditioner for A_N and a Krylov subspace method, than the matrix $P_L^{-1}A_N$ has two distinct eigenvalues, thus convergence is achieved in at most two iterations, see Murphy, Golub and Wathen [56]. However, this preconditioner is prohibitive in practice due to the presence of the pressure Schur complement. The idea here to build an effective preconditioner is to replace matrices F_N and S_N by cheap approximations, say $\tilde{F}_N$ and $\tilde{S}_N$. These approximative versions of the original operators should be chosen such that they constitute good preconditioners for F_N and S_N, respectively.

In the work of Elman and Sylvester [25], a preconditioner based on $P_R = DU$ was used as a right preconditioner together with the GMRES method to solve the steady Stokes and Navier-Stokes equations. This preconditioner has the property that the number of iterations stays bounded independently of the the mesh size h or the polynomial degree of the approximation N. The preconditioner and these results were extended to the unsteady Navier-Stokes case for $N = 2$ in Silvester, Elman, Kay and Wathen [78].

We propose a left preconditioner based on P_L and the ideas in Elman and Sylvester [25] and Silvester, Elman, Kay and Wathen [78] and extend the experiments in [78] to spectral discretizations. Let

$$P = \begin{bmatrix} \tilde{F}_N & 0 \\ D_N & \tilde{S}_N \end{bmatrix} \tag{3.39}$$

where $\tilde{F}_N$ and $\tilde{S}_N$ are suitable approximations of F_N and S_N.

The inverse of P is given by

$$P^{-1} = \begin{bmatrix} \tilde{F}_N^{-1} & 0 \\ R_N & \tilde{S}_N^{-1} \end{bmatrix} \tag{3.40}$$

where $R_N = -\tilde{S}_N^{-1} D_N \tilde{F}_N^{-1}$. If a Krylov subspace method is used to solve problem (3.37), then, at each iteration, we need to solve a system with matrix P. This means that for a given vector $(\mathbf{r}, \mathbf{s})$, we need to calculate $(\mathbf{v}, \mathbf{q})$ such that

$$\begin{bmatrix} \tilde{F}_N & 0 \\ D_N & \tilde{S}_N \end{bmatrix} \begin{bmatrix} \mathbf{v} \\ \mathbf{q} \end{bmatrix} = \begin{bmatrix} \mathbf{r} \\ \mathbf{s} \end{bmatrix}. \tag{3.41}$$

In order to solve (3.41), we follow Algorithm 3.1.

Algorithm 3.1 Steps to solve a system with matrix $\mathcal{P}$.

```
given r and s,
solve $\tilde{F}_N \mathbf{v} = \mathbf{r}$
solve $\tilde{S}_N \mathbf{q} = \mathbf{s} - D_N \mathbf{v}$
return (v, q)
```

Choice for the operator $\tilde{F}_N$

In this section we will consider $\tilde{F}_N = F_N$ and solve any systems with this matrix using a LU factorization. In the case $\alpha = \frac{\beta_{-1}}{\Delta t}$ is "big", a cheap alternative is to take $\tilde{F}_N$ as the diagonal of αM_N. However, this choice is not considered in this work. Other more robust choices are additive Schwarz or multigrid methods. The latter were used in the works by Silvester, Elman, Kay and Wathen [78] and Kay, Loghin and Wathen [50].

Choice for the operator $\tilde{S}_N$

We follow the idea of Kay, Loghin and Wathen [50] and take as approximation of the pressure Schur complement the operator $\tilde{S}_N = A_p {F_p}^{-1} M_p$, where A_p, F_p and M_p are the discretizations of the pressure operators $-\Delta$, $\alpha I - \nu\Delta + \boldsymbol{\beta}\cdot\nabla$ and I, respectively. The quantity $\boldsymbol{\beta}$ is velocity obtained after linearization of the non linear convective term of the momentum equation. If the velocity field is convection dominated, then the discretization of the convection-diffusion-reaction pressure operator should also be stabilized.

The preconditioner that we obtain with the choices of $\tilde{F}_N$ and $\tilde{S}_N$ is called *block triangular pressure convection diffusion* (BTPCD) preconditioner

$$\tilde{P} = \begin{bmatrix} F_N & 0 \\ D_N & A_p F_p^{-1} M_p \end{bmatrix}. \tag{3.42}$$

Remark 3.3.1. *In this work, we assume that the discrete pressures are continuous all over Ω. The case where we look for the pressure in $\mathcal{F}_M^{disc}(\mathcal{T}_{h,\delta})$ is not treated here. We point out a strategy that can be used in this framework. It consists in discretizing the pressure operators, $-\Delta$ and $\alpha I - \nu\Delta + \boldsymbol{\beta}\cdot\nabla$, with suitable discontinuous Galerkin approximations, see for instance Houston, Schwab and Suli [46].*

Regarding the overall computational cost of using $\tilde{S}_N$ as preconditioner, at each iteration, we have to invert the mass matrix M_p and the discrete laplacian A_p (for which efficient solvers can be chosen, for instance, *preconditioned conjugate gradient method*) and apply operator F_p.

In the case we consider a steady Stokes problem, the operator $\tilde{S}_N$ becomes simply $\frac{1}{\nu}M_p$ which is known to be a good preconditioner. Also, in the unsteady Stokes case ($\boldsymbol{\beta} \equiv 0$), the inverse of the pressure Schur complement approximation becomes $\alpha {A_p}^{-1} + \nu {M_p}^{-1}$, which is the preconditioner proposed by Cahouet and Chabard [9].

Imposition of boundary conditions

The most delicate issue in implementing the preconditioner strategy lies in the boundary conditions that are imposed to the pressure operators, namely, F_p and A_p. If we do not impose any boundary conditions to the pressure Laplace problem, the matrix A_p will be singular, thus posing problems when solving a system associated with this matrix. In order to go around this issue, we consider Dirichlet boundary conditions in some part of the boundary and homogeneous Neumann for the rest of the boundary. Since we only care about the entries of matrix A_p, we do not need to devise an actual expression for the Dirichlet boundary condition. The same has to be done to the convection-diffusion-reaction operator F_p.

We remark that in the steady Stokes/Navier-Stokes, the Dirichlet boundary conditions should be imposed at the inflow boundary, that is, where $\boldsymbol{\beta}\cdot\mathbf{n} < 0$. For unsteady problems, this should be done at $\partial\Omega_N$. We highlight that in the test cases we will present, if another part of the boundary is used, either in the steady or unsteady simulations, optimality of the preconditioner is completely lost and the number of GMRES iterations necessary to reach a certain tolerance increases tremendously while decreasing the mesh size or increasing the polynomial degree.

According to Remark 2.4.1, the imposition of Dirichlet boundary conditions is done by eliminating the values in the row corresponding to the all Dirichlet degrees of freedom and putting the value 1 in the diagonal element. We will do a similar approach for the pressure operators A_p and F_p. First, to retain a good scaling with respect to the parameters $\boldsymbol{\beta}$, α and ν, we keep the previous diagonal element of the matrix instead of replacing it with 1. We also symmetrize the matrices, that is, if we perform the previous algorithm in row i^*, then the same is applied to column i^*. Like this, system A_p remains symmetric and positive definite.

One last detail needs to be mentioned. For the steady problems, the convective term should not be incorporated as a contribution to the Dirichlet entries in matrix F_p. If incorporated, then the optimality of the preconditioner is lost, see section 3.3.2.

Regarding the velocity, we proceeded as in section 2.4, Remark 2.4.1.

Inner loop solvers

As mentioned before, for each iteration of a Krylov subspace solver we have to solve a system like (3.41). This operation translates in solving three systems, with matrices $\tilde{F}_N$, M_p and A_p. We use LU factorizations to invert the three of them.

An alternative to invert the discrete pressure operators is the preconditioned conjugate gradient method. This is a valid choice due to the fact that these matrices are s.p.d. Suitable preconditioners for this method can be obtained through incomplete Cholesky factorizations or multigrid method. We remark that the preconditioners for M_p and A_p only need to be calculated once and then reused at each iteration.

3.3.2 Numerical results

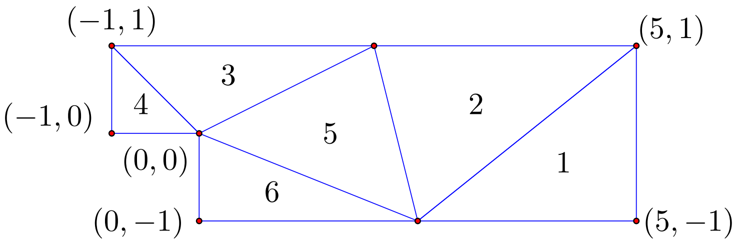
We show now some numerical simulations conducted to test the previous preconditioning strategy.

A steady Navier-Stokes problem

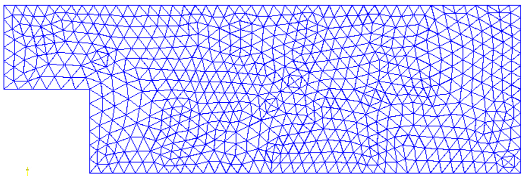
The first numerical benchmark we consider is the steady Navier-Stokes equations

$$\begin{aligned} -\nu\boldsymbol{\Delta}\mathbf{u} + (\mathbf{u}\cdot\boldsymbol{\nabla})\mathbf{u} + \nabla p &= \mathbf{0}, \quad \text{in } \Omega \\ \operatorname{div}(\mathbf{u}) &= 0, \quad \text{in } \Omega \end{aligned} \tag{3.43}$$

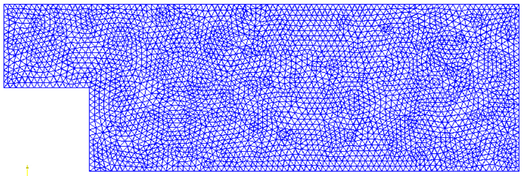
applied to the backward facing step problem, where Ω is depicted in Figure 3.15. Regarding



(a) $h = 2$



(b) $h = 0.125$



(c) $h = 0.0625$

Figure 3.15: Domain description for problem (3.43).

boundary conditions, we impose homogeneous Dirichlet conditions for the velocity everywhere, except in the inflow and outflow boundaries. At the inflow we impose a parabolic profile

$$\mathbf{u} = [y(1-y), 0]^T$$

and at the outflow, homogeneous Neumann boundary conditions. In order to solve the non

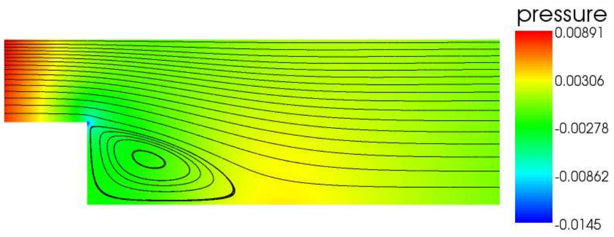


Figure 3.16: Pressure profile with streamlines for the backward facing step problem using $\nu = 0.005$, $h = 0.125$ and $N = 5$.

linearity of the previous system, we will use fixed point iterations, meaning, for $k > 0$, we solve the system

$$\begin{aligned} -\nu \boldsymbol{\Delta} \mathbf{u}^k + (\mathbf{u}^{k-1} \cdot \boldsymbol{\nabla})\mathbf{u}^k + \nabla p^k &= \mathbf{0}, \quad \text{in } \Omega & (3.44) \\ \operatorname{div}(\mathbf{u}^k) &= 0, \quad \text{in } \Omega & (3.45) \end{aligned}$$

for $\mathbf{u}^k$ and p^k. At the space discretization level, we consider the $\mathbb{P}_N - \mathbb{P}_{N-1}$ method up to degree $N = 7$. The bases of the spaces for the discrete velocity and pressure are built with standard Lagrange polynomials associated with Fekete points.

The stopping criteria for the fixed points scheme is the

$$\left(\left\| \mathbf{u}_N^{k-1} - \mathbf{u}_N^k \right\|_2^2 + \left\| p_N^{k-1} - p_N^k \right\|_2^2 \right)^{1/2} < 10^{-6} \qquad (3.46)$$

where $\mathbf{u}_N^k$ and p_N^k denote the spectral element approximations of $\mathbf{u}^k$ and p^k.

Regarding the linear solver, we use the GMRES method. The stopping criteria for the linear solver is when the relative residual in the ℓ_2-norm goes below 10^{-6}. The initial guess for each linear solve is the previous fixed point iteration. We define $\mathbf{u}_N^0$ and p_N^0 as zero vectors.

Table 3.3 shows the maximum number of iterations required to solve an iteration of the fixed point method over all fixed point iterations performed and Table 3.2 reports the number of degrees of freedom for the problems that were solved.

		N					
		2	**3**	**4**	**5**	**6**	**7**
h	2.0	50	101	170	257	362	485
	1.0	112	242	423	655	938	1272
	0.5	347	789	1417	2231	3231	4417
	0.25	1705	4041	7415	11827	17277	23765
	0.125	7040	16947	31342	50225	73596	101455

Table 3.2: Number of degrees of freedom for the problems associated with the backward facing step with varying h and N.

Since the initial guess is the zero vector, the left table of Table 3.3 shows the iteration counts for a pure Stokes problem. In this case, the Schur complement is preconditioned only by a scaled pressure mass matrix, see section 3.3.1. From these results, the number of iterations stays bounded independently of the viscosity ν, the mesh size h or the polynomial degree of the approximation N.

Regarding the right table of Table 3.3, it concerns the Navier-Stokes problem. Similar conclusions can be drawn in this case, although now, there is a mild dependence of the number of iterations with respect to ν. A characteristic of this preconditioner that can be observed from Table 3.3 is that if we reduce the mesh size or increase the polynomial degree, the number of iterations decreases. This behavior occurs because $\widetilde{S}_N$ becomes a good approximation of the continuous pressure operator, as the mesh is refined or the polynomial degree is increased. These results are in agreement with what had been reported by Kay, Loghin and Wathen [50] and Kay and Lungu [51].

An unsteady Navier-Stokes problem

Let us consider now the unsteady Navier-Stokes equations (3.24)-(3.28) applied to the same backward facing step problem. The boundary conditions are the same as for the steady Navier-Stokes problem, with the exception of the inflow boundary condition. In this case, the new boundary conditions are

$$\mathbf{u} = \left[4\left(1 + (t-1)\chi(t \leqslant 1)\right), 0\right]^T$$

where χ denotes the characteristic function, ie, given a set X,

$$\chi(x) = \begin{cases} 1 & \text{if } x \in X, \\ 0 & \text{otherwise.} \end{cases}$$

The mesh that we use is depicted in Figure 3.17. As space-time discretization, we consider the formulation (3.33) with $q = 1$ and $\gamma = 0$. Regarding the the polynomial bases for velocity and pressure, we consider again the Lagrange basis associated with the Fekete point set. At the algebraic level the only difference between the linear system/precondi-

ν	**h**	**N**					
		2	**3**	**4**	**5**	**6**	**7**
	2.0	9	22	27	29	33	38
	1.0	17	24	25	27	29	30
0.1	0.5	22	28	31	32	37	39
	0.25	20	21	21	21	21	21
	0.125	20	21	20	21	21	21
	2.0	9	22	26	28	33	38
	1.0	17	24	24	26	27	29
0.01	0.5	22	27	29	30	34	34
	0.25	19	20	20	19	19	19
	0.125	18	19	18	18	18	18
	2.0	9	22	26	27	37	38
	1.0	17	24	24	26	27	29
0.005	0.5	22	27	29	30	34	34
	0.25	19	20	20	19	19	19
	0.125	18	19	18	18	18	18

N					
2	**3**	**4**	**5**	**6**	**7**
16	47	51	67	59	61
26	47	56	54	59	59
45	56	56	58	58	60
43	39	35	35	35	34
34	33	32	31	31	30
19	81	105	142	150	128
51	83	119	147	143	111
59	103	95	105	110	120
60	65	59	56	56	55
58	56	54	53	52	50
19	82	146	204	195	204
60	114	149	179	199	233
79	198	179	146	150	167
75	72	70	70	68	65
71	68	59	57	56	55

Table 3.3: GMRES iteration count for the steady Stokes (left) and Navier-Stokes (right) problems associated with the backward facing step with varying ν, h and N.

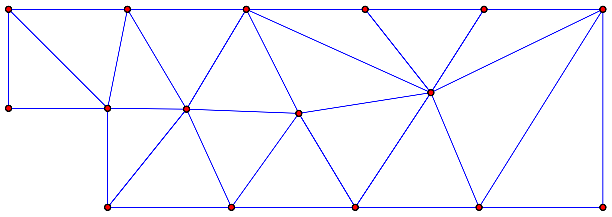


Figure 3.17: Only mesh used for solving the unsteady backward facing step problem.

ν	Δt	2	3	4	5	6	7
0.1	0.5	12	12	13	13	14	15
	0.1	12	12	12	13	13	14
	0.01	16	14	14	14	15	15
	0.001	17	22	21	21	24	23
0.01	0.5	12	11	12	11	11	12
	0.1	15	12	12	12	13	13
	0.01	16	18	18	17	18	17
	0.001	17	24	27	28	31	31
0.005	0.5	13	12	12	12	12	12
	0.1	15	13	12	12	13	13
	0.01	16	19	19	19	20	21
	0.001	17	24	27	29	36	36

Table 3.4: GMRES maximum number of iterations for the unsteady Stokes problem associated with the backward facing step.

tioner obtained in the steady simulations and now, is that a scaled mass matrix is present in the F_N block, as well as in F_p.

In Table 3.4 we present the iteration counts for the Stokes version of this problem. In this case, the preconditioner for the pressure Schur complement takes the form $\tilde{S}_N = \Delta t^{-1} A_p^{-1} + \nu M_p^{-1}$. We observe that, for small time steps, the number of iterations increases mildly with the polynomial degree and the growth rate is dependent on the viscosity also.

We now turn to the complete version of the preconditioner, where F_p involves the discretization of the whole operator $\alpha I - \nu\Delta + \boldsymbol{\beta}\cdot\nabla$. We show in Table 3.5 the variation of the iterations counts depending on the time step, viscosity and polynomials degree. Also here we observe that the number of iterations stays bounded, independently of viscosity, time step and polynomial degree. As a final remark, notice the low average iteration counts that decrease while decreasing Δt and stay bounded while increasing the polynomial degree.

Regarding the robustness of the preconditioner (3.39) in terms of viscosity, time step and mesh size, we refer the reader to the work of Silvester, Elman, Kay and Wathen [78]. Due to the similarity of their preconditioner and the one we propose here, we expect the same behavior. In their work, it is shown that using the $\mathbb{P}_2 - \mathbb{P}_1$ method as space discretization, the number of iterations stays bounded independently of the time step and mesh size.

Comparison with other preconditioners

In this section, we consider a fixed viscosity $\nu = 0.1$ and compare the preconditioner (3.39) with two others strategies: a LU factorization (which translates in practice in solving the system (3.37) with this type of factorization) and an incomplete LU factorization, with fill in 3, see [74]. The LU factorization is calculated with the KLU algorithm, see

$\boldsymbol{\nu}$	$\boldsymbol{\Delta}$**t**	**2**	**3**	**4**	**5**	**6**	**7**
0.1	0.5	14	14	14	15	15	16
	0.1	12 (7)	13 (7)	14 (8)	14 (8)	15 (8)	15 (9)
	0.01	14 (4)	13 (4)	13 (4)	14 (4)	14 (4)	15 (4)
	0.001	14 (3)	16 (2)	15 (2)	16 (2)	16 (2)	16 (2)
0.01	0.5	24	21	19	17	17	17
	0.1	16 (10)	16 (10)	15 (9)	15 (9)	15 (9)	16 (9)
	0.01	14 (5)	16 (5)	16 (5)	17 (5)	17 (5)	18 (5)
	0.001	14 (2)	18 (3)	18 (2)	20 (3)	21 (3)	22 (3)
0.005	0.5	31	37	35	29	23	22
	0.1	17 (12)	18 (12)	19 (12)	18 (11)	17 (10)	18 (10)
	0.01	14 (6)	17 (5)	17 (5)	10 (5)	19 (6)	21 (6)
	0.001	14 (3)	18 (3)	19 (3)	21 (3)	22 (3)	24 (3)

Table 3.5: GMRES maximum number of iterations for the unsteady Navier-Stokes problem associated with the backward facing step. In parenthesis, we provide the average number of iterations over all time steps for different values of $\nu, \Delta t$ and N.

[17, 80]. We compare these three solution strategies in terms of the time to calculate the preconditioner and the time to solve the linear system, both regarding the mesh size h and the polynomial degree N. We highlight that our results are obtained using only one processor. The use of more processors and parallel implementations of the LU/ILU(3) factorization are not discussed in this work.

In the case of the LU and incomplete LU factorization, that we hereby denote by ILU(3), the preconditioner is calculated directly from the matrix of system (3.37). However, for the BTPCD preconditioner, at each iteration of the fix point method, the cost of constructing this preconditioner is dependent on the calculation of the LU factorization of the F_N block and the assembly of the pressure convective term plus the construction of matrix F_p.

Tables 3.6 show the maximum number of iterations used by the GMRES method, N_{it}, to solve the steady Navier-Stokes problem (3.44)-(3.45), as well as the time to calculate the preconditioner, t_{prec}, and the maximum time to solve the linear system, t_{solve}. These results were obtained using a Dual Core AMD Opteron(tm) Processor 270, 2GHz cpu and 3Gb of RAM memory.

We observe from Tables 3.6 that for the size of problems we tested ($\lesssim$ 100000 degrees of freedom), the LU factorization applied to the linear system proves to be the fastest solution strategy. However, for bigger problems, this factorization takes too much memory and time to calculate and stops being an acceptable option. The same conclusion can be taken from the results regarding the ILU(3) factorization as preconditioner. In this case, the cost of the calculation of the preconditioner is where the most amount of time is spent.

Regarding the number of iterations used by each algorithm, the ILU(3) preconditioner together with GMRES, uses a number of iterations that stays bounded when we increase

	N																	
	2			**3**			**4**			**5**			**6**			**7**		
h	N_{it}	t_{prec}	t_{solve}	N_{it}	t_{prec}	t_{solve}	N_{it}	t_{prec}	t_{solve}	N_{it}	t_{prec}	t_{solve}	N_{it}	t_{prec}	t_{solve}	N_{it}	t_{prec}	t_{solve}
2.0	16	0.01	0.04	42	0.01	0.12	54	0.02	0.13	67	0.02	0.23	62	0.04	0.3	61	0.1	0.5
1.0	26	0.01	0.05	55	0.01	0.17	58	0.02	0.27	56	0.03	0.43	57	0.06	0.69	59	0.1	1.07
0.5	45	0.02	0.15	58	0.02	0.47	57	0.05	0.9	57	0.12	1.52	60	0.15	2.53	58	0.27	3.66
0.25	45	0.04	0.76	41	0.1	1.72	35	0.23	3.01	35	0.46	5.37	35	1.16	9.08	34	1.81	12.39
0.125	35	0.19	2.56	33	0.77	6.54	33	1.59	12.83	32	3.97	23.41	31	6.5	33.75	30	11.41	49.24
0.0625	30	1.78	8.95	30	5.53	24.58	30	13.93	47.8	30	25.64	84.45						

	N																	
	2			**3**			**4**			**5**			**6**			**7**		
h	N_{it}	t_{prec}	t_{solve}	N_{it}	t_{prec}	t_{solve}	N_{it}	t_{prec}	t_{solve}	N_{it}	t_{prec}	t_{solve}	N_{it}	t_{prec}	t_{solve}	N_{it}	t_{prec}	t_{solve}
2.0	1	0.01	0.01	1	0.02	0.01	1	0.05	0.01	1	0.14	0.01	1	0.36	0.01	1	0.88	0.02
1.0	2	0.02	0.01	3	0.09	0.01	2	0.42	0.02	3	1.59	0.02	3	4.73	0.03	2	11.82	0.04
0.5	5	0.1	0.01	5	1.09	0.02	5	6.43	0.05	5	24.79	0.12	5	75.07	0.24	5	189.6	0.43
0.25	9	1.39	0.05	10	15.85	0.22	10	90.64	0.68	10	353.99	1.68	10	1033.4	2.72	10	$> 10^5$	5.13
0.125	18	7.97	0.51	21	92.32	2.18	21	519.48	6.78	22	1887.45	13.42						
0.0625	51	30.28	4.42	62	398.59	27.56	62	2144.56	81.86									

	N											
	2		**3**		**4**		**5**		**6**		**7**	
h	t_{prec}	t_{solve}	t_{prec}	t_{solve}	t_{prec}	t_{solve}	t_{prec}	t_{solve}	t_{prec}	t_{solve}	t_{prec}	t_{solve}
2.0	0.01	0.01	0.01	0.01	0.01	0.01	0.02	0.01	0.02	0.01	0.04	0.01
1.0	0.01	0.01	0.01	0.01	0.02	0.01	0.03	0.01	0.06	0.02	0.1	0.02
0.5	0.02	0.01	0.02	0.01	0.05	0.01	0.11	0.02	0.21	0.03	0.37	0.04
0.25	0.04	0.01	0.15	0.02	0.38	0.04	0.79	0.08	1.68	0.13	2.95	0.21
0.125	0.28	0.03	1.62	0.08	3.78	0.26	7.5	0.44	14.35	0.65	23.52	0.92
0.0625	2.66	0.13	11.4	0.37	33.03	0.98	58.98	1.48				

Table 3.6: Timings (in seconds) to calculate preconditioners, solve the linear system and the number of GMRES iterations. BTPCD preconditioner (top), ILU(3) factorization (middle) and LU factorization (bottom).

the polynomial degree. The same does not happen when the mesh size is decreased. In this case, the number of iterations increases.

3.4 Yosida algebraic factorization methods

An efficient solution technique of the Navier-Stokes equations are splitting methods, of differential or algebraic type, see [40, 41, 39, 69, 68, 75] for a few references on this kind of methods. The former methods split the differential operators of the equations at hand, while the latter attempts to split a linear system like (3.35) using inexact block LU factorizations. In this section we present a class of algebraic factorization methods, named as *Yosida-q* schemes.

3.4.1 The Yosida-q schemes

The starting point of the Yosida schemes is the factorization (3.38). The idea behind the first Yosida scheme, introduced by Quarteroni, Saleri and Veneziani [69], was to approximate matrix F_N^{-1} in the pressure Schur complement S_N by a second order in time approximation

$$F_N^{-1} \approx \frac{\Delta t}{\beta_{-1}} M_N^{-1}.$$

This leads to the approximate matrix $\tilde{A}_N$ given by

$$\tilde{A}_N = \begin{bmatrix} F_N & 0 \\ D_N & -\frac{\Delta t}{\beta_{-1}} D_N M_N^{-1} G_N \end{bmatrix} \begin{bmatrix} I_N & F_N^{-1} G_N \\ 0 & I_N \end{bmatrix}.$$

It was seen by Quarteroni, Saleri and Veneziani [68] that this scheme applied to the unsteady Stokes equations, together with a BDF2 time discretization leads to second order in time convergence for the velocity, order 3/2 for the pressure and unconditional stability.

Remark 3.4.1. *Replacing the pressure Schur complement by* $S_N^{app} = -\frac{\Delta t}{\beta_{-1}} D_N M_N^{-1} G_N$, *called* approximate pressure Schur complement, *allows to reduce the computational cost to solve system* (3.37), *while introducing a splitting error of the same order as the time discretization used for the Navier-Stokes equations. Since* S_N^{app} *is s.p.d. we can use the preconditioned conjugate gradient method to efficiently invert it. Moreover, in the case matrix* M_N *is lumped, its inversion is very cheap.*

Later versions of this first Yosida scheme, now called *Yosida-2* scheme, have been proposed and improved the order of convergence in time for velocity and pressure. By introducing a matrix J_N in the inexact block LU factorization

$$\tilde{A}_N = \begin{bmatrix} F_N & 0 \\ D_N & -\frac{\Delta t}{\beta_{-1}} D_N M_N^{-1} G_N \end{bmatrix} \begin{bmatrix} I_N & F_N^{-1} G_N \\ 0 & J_N \end{bmatrix}.$$

and choosing it carefully, one can obtain schemes of order q for the velocity and $q-1/2$ for the pressure, named *Yosida-q*. Such choice is based on the minimization of the splitting error originated from approximating A_N with $\tilde{A}_N$, see Saleri and Veneziani [75], Gervasio, Saleri and Veneziani [34] and Gervasio [33].

All three Yosida schemes differ only in the expression of matrix J_N. While for Yosida-*2* it is equal to the identity matrix, the higher order versions take more complicated expressions. If we define

$$B_N = -D_N \frac{\Delta t}{\beta_{-1}} M_N^{-1} F_N \frac{\Delta t}{\beta_{-1}} M_N^{-1} G_N$$

then for Yosida-*3*

$$J_N = B_N^{-1} S_N^{app}. \tag{3.47}$$

The fourth order version of the Yosida schemes, Yosida-*4*, is obtained by replacing B_N in (3.47) by

$$\hat{B}_N = B_N (S_N^{app})^{-1} B_N + B_N + D_N \left(\frac{\Delta t}{\beta_{-1}} M_N^{-1} F_N \right)^2 \frac{\Delta t}{\beta_{-1}} M_N^{-1} G_N.$$

Though appearing complicated to calculate, the three Yosida schemes can be summarized in Algorithm 3.2.

Algorithm 3.2 A step of the Yosida algorithm.

```
given $\mathbf{f}$ and $\mathbf{g}$,
solve $F_N \tilde{\mathbf{u}} = \mathbf{f}$
solve $S_N^{app} \tilde{\mathbf{p}} = \mathbf{g} + D_N \tilde{\mathbf{u}}$
if $q > 2$ then
    solve $\mathbf{z} = B_N \tilde{\mathbf{p}}$
    solve $S_N^{app} \mathbf{p} = \mathbf{z}$
    if $q = 4$ then
        compute $\mathbf{p}_B = B_N \mathbf{p} + \mathbf{z} + D_N \left( \frac{\Delta t}{\beta_{-1}} M_N^{-1} F_N \right)^2 \frac{\Delta t}{\beta_{-1}} M_N^{-1} G_N \tilde{\mathbf{p}}$
        solve $S_N^{app} \mathbf{p} = \mathbf{p}_B$
    end if
else
    $\mathbf{p} = \tilde{\mathbf{p}}$
end if
$F_N(\mathbf{u} - \tilde{\mathbf{u}}) = -G_N \mathbf{p}$
return $(\mathbf{u}, \mathbf{p})$
```

A complete analysis on the convergence properties of all Yosida schemes is provided in [33] for a time-dependent Stokes problem.

Remark 3.4.2. *The use of the pressure operator arising from the Yosida-3 method,*

$$\mathcal{P}_{schur} = S_N^{app} B_N^{-1} S_N^{app},$$

has been studied as a preconditioner for the pressure Schur complement in Gauthier, Saleri and Veneziani [32].

3.4.2 Preconditioners for the approximate pressure Schur complement

The systems to solve appearing in algorithm 3.2 are either with matrix F_N or with the approximate pressure Schur complement. The computational efficiency of the Yosida schemes is dependent on efficient solvers for the latter operator, that is, in solving equations of the form

$$S_N^{app}\mathbf{p} = \mathbf{q}. \tag{3.48}$$

As we already mentioned, the matrix S_N^{app} is s.p.d. and therefore, techniques as Cholesky factorizations or the preconditioned conjugate gradient method should be used. In the former case, the matrix S_N^{app} needs to be built explicitly, which is a costly operation, even in the case M_N is lumped.

In this section we consider several preconditioners for S_N^{app} and report the number of iterations it takes to solve a system like (3.48) with the preconditioned conjugate gradient method.

Let us consider matrices D_N, M_N and G_N arising from the discretization of the Navier--Stokes used in the previous section. In Figure 3.18 we report the number of iterations using several preconditioners so that the initial residual is reduced by a factor of 10^{-10}. A few remarks are in order about the building blocks for the considered preconditioners. First, the matrix $M_{N,lumped}$ is obtained from the mass matrix M_N by summation of all the elements in the rows into the diagonal. Second, the matrix A_p is the matrix associated with the Laplace operator for the pressure; the Neumann boundary conditions of the velocity operator are applied to A_p as Dirichlet ones, the system is symmetrized (row and column elimination) and the diagonal element from the original A_p is kept. The matrix $A_{p,fe}$ is obtained the same way as A_p except we use the $\mathbb{P}_1$ function space from section 2.3. Regarding the factor M_N in the approximate Schur complement, it needs to be inverted at each iteration of the PCG method. For this, we use the PCG method with $M_{N,lumped}$ as preconditioner.

We start by remarking that, without the use of any preconditioner, the conjugate gradient method takes $\mathcal{O}(N^2)$ iterations to converge when applied to equation (3.48), see Figure 3.18. This is to be expected since the approximate Schur complement behaves like the discretization of a Laplace operator. From section 2.4, we know that its condition number number grows like $\mathcal{O}(N^4)$. Therefore, the conjugate gradient method converges in $\mathcal{O}(\sqrt{N^4}) = \mathcal{O}(N^2)$ iterations.

The use of the diagonal based preconditioners does not change this behavior. The finite element preconditioner $A_{p,fe}$ also reports a growth of $\mathcal{O}(N^2)$ for the iteration count and this is probably due to the high aspect ratio of the elements of the mesh created in the reference element. However, it reduces the number of iterations from the non-preconditioned version by a factor of 3. On the other side, using A_p as preconditioner is a good choice. The

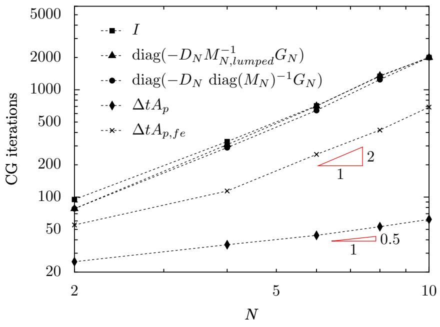


Figure 3.18: Comparison between several preconditioners for S_N^{app} ($h = 0.125$).

$N \setminus h$	1.0	0.5	0.25	0.125	0.625	0.03125
2	3	16	17	25	30	31
4	16	27	31	36	37	
6	21	38	41	44	43	
8	27	49	50	54	51	
10	33	60	59	62		

Table 3.7: Number of iterations for the $\Delta t A_p$ preconditioner, varying N and h.

dependence on the number of iterations reduces to $\mathcal{O}(\sqrt{N})$, see Figure 3.18. We have no explanation for this at the moment. Also, regarding the dependence of this preconditioner in terms of the mesh size, we can see from Table 3.7 that it remains bounded.

3.4.3 Convergence tests

Let $\Omega = (-1,1)^2$ and $t \in [0,1]$. We consider Γ^N as the lower edge of the domain Ω and $\Gamma^D = \partial\Omega \setminus \Gamma^N$. We consider the Navier-Stokes equations as in (3.24)-(3.25) and determine the forcing term as well as the proper Dirichlet and Neumann boundary conditions so that

$$\begin{aligned} \mathbf{u}(x,y,t) &= [\sin(x)\sin(y+t), \cos(x)\cos(y+t)]^T \\ p(x,y,t) &= \cos(x)\sin(y+t) \end{aligned}$$

is the exact solution of that system of equations.

We define the norms in which the error is going to be calculated

$$E_u = \left(\Delta t \sum_{n=0}^{N_T} \|\mathbf{u}(t_n) - \mathbf{U}_N^n\|^2_{\mathbf{H}^1(\Omega)} \right)^{1/2}$$

and

$$E_p = \left(\Delta t \sum_{n=0}^{N_T} \|\mathbf{p}(t_n) - \mathbf{P}_N^n\|^2_{L^2(\Omega)} \right)^{1/2}.$$

where $(\mathbf{U}_N^n, \mathbf{P}_N^n)$ denotes the numerical solution obtained with the Yosida schemes. We plot

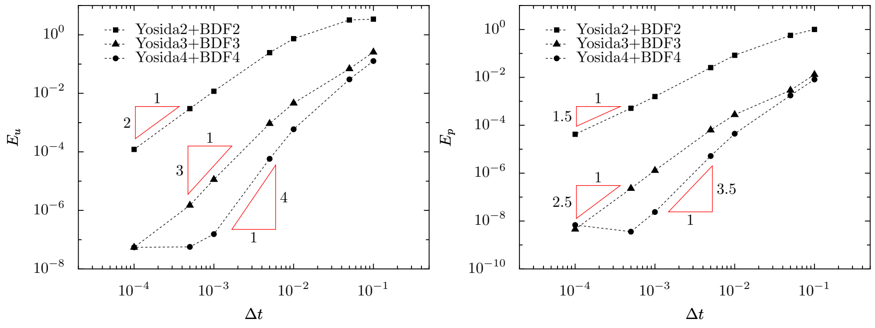


Figure 3.19: Convergence analysis using the $\mathbb{P}_{10} - \mathbb{P}_9$ element and $\nu = 0.01$.

in Figure 3.19 the quantities E_u and E_p. In Figure 3.20 we also plot the $L^\infty(0,1;L^2(\Omega))$-norm of the divergence of the velocity. These results agree with the rates predicted for the

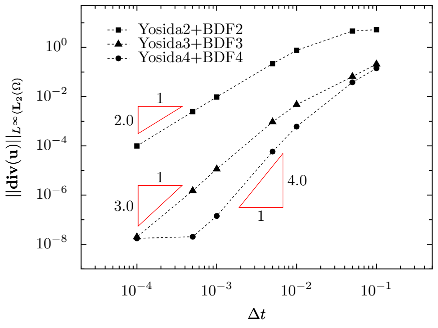


Figure 3.20: Decay of the divergence while reducing the time step.

time-dependent Stokes equations in [33].

Remark 3.4.3. *The curves in Figures 3.20 and 3.19 do not go below* 10^{-8}. *This is because of the accumulation of spacial error during the time integration. Nevertheless, the results we obtain confirm the convergence rates expected.*

3.5 Conclusion

We have presented several algorithms to solve the incompressible steady/unsteady Stokes/Navier-Stokes equations. Regarding the steady equations, the preconditioning strategy is very efficient in the test cases performed, although we showed that a LU factorization is a faster solver for the problems of the size we considered.

For the unsteady equations, two strategies were presented: one to precondition the whole system and another to directly solve the linear system by inexact block LU factorizations. Both have advantages and disadvantages. For instance, the block preconditioning strategy is strongly dependent on finding proper boundary conditions for the pressure operator. In our case, this seems to be solved, but for more general boundary conditions, one would need to derive the proper conditions. This is not the case if we consider either a direct method or an incomplete LU preconditioner, but this approach becomes impractical with the increasing size of the linear systems to solve, at least using one processor only. A parallel LU solver might be competitive with the parallel version of the BTPCD preconditioner but this is out of the scope of the present work.

The algebraic factorization methods we present do not suffer from the issue of finding appropriate boundary conditions. Once the equations are discretized (boundary conditions included, whatever they are) and in linear form, the block factorization takes care of solving the system. Moreover, the approximate Schur complement that appears does not change in time and one could even lump the mass matrix to obtain a fast solver for this operator.

The analysis in this chapter is restricted to two dimensional problems. The extension of the above algorithms to three dimensional flows in terms of discretization strategy, high order geometrical elements, algebraic factorization methods and the BTPCD preconditioning strategy is an on going project. In terms of implementation, the work is already done regarding the discretization spaces, but the high order geometrical transformation is not yet implemented, nor a 3D high order mesh generator. Another topic of interest for future work is to use a cheaper operator $\tilde{F}_N$ that maintains the properties of the BTPCD preconditioner, or at least, to keep the number of iterations of GMRES within reasonable bounds. From this perspective, one could also consider the Yosida-q as preconditioners, by replacing the matrices F_N and S_N^{app} by suitable approximations.

Chapter 4

The incompressible Navier-Stokes equations in a moving domain

The formulation of initial boundary value problems in a moving domain is typically done in the so called *Arbitrary Lagrangian Eulerian* (ALE) framework. Whether we are dealing with the unsteady Navier-Stokes equations or more general evolutionary problems, this framework allows to keep track of the domain's deformation. The main goal of this chapter is to propose strategies for the ALE method to be effectively used in combination with high degree space discretization methods like those that we have addressed in the previous chapters.

The chapter is organized as follows: in section 4.1, we present the Arbitrary Lagrangian Eulerian formulation; section 4.2 is dedicated to write the incompressible Navier-Stokes equations in the new ALE framework and to derive the weak formulation for this system of equations. In section 4.3.1, the construction of the high order map is addressed; we propose a high order ALE map using a high degree polynomial description of the boundary of the computational domain and the Laplace operator. This construction will only be done for 2D geometries and triangular meshes. As we will see, the construction of the map relies on a straight edge mesh in the reference domain. The map we propose also has the property of conserving the triangular shape of the interior elements of the mesh. Also in this section, we present the space-time discretization we consider for the problem at hand. In section 4.3.4 we show that the schemes we advocate satisfy the Geometric Conservation Law, as well as the equivalent Yosida-q methods. Finally, in the last section, some numerical results are provided regarding the methods proposed.

4.1 The Arbitrary Lagrangian Eulerian framework

Let us denote by Ω_{t_0} a reference configuration, for instance, the domain at time $t = t_0$ in which we want to solve some evolutionary differential equation. The position of a point in the current domain Ω_t, $t > t_0$, is denoted by $\mathbf{x}$ (in the Eulerian coordinate system) and by $\mathbf{Y}$ in the reference domain Ω_{t_0}. The system's evolution is studied in the interval

$I = [t_0, T]$.

In practical situations, like for instance, the flow inside a portion of a compliant artery, we have to calculate the solution of the fluid equations in Ω_t (or an approximation of it). Since the boundaries of the domain are changing with $t \in I$, Eulerian variables are not usable. An alternative is to use a Lagrangian coordinate system. In this approach, we consider a family of mappings

$$\mathcal{L}_t : \Omega_{t_0} \longrightarrow \Omega_t, \quad \mathbf{x}(\mathbf{Y}, t) = \mathcal{L}_t(\mathbf{Y}), \quad t \in [t_0, T] \tag{4.1}$$

and follow the trajectory of a particle $\xi \in \Omega_{t_0}$ in time. However, this procedure is undesirable in some cases, as for instance, blood flow in an artery. In this scenario, we need artificial boundaries in the inlet and outlet of the artery (otherwise, we would have to simulate the whole cardiovascular system), see Figure 4.1. If the Lagrangian approach is

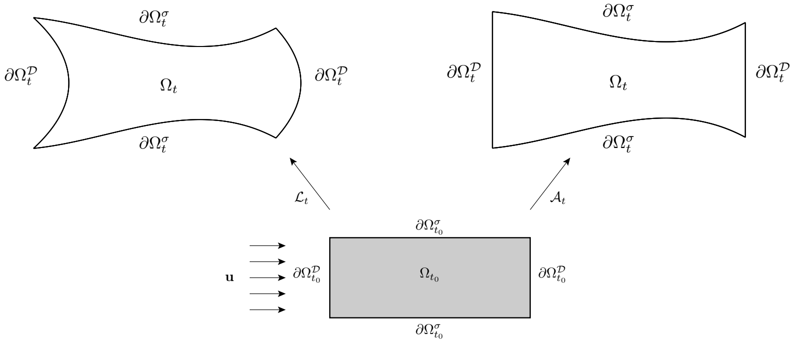

$\partial\Omega_t^\sigma$
$\partial\Omega_t^\mathcal{D}$
Ω_t
$\partial\Omega_t^\mathcal{D}$
$\partial\Omega_t^\sigma$
$\partial\Omega_t^\sigma$
$\partial\Omega_t^\mathcal{D}$
Ω_t
$\partial\Omega_t^\mathcal{D}$
$\partial\Omega_t^\sigma$
$\mathcal{L}_t$
$\mathcal{A}_t$
$\partial\Omega_{t_0}^\sigma$
$\mathbf{u}$
$\partial\Omega_{t_0}^\mathcal{D}$
Ω_{t_0}
$\partial\Omega_{t_0}^\mathcal{D}$
$\partial\Omega_{t_0}^\sigma$

Figure 4.1: Reference domain at time $t = t_0$ (bottom) and computational domain at time t (top). On the top right figure, we use a Lagrangian coordinate system and on the top left, a ALE coordinate system.

used, these boundaries will be transported and the domain might deform too much, if the time interval is too big. The other situation in Figure 4.1 is preferred since it maintains the position of the fictitious boundaries and allows for other parts of the boundary of the domain to change in time. This is the motivation to use the Arbitrary Lagrangian Eulerian approach, see [47, 22, 71, 57, 67] for a few references in this subject.

Let us introduce a family of mappings $\mathcal{A}_t$ such that, for each t associates a point $\mathbf{Y} \in \Omega_{t_0}$ to a point $\mathbf{x} \in \Omega_t$:

$$\mathcal{A}_t : \Omega_{t_0} \longrightarrow \Omega_t, \quad \mathbf{x}(\mathbf{Y}, t) = \mathcal{A}_t(\mathbf{Y}), \quad t \in [t_0, T]. \tag{4.2}$$

The mapping $\mathcal{A}_t$ is assumed to be an homeomorphism in $\overline{\Omega}_{t_0}$, i.e., $\mathcal{A}_t$ is a continuous bijection from the closure $\overline{\Omega}_{t_0}$ onto $\overline{\Omega}_t$, as well as it's inverse, from $\overline{\Omega}_t$ onto $\overline{\Omega}_{t_0}$. We also assume that the application

$$t \mapsto \mathbf{x}(\mathbf{Y}, t), \quad \mathbf{Y} \in \Omega_{t_0}$$

is differentiable almost everywhere in I. The application $\mathcal{A}_t$ is called *ALE map*.

Often, we need to transport functions defined in the reference configuration Ω_{t_0}, to Ω_t and vice-versa. Let $f : \Omega_t \times I \longrightarrow \mathbb{R}$ be a function defined in the Eulerian frame, and $\hat{f} := f \circ \mathcal{A}_t$ the corresponding function defined in the ALE framework, defined as

$$\hat{f} : \Omega_{t_0} \times I \longrightarrow \mathbb{R}, \quad \hat{f}(\mathbf{Y}, t) = f(\mathcal{A}_t^{-1}(\mathbf{Y}), t). \tag{4.3}$$

We introduce the following notation to designate the ALE time derivative, that is the time derivative in the ALE framework

$$\left.\frac{\partial f}{\partial t}\right|_{\mathbf{Y}} : \Omega_t \times I \longrightarrow \mathbb{R}, \qquad \left.\frac{\partial f}{\partial t}\right|_{\mathbf{Y}}(\mathbf{x}, t) = \frac{\partial \hat{f}}{\partial t}(\mathcal{A}_t^{-1}(\mathbf{x}), t)$$

A very important quantity in the context of the Arbitrary Lagrangian Eulerian formulation is the velocity at which the domain Ω_t deforms, called *domain's velocity of deformation*. We denote it by $\mathbf{w}$ and it is defined as

$$\mathbf{w}(\mathbf{x}, t) = \left.\frac{\partial \mathbf{x}}{\partial t}\right|_{\mathbf{Y}}. \tag{4.4}$$

It can be shown by applying the chain rule to (4.3) that the following relationship holds between ALE and Eulerian time derivatives

$$\left.\frac{\partial f}{\partial t}\right|_{\mathbf{Y}} = \left.\frac{\partial f}{\partial t}\right|_{\mathbf{x}} + \mathbf{w} \cdot \nabla_{\mathbf{x}} f. \tag{4.5}$$

4.2 ALE formulation of the incompressible Navier-Stokes equations

In the Eulerian framework, the unsteady Navier-Stokes equations read as

$$\begin{aligned}
\left.\frac{\partial \mathbf{u}}{\partial t}\right|_{\mathbf{x}} - \mathbf{div}_{\mathbf{x}}(2\nu \mathbf{D}_{\mathbf{x}}(\mathbf{u})) + (\mathbf{u} \cdot \boldsymbol{\nabla}_{\mathbf{x}})\mathbf{u} + \nabla_{\mathbf{x}} p &= \mathbf{f}, \quad \text{in } \Omega_t \times I && (4.6)\\
\mathrm{div}_{\mathbf{x}}(\mathbf{u}) &= 0, \quad \text{in } \Omega_t \times I && (4.7)\\
\mathbf{u} &= \mathbf{0}, \quad \text{on } \Gamma_t^D \times I && (4.8)\\
(-p\mathbf{I} + 2\nu \mathbf{D}_{\mathbf{x}}(\mathbf{u}))\,\mathbf{n} &= \mathbf{0}, \quad \text{on } \Gamma_t^N \times I && (4.9)\\
\mathbf{u} &= \mathbf{u}_0, \quad \text{in } \Omega_{t_0} && (4.10)
\end{aligned}$$

where all differential operators are defined w.r.t. the Eulerian coordinate system. Here, we introduce the *strain tensor* $\mathbf{D}_{\mathbf{x}}(\mathbf{u})$ that is defined as

$$\mathbf{D}_{\mathbf{x}}(\mathbf{u}) = \frac{1}{2}\left(\boldsymbol{\nabla}_{\mathbf{x}}\mathbf{u} + (\boldsymbol{\nabla}_{\mathbf{x}}\mathbf{u})^T\right).$$

For simplicity of the exposition, we will consider homogeneous Dirichlet and Neumann boundary conditions in all $\partial\Omega_t = \Gamma_t^D \cup \Gamma_t^N$ as we did in section 3.2. The more general case can be dealt as in the previous chapter: the non homogeneous Dirichlet conditions are treated algebraically and the Neumann boundary condition just implies adding a term to the variational formulation.

Remark 4.2.1. *A constant viscosity ν implies that from the differential point of view, equation* (4.6) *is the same as* (3.24)*.*

The equivalent formulation of system of equations (4.6)-(4.7) in the ALE framework just presented is

$$\begin{aligned} \left.\frac{\partial \mathbf{u}}{\partial t}\right|_{\mathbf{Y}} - \mathbf{div}_{\mathbf{x}}(2\nu \mathbf{D}_{\mathbf{x}}(\mathbf{u})) + ((\mathbf{u}-\mathbf{w})\cdot\boldsymbol{\nabla}_{\mathbf{x}})\mathbf{u} + \nabla_{\mathbf{x}} p &= \mathbf{f}, \quad \text{in } \Omega_t \times I \qquad (4.11)\\ \mathrm{div}_{\mathbf{x}}(\mathbf{u}) &= 0, \quad \text{in } \Omega_t \times I \qquad (4.12)\end{aligned}$$

By adding the ALE time derivative, we introduce an additional term to the convection involving the domain's velocity, see (4.5).

The derivation of a weak formulation for system (4.11)-(4.12) is typically done recurring to special function spaces for trial and test functions built with the ALE map $\mathcal{A}_t$ and spaces defined in the reference domain. Using the ALE map, functions from the spaces $\mathbf{H}^1_{\Gamma^D}(\Omega_{t_0})$ and $L^2(\Omega_{t_0})$ are extended to the domain Ω_t. Let $\mathbf{V}(\Omega_t)$ and $Q(\Omega_t)$ be defined as

$$\mathbf{V}(\Omega_t) = \left\{\mathbf{v} : \Omega_t \times I \longrightarrow \mathbb{R}^d, \quad \mathbf{v} = \hat{\mathbf{v}} \circ \mathcal{A}_t^{-1}, \quad \hat{\mathbf{v}} \in \mathbf{H}^1_{\Gamma^D}(\Omega_{t_0})\right\} \qquad (4.13)$$

and

$$Q(\Omega_t) = \left\{q : \Omega_t \times I \longrightarrow \mathbb{R}, \quad q = \hat{q} \circ \mathcal{A}_t^{-1}, \quad \hat{q} \in L^2(\Omega_{t_0})\right\}. \qquad (4.14)$$

Remark 4.2.2. *We must ensure that $\mathbf{V}(\Omega_t) \subset \mathbf{H}^1(\Omega_t)$ and $Q(\Omega_t) \subset L^2(\Omega_t)$, for all $t \in I$. Such inclusions impose restrictions on the regularity of the ALE map $\mathcal{A}_t$. It was shown in Nobile [57] that if Ω_{t_0} and $\Omega_t = \mathcal{A}_t(\Omega_{t_0})$ are bounded domains, with Lipschitz continuous boundaries, and*

$$\mathcal{A}_t \in \mathbf{W}^{1,\infty}(\Omega_{t_0}), \qquad \mathcal{A}_t^{-1} \in \mathbf{W}^{1,\infty}(\Omega_t)$$

then, $v \in \mathbf{H}^1(\Omega_t)$ if and only if $\hat{v} = v \circ \mathcal{A}_t \in \mathbf{H}^1(\Omega_{t_0})$. Moreover, $\|v\|_{\mathbf{H}^1(\Omega_t)}$ is equivalent to $\|\hat{v}\|_{\mathbf{H}^1(\Omega_{t_0})}$, for all $v \in \mathbf{H}^1(\Omega_t)$. This result allows to consider (4.13) *and* (4.14) *as admissible spaces for the weak formulation of* (4.11)-(4.12)*.*

Then, the weak formulation of the Navier-Stokes equations in the ALE framework reads as

Problem 4.2.1. *For almost every $t \in I$, find $\mathbf{u}(t) \in \mathbf{V}(\Omega_t)$, with $\mathbf{u}(t_0) = \mathbf{u}_0$ in Ω_{t_0} and $p(t) \in Q(\Omega_t)$, such that*

$$\begin{aligned} &\rho \int_{\Omega_t} \left.\frac{\partial \mathbf{u}}{\partial t}\right|_{\mathbf{Y}} \cdot \mathbf{v}\, dx + \rho \int_{\Omega_t} [(\mathbf{u}-\mathbf{w})\cdot\boldsymbol{\nabla}_{\mathbf{x}}]\,\mathbf{u}\cdot\mathbf{v}\, dx \\ &+2\nu \int_{\Omega_t} \mathbf{D}_{\mathbf{x}}(\mathbf{u}) : \boldsymbol{\nabla}_{\mathbf{x}}\mathbf{v}\, dx + \int_{\Omega_t} \mathrm{div}_{\mathbf{x}}(\mathbf{v})\, p\, dx = \int_{\Omega_t} \mathbf{f}\cdot\mathbf{v}\, dx, \qquad \forall \mathbf{v} \in \mathbf{V}(\Omega_t) \qquad (4.15)\\ &\int_{\Omega_t} \mathrm{div}_{\mathbf{x}}(\mathbf{u})\, q\, dx = 0, \qquad \forall q \in Q(\Omega_t)\end{aligned}$$

The previous problem is said to be in the *non-conservative form* due to the fact that the ALE time derivative is under the integral over Ω_t. Examples of the treatment of the conservative form of the Navier-Stokes equations in this formulation can be found in Nobile [57] and Deparis [18].

Remark 4.2.3. *In the case of a pure Dirichlet boundary conditions problem, that is, when $\Gamma_t^N = \emptyset$, then Remark 3.2.1 is still valid and similar adaptations have to be done for the function spaces introduced.*

4.3 Discretization strategy

We address now the discretization of the system of equations (4.15). When dealing with the Navier-Stokes equations expressed in the ALE framework, besides the space and time discretizations that one needs to approximate the fluid variables, we also have to define the ALE map.

In the literature, the construction of the ALE map using low and high degree finite elements has been addressed. In the works of Ho and Rønquist [45], Veneziani [88], Nobile [57] and more recently, Sy and Murea [83], this map is defined through an elliptic operator. In the first reference, the authors consider spectral elements while the others considered the simple finite elements. More recently, Bouffanais [4] solved a steady Stokes problem to calculate the domain's velocity. Integrating (4.4), the ALE map is obtained. If the Stokes problem is solved with spectral elements, the ALE map constructed in the end of the process can describe a domain with curved boundary using high degree polynomials. This section is dedicated to explain our approach to construct an ALE map with high order geometrical elements and present the space-time discretization strategy adopted.

We start by introducing some notations. Let $\mathcal{T}_{t_0,\delta}$ be a triangulation of the reference domain Ω_{t_0}, where $\delta = (h, N_{\text{geo}})$. Here, h denotes the maximum diameter of all the elements in the partition and N_{geo} is the polynomial degree of the geometrical mapping associated with each element in the partition. Often $\mathcal{T}_{t_0,\delta}$ and Ω_{t_0} do not coincide and the triangulation only approximates the domain. Let $\Omega_{t_0,\delta}$ be a domain, obtained by the reunion of all elements in the triangulation $\mathcal{T}_{t_0,\delta}$, that approximates Ω_{t_0}.

4.3.1 Construction of the discrete ALE map

Let $t \in I$. We denote by $\Omega_{t,\delta}$ the domain where the Navier-Stokes equations are to be solved, called *computational domain*, and

$$\mathbf{g}_{t,\delta} : \partial\Omega_{t_0,\delta} \longrightarrow \partial\Omega_{t,\delta}$$

the map that transforms the boundary of $\Omega_{t_0,\delta}$ onto the boundary of $\Omega_{t,\delta}$ (which we assume known *a priori*).

Remark 4.3.1. *In the context of the incompressible Navier-Stokes equations applied to blood flow, we have in the simplest case, a situation as in Figure 4.1. The part of the* *boundary $\partial\Omega^{\mathcal{D}}_{t,\delta}$ represents where the blood goes in or out and it does not change in time. On the other hand, $\partial\Omega^{\sigma}_{t,\delta}$ represents the arterial wall, evolving in time according to some viscoelastic model.*

Clearly, we want to construct a discrete ALE map $\mathcal{A}_{t,\delta}$ such that

$$\mathcal{A}_{t,\delta}{}_{|_{\Omega_{t_0,\delta}}} = \mathbf{g}_{t,\delta}, \qquad \mathcal{A}_{t,\delta}(\Omega_{t_0,\delta}) = \Omega_{t,\delta}, \tag{4.16}$$

that is, a map that is capable of retrieving at any time t the actual shape of the computational domain.

To build $\mathcal{A}_{t,\delta}$, and given the fact that in practice we only have a description of the boundary of $\Omega_{t,\delta}$, several approaches can be followed. One possible approach is to construct an approximation of the domain's velocity field in $\Omega_{t,\delta}$

$$\mathbf{w}_\delta(\mathbf{x},t) = \frac{\partial \mathcal{A}_{t,\delta}}{\partial t} \circ \mathcal{A}^{-1}_{t,\delta} \tag{4.17}$$

and use a time integration scheme to obtain $\mathcal{A}_{t,\delta}$ at discrete time levels. In Bouffanais [4], a steady Stokes problem having $\mathbf{w}_\delta$ as velocity field was solved to insure that the domain velocity $\mathbf{w}_\delta$ was divergence free, thus satisfying automatically the Geometric Conservation Law (GCL), see Nobile [57]. We will give more details about the GCL in section 4.3.4.

The most widely used approach is the *harmonic extension.* It has been proposed by Ho and Rønquist [45], Nobile [57] and Veneziani [88] and it is a valid option when considering small displacements.

Associated with the triangulation $\mathcal{T}_{t_0,\delta}$, we construct the space $\mathcal{F}_{N_{\text{geo}}}(\mathcal{T}_{t_0,\delta})$, as described in section 2.1. We recall briefly its definition

$$\mathcal{F}_{N_{\text{geo}}}(\mathcal{T}_{h,N_{\text{geo}}}) = \left\{ v \in C^0(\overline{\Omega}) : \; v_{|\Omega_e} \in \mathbb{P}_{N_{\text{geo}}}(\Omega_e), \; \forall \Omega_e \in \mathcal{T}_{t_0,\delta} \right\}.$$

For each $t \in I$, the harmonic extension of $\mathbf{g}_{t,\delta}$ to the whole domain $\Omega_{t_0,\delta}$ is obtained by solving the following problem:

Problem 4.3.1. *Find $\mathcal{A}_{t,\delta} \in \left(\mathcal{F}_{N_{geo}}(\mathcal{T}_{t_0,\delta})\right)^d$ such that*

$$\begin{cases} \displaystyle\int_{\Omega_{t_0,\delta}} \nabla \mathcal{A}_{t,\delta} : \nabla \mathbf{z} \; dx = 0, & \forall \mathbf{z} \in \left(\mathcal{F}_{N_{geo}}(\mathcal{T}_{t_0,\delta}) \cap H^1_0(\Omega_{t_0,\delta})\right)^d \\ \mathcal{A}_{t,\delta} = \mathbf{g}_{t,\delta}, & \text{on } \partial\Omega_{t_0,\delta} \end{cases} \tag{4.18}$$

This is the simplest situation for an elliptic extension operator and it can be solved in the same way as the problem in section 2.4 or by using a Krylov subspace solver.

The map $\mathcal{A}_{t,\delta}$ induces a triangulation in $\Omega_{t,\delta}$, that we denote by $\mathcal{T}_{t,\delta}$, obtained by calculating the image of each element of $\mathcal{T}_{t_0,\delta}$. Notice that if a nodal basis is used to represent $\mathcal{A}_{t,\delta}$ then the degrees of freedom associated with this map, i.e. the coefficients of the basis chosen for $\left(\mathcal{F}_{N_{\text{geo}}}(\mathcal{T}_{t_0,\delta})\right)^d$, will be the coordinates of the (possibly high order) nodes of $\mathcal{T}_{t,\delta}$.

Remark 4.3.2. *Ho and Rønquist [45] used the harmonic extension just as described in Problem 4.3.1, thus generating a triangulation in the computational domain where all elements are possibly curved. On the other side, Nobile [57] only considered the case $N_{geo} = 1$ in his simulations and therefore, the triangulations used have all straight edges.*

A practical construction of a high order ALE map

Let us start by dividing the boundary $\partial\Omega_{t,\delta}$ into two parts: $\partial\Omega^{\mathcal{D}}_{t,\delta}$ a part of the boundary that we wish to remain fixed in time and $\partial\Omega^{\sigma}_{t,\delta}$ the part the changes with t. Clearly, $\partial\Omega_{t,\delta} = \partial\Omega^{\mathcal{D}}_{t,\delta} \cup \partial\Omega^{\sigma}_{t,\delta}$.

Let us assume that we have a description of $\partial\Omega^{\sigma}_{t,\delta}$ in terms of polynomials of degree N. To more clearly explain the methodology, we consider the domains of Figure 4.1.

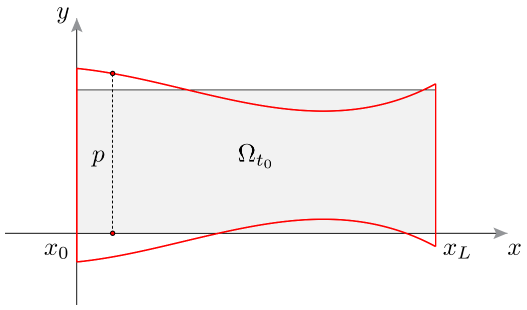


Figure 4.2: Description of the reference and computational domains. The top and bottom boundaries are described in terms of polynomials.

We do the following assumptions: (i) the upper and lower parts of the boundary $\partial\Omega^{\sigma}_{t,\delta}$ are described by polynomials p of degree N defined in $[x_0, x_L]$; (ii) $\Omega_{t_0,\delta}$ can be covered exactly by a triangulation composed of elements with straight edges. This assumption is not as restrictive as it seems. Actually, given a geometry, mesh generators can typically create triangulations for $N_{\text{geo}} = 1, 2$. Some exist, like GMSH, that can discretize a domain with geometrical elements of order up to 5, but it might happen that elements in the boundary are invalid (we will explain this situation later on).

Remark 4.3.3. *In the case of a three dimensional geometry, then one would need a parametrization of the surface of the domain, which could be done using at least two polynomials.*

Given the description of the boundary in terms of polynomials, we perform a standard harmonic extension of the function $\mathbf{g}_{t,\delta}$ using $\mathcal{T}_{t_0,\delta}$ and $N_{\text{geo}} = 1$ in the function space of Problem 4.3.1, that is, we use $\mathbb{P}_1$ standard finite elements to solve the harmonic extension problem. Like this, we obtain an ALE map $\mathcal{A}_{t,\delta_1}$, where $\delta_1 = (h, 1)$, that once applied to the triangulation $\mathcal{T}_{t_0,\delta}$ generates a mesh $\mathcal{T}_{t,\delta_1}$ for $\Omega_{t,\delta}$. This process is depicted in Figure 4.3. We highlight that at this point, the triangulation $\mathcal{T}_{t,\delta_1}$ has only straight edges by construction and the map $\mathcal{A}_{t,\delta_1}$ is piecewise linear in $\Omega_{t_0,\delta}$. Our goal is to construct a map

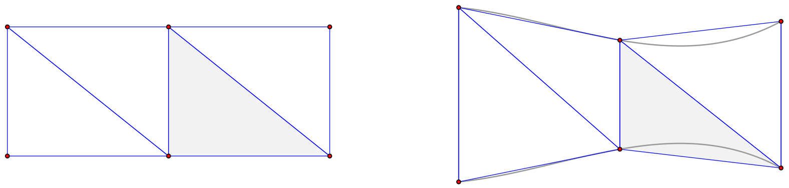


Figure 4.3: Transformation from the mesh in the reference domain, $\mathcal{T}_{t_0,\delta}$, to the mesh in the computational domain, $\mathcal{T}_{t,\delta_1}$ through the map $\mathcal{A}_{t,\delta_1}$.

that verifies (4.16). For the time being, $\mathcal{A}_{t,\delta_1}$ might not satisfy this property. We see from Figure 4.3 that if the boundary of the domain are curved, the mesh that is generated (that has straight edges) might not cover the computational domain. In order to construct a map that conforms with the curved boundary, we need more degrees of freedom to describe the transformation.

The next step is to project $\mathcal{A}_{t,\delta_1}$ onto the space $\left(\mathcal{F}_{N_{\text{geo}}}(\mathcal{T}_{t_0,\delta})\right)^d$. Let $\mathcal{B} = \{\boldsymbol{\phi}_i\}_i$ be a nodal basis for this space (in our simulations, $\mathcal{B}$ is the Lagrange basis associated with Fekete points). With these notations, the projection of $\mathcal{A}_{t,\delta_1}$ onto $\left(\mathcal{F}_{N_{\text{geo}}}(\mathcal{T}_{t_0,\delta})\right)^d$, that we denote by $\mathcal{A}^*_{t,\delta}$, is

$$\mathcal{A}^*_{t,\delta} = \sum_i \alpha_i \boldsymbol{\phi}_i$$

where the coefficients α_i are determined by interpolating $\mathcal{A}_{t,\delta_1}$ over the high order nodes associated with $\left(\mathcal{F}_{N_{\text{geo}}}(\mathcal{T}_{t_0,\delta})\right)^d$. Since the basis is nodal, these coefficients are no more than the evaluation of $\mathcal{A}_{t,\delta_1}$ at these nodes.

From the functional point of view, $\mathcal{A}_{t,\delta_1}$ and $\mathcal{A}^*_{t,\delta}$ are the same polynomial. However, the number of degrees of freedom are substantially different. These extra degrees of freedom are the key to construct the high order ALE map.

Let us look now more closely to the effect of $\mathcal{A}^*_{t,\delta}$ on a point set defined in the reference domain. We see from Figure 4.4 that the edges of the triangulation $\mathcal{T}_{t,\delta_1}$ in contact with the curved wall do not conform with it. In order for that to happen, we can change the value of the degrees of freedom, associated with that edge, of $\mathcal{A}^*_{t,\delta}$. Since the basis in which $\mathcal{A}^*_{t,\delta}$ is expressed is the Lagrange nodal basis, this can be done by shifting the value of the degrees of freedom. If $\mathbf{x}_0$ is a point in the mentioned edge that corresponds to a degree of freedom and (x_t, y_t) are the coordinates of it's image through $\mathcal{A}^*_{t,\delta}$ then we take

$$\mathcal{A}^*_{t,\delta}(\mathbf{x}_0) = (x_t, p(x_t)).$$

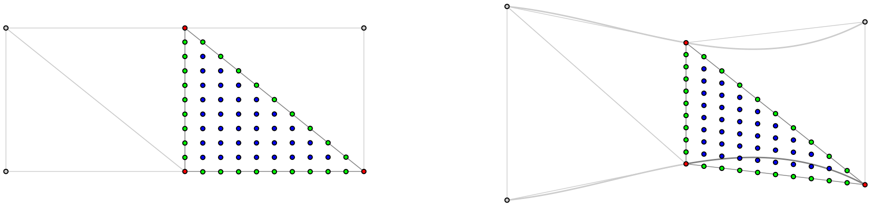


Figure 4.4: Effect of $\mathcal{A}^*_{t,\delta}$ on a equidistributed point set defined in an element of the reference mesh.

This shifting in the coordinates solves the problem of making the edges of the elements conform with the curved boundary. However, this might create a map that has a singular Jacobian since part of the interior of the element in the reference domain is mapped outside the corresponding element in the computational mesh, as seen in Figure 4.5.

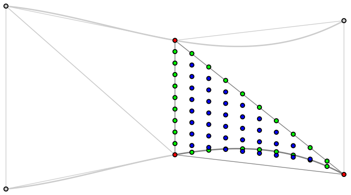


Figure 4.5: Effect of $\mathcal{A}^*_{t,\delta}$ if only the degrees of freedom in the edges are shifted.

The final step to obtain a valid ALE map is to shift also the nodes in the faces of the elements of the reference domain and obtain a situation as in Figure 4.6. The coordinates of the new nodes are obtained using a transformation of the type presented in section 1.4.

Remark 4.3.4. *We highlight that at the time this thesis is being written, GMSH does not shift the points in the interior of the elements as we do, it only does so to the points in the edges. If the deformation of the elements is "big", then it can happen that the curved elements generated are simply invalid, due to the fact that the image by the geometrical transformation is not contained inside the three edges of the triangle, just like in Figure 4.5.*

To minimize the evaluation of the functions describing the boundary, we perform a loop in all elements that intersect the boundary and have a curved edge. Then, for each element, we use the a transformation as the ones presented in section 1.4 to generate the new values at the degrees of freedom and replace these values in $\mathcal{A}^*_{t,\delta}$. Let $\mathcal{A}_{t,\delta}$ denote the updated map.

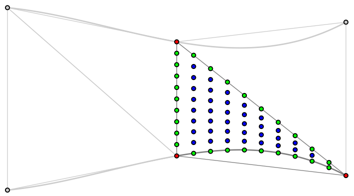


Figure 4.6: Transformation from the reference mesh to the computational one (update on face's degrees of freedom).

After this update procedure, $\mathcal{A}_{t,\delta}$, transforms the straight boundary edges of the reference mesh into the curved edges that conform with the curved boundary, as we see in Figure 4.6. This is precisely the kind of discrete ALE map that we consider from now on.

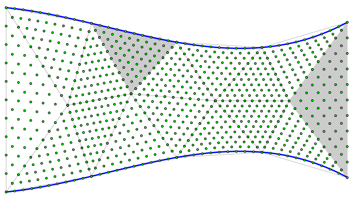


Figure 4.7: The effect of the mapping $\mathcal{A}_{t,\delta}$ to an equidistributed point set in the reference domain. The shadowed elements are, from left to right, triangles where the geometrical transformation is of high degree or linear, respectively.

Remark 4.3.5. *We highlight that the construction just presented does not depend on the extension operator that was used to generate the first mesh in the computational domain. Therefore, it can be coupled with more complicated procedures to create such mesh.*

Advantages and disadvantages. A consequence of the definition of the map just presented is that it is affine for elements that do not share an edge with the curved boundary. For these elements T,

$$\mathcal{A}_{t,\delta_{|T}} = \mathcal{A}_{t,\delta_1}{}_{|T}.$$

This also means that the geometric mapping associated with these elements is affine. This situation is illustrated in Figure 4.7. The advantage of this property is that when integrating linear/bilinear forms in these elements, a constant Jacobian is associated with the geometrical transformation and therefore a minimal order quadrature can be used (in the sense that the geometrical transformation does not need to be taken into account), see section 1.5. For the elements that intersect the curved boundary, the quadrature order has to account for the possibly non constant Jacobian. However, with this approach, we can reduce the integration cost in the whole mesh and confine it to the elements in contact with the curved boundary.

A final remark concerns possible strategies in the case the boundary's deformation is "big". In this case, it can happen that an interior straight edge of the mesh intersects the curved boundary, see Figure 4.8. This originates an invalid element and several techniques are proposed in literature to deal with this issue, see Sherwin and Karniadakis [49]. The map we propose here is built in such a way that increasing the order of the map will not solve the problem. This is a matter regarding the configuration of the straight edge mesh. If we increase the polynomial degree to represent the geometrical transformation, at best, we describe the curved boundary better, but the problem of the edge crossing the inner edge of the mesh still remains. To go around this issue, one can perform edge swapping or

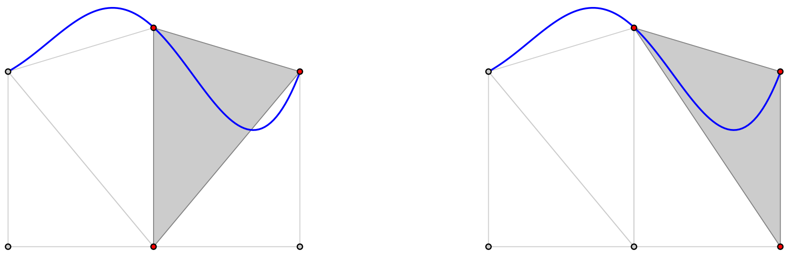


Figure 4.8: Invalid element created due to the high distortion in the boundary (left). Edge swap technique to correct possible invalid elements (right).

similar techniques, see Figure 4.8. In this figure, the edge that crosses the curved boundary is moved to allow the creation of a valid element. However, since we do not want to change the structure of our reference mesh, other possibilities are to use a control function in the Laplace operator of the harmonic extension to somehow follow the boundary movement or use a reference mesh that is refined near the curved boundary. A final, more costly possibility, is to re-mesh the whole domain. In our approach, since the displacements we consider are small, we do not control the quality of the mesh.

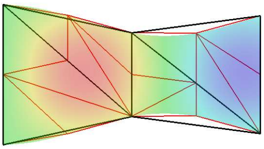
Some numerical results

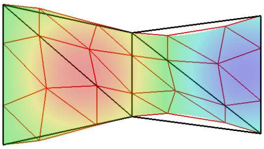
We present now some numerical results to assess the accuracy of the ALE map describing the surface of a domain. Let us consider the reference domain $\Omega_{t_0} = (0,5) \times (-1,1)$.

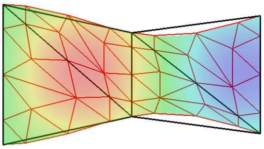
We define Ω as the domain we obtain by moving the upper and lower sides of the rectangle Ω_{t_0} using the following displacement functions:

- upper boundary: $\boldsymbol{\eta}(x,1) = [x, 1 + 0.3\cos(x)]^T$

- lower boundary: $\boldsymbol{\eta}(x,-1) = [x, -1.1 - 0.3\cos(x)]^T$


(a) $\mathbb{P}_2$ ALE map


(b) $\mathbb{P}_3$ ALE map


(c) $\mathbb{P}_4$ ALE map

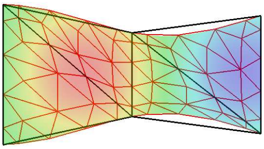

(d) $\mathbb{P}_5$ ALE map

Figure 4.9: In black the $\mathbb{P}_1$ mesh used in the construction of the ALE maps. In red, the $\mathbb{P}_1$ mesh constructed on top of the high order nodes.

In Figures 4.9(a)-4.9(d) we show the application of the ALE maps $\mathcal{A} : \Omega_{t_0} \longrightarrow \Omega$ constructed using polynomials of degree two to five to a mesh of the reference domain.

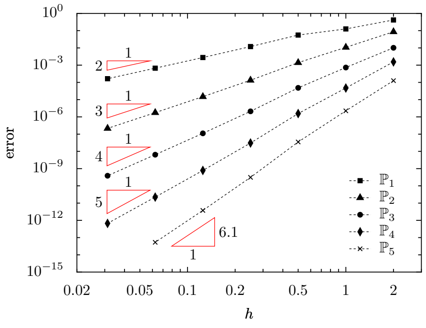


Figure 4.10: Convergence plot for the high order ALE maps

We also wanted to determine the accuracy at which the ALE maps describe the boundary of the domain Ω. For this, we measured the error

$$\|(\mathcal{A}(\cdot,1) - \boldsymbol{\eta}(\cdot,1)) \cdot \mathbf{e}_2\|_{L^2(0,5)}$$

in the upper boundary of the reference domain and

$$\|(\mathcal{A}(\cdot,-1)-\boldsymbol{\eta}(\cdot,-1))\cdot\mathbf{e}_2\|_{L^2(0,5)}$$

in the lower part of Ω_{t_0}. We plot the sum of both quantities in Figure 4.10. The error decreases with the expected rates, ie, $\mathcal{O}(h^{N_{\text{geo}}+1})$

4.3.2 Space discretization

We address now the discretization in space of the system of equations (4.15). Let

$$\mathbf{V}_\delta(\Omega_{t,\delta}) = \left\{\mathbf{v}:\Omega_{t,\delta}\times I \longrightarrow \mathbb{R}^d,\ \ \mathbf{v}=\hat{\mathbf{v}}\circ\mathcal{A}_{t,\delta}^{-1},\ \ \hat{\mathbf{v}}\in\mathbf{H}^1_{\Gamma^D}(\Omega_{t_0})\cap(\mathcal{F}_N(\mathcal{T}_{t_0,\delta}))^d\right\} \tag{4.19}$$

and

$$Q_\delta(\Omega_{t,\delta}) = \left\{q:\Omega_{t,\delta}\times I\longrightarrow\mathbb{R},\ \ q=\hat{q}\circ\mathcal{A}_{t,\delta}^{-1},\ \ \hat{q}\in\mathcal{F}_M(\mathcal{T}_{t_0,\delta})\right\} \tag{4.20}$$

where $M=N-1$ or $M=N-2$. These are going to be the finite dimensional functional spaces in which velocity and pressure will be discretized in space.

Remark 4.3.6. *Once again, we apply a transformation to an element (not the reference element now, but an element in the reference mesh) to obtain an element in the computational mesh. Taking into account Remark 1.4.1, it is clear that this has consequences of the approximation properties of the spaces* (4.19) *and* (4.20)*. However, the consideration in Remark 1.4.1 are also valid here. Given the reference mesh, we apply the discrete ALE map to construct the mesh for the computational domain. This implies that in the new mesh, we are in the conditions of Remark 1.4.1 and therefore, if we want to represent exactly degree N polynomials in $\Omega_{t,\delta}$, we need to consider, in our construction, polynomials of degree N in $\Omega_{t_0,\delta}$ and polynomials of degree $N\cdot N_{geo}$ in the reference element. Under these hypotheses and the regularity assumptions of Remark 4.2.2*

$$\mathbf{V}_\delta(\Omega_{t,\delta}) = (\mathcal{F}_N(\mathcal{T}_{t,\delta}))^d\cap\mathbf{H}^1_{\Gamma^D}(\Omega_{t,\delta})$$

and

$$Q_\delta(\Omega_{t,\delta}) = \mathcal{F}_M(\mathcal{T}_{t,\delta}).$$

With these functional spaces, the semi-discrete variational problem reads as

Problem 4.3.2. *For almost every $t\in I$, find $\mathbf{u}_\delta(t)\in(\mathcal{F}_N(\mathcal{T}_{t,\delta}))^d$, with $\mathbf{u}_\delta(t_0)=\mathbf{u}_{0,\delta}$ in $\Omega_{t_0,\delta}$ and $p_\delta(t)\in Q(\Omega_{t,\delta})$, such that*

$$\begin{aligned}
&\rho\int_{\Omega_t}\left.\frac{\partial\mathbf{u}_\delta}{\partial t}\right|_{\mathbf{Y}}\cdot\mathbf{v}\,dx+\rho\int_{\Omega_t}[(\mathbf{u}_\delta-\mathbf{w}_\delta)\cdot\boldsymbol{\nabla}_{\mathbf{x}}]\,\mathbf{u}_\delta\cdot\mathbf{v}\,dx\\
&+\frac{\rho}{2}\int_{\Omega_t}\mathrm{div}_{\mathbf{x}}(\mathbf{u}_\delta)\mathbf{u}_\delta\cdot\mathbf{v}\,dx+2\nu\int_{\Omega_t}\mathbf{D}_{\mathbf{x}}(\mathbf{u}_\delta):\boldsymbol{\nabla}_{\mathbf{x}}\mathbf{v}\,dx\\
&+\int_{\Omega_t}\mathrm{div}_{\mathbf{x}}(\mathbf{v})\,p_\delta\,dx=\int_{\Omega_t}\mathbf{f}\cdot\mathbf{v}\,dx, && \forall\mathbf{v}\in\mathbf{V}_\delta(\Omega_{t,\delta})\\
&\int_{\Omega_t}\mathrm{div}_{\mathbf{x}}(\mathbf{u}_\delta)\,q\,dx=0, && \forall q\in Q_\delta(\Omega_{t,\delta})
\end{aligned} \tag{4.21}$$

The *discrete domain's velocity* $\mathbf{w}_\delta$ is defined as in (4.17). Since the construction of the discrete ALE map lies upon the discretization of a differential problem in a mesh, we also call this quantity *mesh velocity.*

The reader will notice that this formulation has an extra term

$$\frac{\rho}{2}\int_{\Omega_t} \operatorname{div}_{\mathbf{x}}(\mathbf{u}_\delta)\mathbf{u}_\delta \cdot \mathbf{v}\; dx.$$

This term is added to enhance the stability of the spatial discretization and it is consistent with the Navier-Stokes equations, since at the fully continuous level, $\operatorname{div}_{\mathbf{x}}(\mathbf{u}) = 0$. For a more detailed explanation of the importance of adding such term to the formulation in the context of stability, see Nobile [57].

4.3.3 Time integration

The temporal discretization described in section 3.2.1 for the incompressible Navier--Stokes equations in a fixed domain can be extended to (4.21) in a straightforward way. The BDFq schemes applied to (4.21) will read

Problem 4.3.3. *For each $n \geqslant q-1$, let $t_n = t_0 + n\Delta t$ and $N_T = \left[\frac{T-t_0}{\Delta t}\right]$. Then, we look for the solution $(\mathbf{u}_\delta^{n+1}, p_\delta^{n+1}) \in \left(\mathcal{F}_N(\mathcal{T}_{t_{n+1},\delta})\right)^d \times Q_\delta(\Omega_{t_{n+1},\delta})$, with $\mathbf{u}_\delta^0 = \mathbf{u}_{0,\delta}$ in $\Omega_{t_0,\delta}$, such that*

$$\begin{aligned}
&\rho\int_{\Omega_{t_{n+1}}} \frac{\beta_{-1}}{\Delta t}\mathbf{u}_\delta^{n+1}\cdot \mathbf{v}\; dx + \rho\int_{\Omega_{t_{n+1}}} \left[\left(\mathbf{u}_\delta^{*} - \mathbf{w}_\delta^{n+1}\right)\cdot \boldsymbol{\nabla}_{\mathbf{x}}\right]\mathbf{u}_\delta^{n+1}\cdot \mathbf{v}\; dx \\
&+\frac{\rho}{2}\int_{\Omega_{t_{n+1}}} \operatorname{div}_{\mathbf{x}}(\mathbf{u}_\delta^{*})\mathbf{u}_\delta^{n+1}\cdot \mathbf{v}\; dx + 2\nu\int_{\Omega_{t_{n+1}}} \mathbf{D}_{\mathbf{x}}(\mathbf{u}_\delta^{n+1}) : \boldsymbol{\nabla}_{\mathbf{x}}\mathbf{v}\; dx + \\
&\int_{\Omega_{t_{n+1}}} \operatorname{div}_{\mathbf{x}}(\mathbf{v})\; p_\delta^{n+1}\; dx = \int_{\Omega_{t_{n+1}}} \tilde{\mathbf{f}}_\delta^{n+1}\cdot \mathbf{v}\; dx, \qquad \forall \mathbf{v}\in \mathbf{V}_\delta(\Omega_{t_{n+1},\delta}) \\
&\int_{\Omega_{t_{n+1}}} \operatorname{div}_{\mathbf{x}}(\mathbf{u}_\delta^{n+1})\; q\; dx = 0, \qquad \forall q\in Q_\delta(\Omega_{t_{n+1},\delta})
\end{aligned} \tag{4.22}$$

where

$$\tilde{\mathbf{f}}_\delta^{n+1} = \mathbf{f}^{n+1} + \sum_{j=0}^{q-1}\frac{\beta_j}{\Delta t}\mathbf{u}_\delta^{n-j}$$

Notice that the functions $\mathbf{u}_\delta^{n-j}$ should be defined in $\Omega_{t_{n-j},\delta}$, which might not coincide with the integration domain $\Omega_{t_{n+1},\delta}$. However, these quantities can be ported from their domain of definition to the current one by applying ALE maps. If we denote by $\mathbf{u}_\delta^{n-j,*}$ the approximation of $\mathbf{u}(t_{n-j})$ defined in $\Omega_{t_{n-j},\delta}$, then

$$\mathbf{u}_\delta^{n-j} = \mathbf{u}_\delta^{n-j,*}\circ \mathcal{A}_{t_{n+1},\delta}\circ \mathcal{A}_{t_{n-j},\delta}^{-1}.$$

Similar considerations are valid every time a quantity defined in a domain of the type $\Omega_{t_\cdot,\delta}$ needs to be ported to the current computational domain $\Omega_{t_{n+1},\delta}$.

In equation (4.22), there are two quantities that we have not yet defined, or at least said how to calculate: $\mathbf{u}^*_\delta$ and $\mathbf{w}^{n+1}_\delta$. Regarding the former, this is a linearization of the convective term of the Navier-Stokes equations. We extrapolate the velocity in the same way as in section 3.2.2 and define $\mathbf{u}^*_\delta$ exactly as (3.34), for each $q = 1, 2, 3, 4$.

Regarding the second quantity, $\mathbf{w}^{n+1}_\delta$, since it is defined as the time derivative of the ALE map, we also adopt the BDFq schemes to approximate it. For instance, for $q = 2$, we have

$$\mathbf{w}^{n+1}_\delta = \frac{1}{\Delta t}\left(\frac{3}{2}\mathcal{A}_{t_{n+1},\delta} - 2\mathcal{A}_{t_n,\delta} + \frac{1}{2}\mathcal{A}_{t_{n-1},\delta}\right) \circ \mathcal{A}^{-1}_{t_{n+1},\delta}. \tag{4.23}$$

Remark 4.3.7. *Also in system* (4.22)*, we can add an IP stabilization term as in section 3.2.2, in a straightforward fashion. We highlight that since our ALE map transforms straight interior edges from the reference triangulation into straight interior edges in the computational one, the computational cost of this approach is not increased by the fact that we have high order geometrical elements in the boundary.*

Numerical schemes of the type (4.22) have been analyzed in literature in the context of a linear advection diffusion problem. It has been shown in Nobile [57] that when applying the Backward Euler time integration method (equivalent to our method with $q = 1$) to the advection diffusion problem in the non-conservative form, the scheme is only conditionally stable. The stability condition (derived in [57]) is

$$\Delta t < \left(\|\mathrm{div}(\mathbf{w}^n_\delta)\|_{L^\infty(\Omega_{t_n,\delta})} + \sup_{t\in(t_n,t_{n+1})} \left\| J_{\mathcal{A}_{t_n,t_{n+1}}} \mathrm{div}(\mathbf{w}_\delta)\right\|_{L^\infty(\Omega_{t,\delta})} \right)^{-1} \tag{4.24}$$

for all $n = 1, \dots, N_T$. We remark that the quantities involved in (4.24) are only geometrical. If the mesh velocity is calculated in such a way that it is divergence free, then the scheme is unconditionally stable. This is a sufficient condition to satisfy the Geometric Conservation Law (GCL), see section 4.3.4.

Also in Nobile [57], for the case $q = 2$, again in the context of a linear advection diffusion equation, it is shown that the method is conditionally stable and the time step restriction depends only on geometrical quantities, just like (4.24).

4.3.4 Satisfying the Geometric Conservation Law

We say that an equation/numerical scheme satisfies the *Geometric Conservation Law* (GCL) if it is able to reproduce a constant solution (in the absence of source terms and proper boundary conditions).

It is clear that the system of equations (4.11)-(4.12) trivially satisfies the GCL, with the proper choice of boundary conditions. However, at the discrete level, such property is no longer automatically satisfied. Regarding numerical schemes, it has been proved in the context of finite-volume methods that the GCL is linked with convergence properties of the proposed scheme, see Farhat, Geuzaine and Grandmont [27] and Lesoinne and Farhat [53]. In the context of the Navier-Stokes in the ALE framework, no evidence exists that makes the GCL a sufficient or necessary condition for convergence or stability, see Mavriplis and Yang [54].

In Nobile [57], two strategies for satisfying the GCL are proposed. The first introduces appropriate time integration schemes in the formulation, leading to the appearance of intermediate integration domains. The second is to have a mesh velocity that is divergence free. In this case, the GCL is automatically satisfied. This was the approach followed by Bouffanais [4].

For our schemes, we use the definition of the GCL property to verify that it is satisfied. Let us suppose that $\mathbf{u}_\delta^i \equiv \tilde{\mathbf{u}}$ and $p_\delta^i \equiv 0$ are constant, for all $i = 0, \ldots, n$. Notice that if a constant velocity is solution of the Navier-Stokes system, then the pressure is zero all over, in the presence of homogeneous Neumann boundary conditions.

Then, from the system of equations (4.22), in order that $(\tilde{\mathbf{u}}, 0)$ is a solution of (4.22), we need that

$$\rho \int_{\Omega_{t_{n+1}}} \frac{\beta_{-1}}{\Delta t} \mathbf{u}_\delta^{n+1} \cdot \mathbf{v} \; dx = \int_{\Omega_{t_{n+1}}} \sum_{j=0}^{q-1} \frac{\beta_j}{\Delta t} \mathbf{u}_\delta^{n-j} \cdot \mathbf{v} \; dx, \ \forall \mathbf{v} \in \mathbf{V}_\delta(\Omega_{t_{n+1},\delta})$$

which is true if

$$\beta_{-1} = \sum_{j=0}^{q-1} \beta_j.$$

The previous equality is a consequence of the consistency of the BDFq schemes. Therefore, our formulation of the Navier-Stokes equations in the ALE frame satisfies the GCL, for all BDFq schemes considered.

Remark 4.3.8. *Since the GCL property only depends on the momentum equation, also the Yosida-q schemes, when applied to solve equation* (4.25)*, satisfy the GCL.*

4.3.5 Linear system solution

Let us consider basis functions for the spaces $\mathbf{V}_\delta(\Omega_{t_{n+1},\delta})$ and $Q_\delta(\Omega_{t_{n+1},\delta})$, say

$$\mathbf{V}_\delta(\Omega_{t_{n+1},\delta}) = \text{span}\{\boldsymbol{\phi}_i\}_{i=0}^{N_u}, \quad Q_\delta(\Omega_{t_{n+1},\delta}) = \text{span}\{\psi_i\}_{i=0}^{N_p}.$$

In practice, we remark that the construction of $\{\boldsymbol{\phi}_i\}$ and $\{\psi_i\}$ does not have to be done for each t. By creating the functional space $\mathbf{V}_\delta(\Omega_{t_0,\delta})$, for instance, and using the ALE map to move the nodes of the subjacent mesh, one can obtain basis functions for $\mathbf{V}_\delta(\Omega_{t_{n+1},\delta})$. Actually, we only modify the geometrical transformation in each element to produce the elements of $\mathcal{T}_{t_{n+1},\delta}$.

We introduce the following matrices and vectors (we omit the superscript $n+1$ to indicate the dependence of the basis functions and the matrices on n to simplify the notation):

$$\begin{array}{rcll}
G_\delta(i,j) &=& -\displaystyle\int_{\Omega_{t_{n+1}}} \mathrm{div}_{\mathbf{x}}(\boldsymbol{\phi}_i)\ \psi_j\ dx, & 0 \leqslant i \leqslant N_u, \\
 & & & 0 \leqslant j \leqslant N_p \\
D_\delta(i,j) &=& \displaystyle\int_{\Omega_{t_{n+1}}} \mathrm{div}_{\mathbf{x}}(\boldsymbol{\phi}_j)\ \psi_i\ dx, & 0 \leqslant j \leqslant N_u, \\
 & & & 0 \leqslant i \leqslant N_p \\
H_\delta(i,j) &=& 2\nu \displaystyle\int_{\Omega_{t_{n+1}}} \mathbf{D}_{\mathbf{x}}(\boldsymbol{\phi}_i) : \boldsymbol{\nabla}_{\mathbf{x}}\boldsymbol{\phi}_j\ dx, & 0 \leqslant i,j \leqslant N_u \\
C_\delta(i,j) &=& \rho \displaystyle\int_{\Omega_{t_{n+1}}} \left[\left(\mathbf{u}_\delta^* - \mathbf{w}_\delta^{n+1}\right) \cdot \boldsymbol{\nabla}_{\mathbf{x}}\boldsymbol{\phi}_i + \frac{1}{2}\mathrm{div}_{\mathbf{x}}(\mathbf{u}_\delta^*)\boldsymbol{\phi}_i \right] \cdot \boldsymbol{\phi}_j\ dx, & 0 \leqslant i,j \leqslant N_u \\
M_\delta(i,j) &=& \displaystyle\int_{\Omega_{t_{n+1}}} \boldsymbol{\phi}_i \cdot \boldsymbol{\phi}_j\ dx, & 0 \leqslant i,j \leqslant N_u \\
\mathbf{F}_\delta(j) &=& \displaystyle\int_{\Omega_{t_{n+1}}} \tilde{\mathbf{f}}_\delta^{n+1} \cdot \boldsymbol{\phi}_j\ dx, & 0 \leqslant j \leqslant N_u
\end{array}$$

and

$$F_\delta = \rho \frac{\beta_{-1}}{\Delta t} M_\delta + \nu H_\delta + C_\delta.$$

Then Problem 4.3.3 is equivalent to solve, for each $n \geqslant 1$ a system of the form

$$\begin{bmatrix} F_\delta & G_\delta \\ D_\delta & 0 \end{bmatrix} \begin{bmatrix} \mathbf{U}_\delta^{n+1} \\ \mathbf{P}_\delta^{n+1} \end{bmatrix} = \begin{bmatrix} \mathbf{F}_\delta \\ 0 \end{bmatrix} \tag{4.25}$$

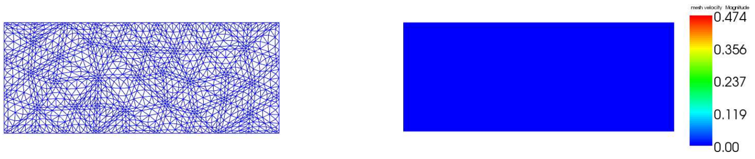
where $\mathbf{U}_\delta^{n+1}$ and $\mathbf{P}_\delta^{n+1}$ are the representations of $\mathbf{u}_\delta^{n+1}$ and p_δ^{n+1} in the bases of $\mathbf{V}_\delta(\Omega_{t_{n+1},\delta})$ and $Q_\delta(\Omega_{t_{n+1},\delta})$, respectively. Again, we remark, though we did not include such term in the definition of C_δ that the IP stabilization term might be included (now considering also the term coming from the mesh velocity in the stabilization).

Remark 4.3.9. *To solve system* (4.25) *one can use several approaches, namely, the ones described in the previous chapter. We will consider in section 4.4 the solution of* (4.25) *by using a direct method applied to the whole system and the Yosida-q schemes.*

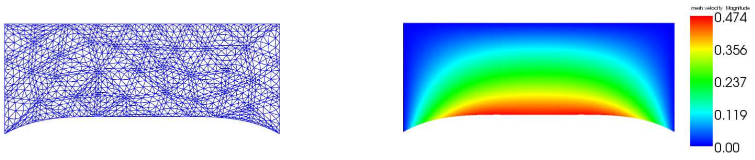
4.4 Numerical results

Consider $\Omega_{t_0} = (0,5) \times (-1,1)$ and Ω_t obtained from the reference domain by applying the following displacement law

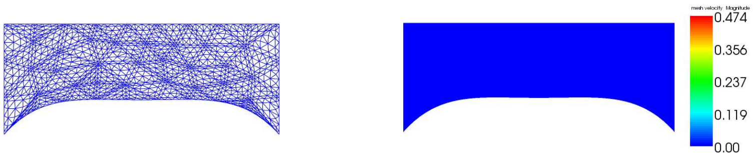
$$\mathbf{d}(\mathbf{x},t) = 0.02\left((\mathbf{x}-2.5)^2+5\right)\mathbf{x}(5-\mathbf{x})\left(f(t)\chi(t\in[1,3]) + \chi(t>3)\right), \tag{4.26}$$



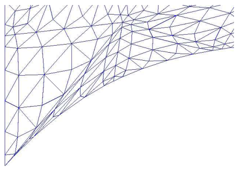
(a) $t = 1$.



(b) $t = 2$.



(c) $t = 3$.



(d) Zoom of the left lower corner of the mesh.

Figure 4.11: Plot of the high order mesh used in visualizing the simulations (left) and the magnitude of the mesh velocity (right) at different time steps ($\nu = 10^{-3}$).

to the lower edge of the rectangle, with $t \in I = [0,5]$, $\chi(\cdot)$ the characteristic function and

$$\begin{aligned} f(t) &= -0.15625t^7 + 2.1875t^6 - 12.46875t^5 + 37.1875t^4 - 62.34375t^3 \\ &\quad +59.0625t^2 - 29.53125t + 6.0625 \end{aligned}$$

Remark 4.4.1. *The function f satisfies $f(1) = 0$, $f(3) = 1$ and $f^{(k)}(1) = f^{(k)}(3) = 0, k = 1, 2, 3$. It is the only polynomial of degree 7 that satisfies these conditions. It was devised so that the variation of the mesh velocity in time is smooth enough.*

Let $\hat{\mathbf{u}} = (1 - y^2, 0)^T$ and $\hat{p} = -2\nu(x - 5)$ be the solution of the steady Navier-Stokes equations in the reference domain Ω_{t_0}.

We consider equations (4.11)-(4.12) defined in Ω_t with $\mathbf{f} \equiv 0$. Regarding boundary conditions, we define

$$\Gamma_t^N = \{5\} \times (-1,1) \qquad \text{and} \qquad \Gamma_t^D = \partial\Omega_t \setminus \Gamma_t^N.$$

We set as boundary conditions

$$\mathbf{u} = \hat{\mathbf{u}}, \quad \text{on } \Gamma_t^D \tag{4.27}$$

and

$$(-p\mathbf{I} + \mathbf{D_x}(\mathbf{u}))\,\mathbf{n} = (-\hat{p}\mathbf{I} + \mathbf{D_x}(\hat{\mathbf{u}}))\,\mathbf{n}, \quad \text{on } \Gamma_t^N$$

We remark that since the boundary of Ω_t deforms inside Ω_{t_0}, equation (4.27) makes sense in the part of the domain that changes in time. Also, the pair $(\hat{\mathbf{u}}, \hat{p})$ is the solution of (4.11)-(4.12) with the boundary conditions that we presented. In Figure 4.11 we show the evolution of the domain Ω_t for three time steps. We also show in the same figure, the macro structure of a fourth order mesh, that is, a triangulation such that $N_{\text{geo}} = 4$, that describes the geometry of the domain exactly. It can be seen that for each macro triangle, there is a mesh, built upon the high order nodes of a six degree polynomial space (using the procedure described in section 2.3).

This benchmark test allows us to test the Navier-Stokes ALE framework with spectral elements in space, high order time integration, high order geometrical elements and also the IP stabilization.

The discretization of this problem is done with the scheme proposed in Problem 4.3.3. We try two strategies to solve the linear system (4.25): the first, using a direct method and the second, using the Yosida-q schemes.

Let us first define the following error quantities that we are going to measure in order to assess the accuracy of the solver. We denote by

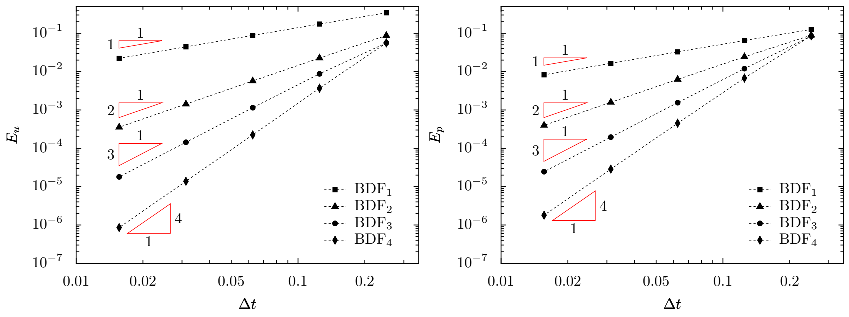
$$E_u = \left(\Delta t \sum_{n=0}^{N_T} \|\mathbf{u}(t_n) - \mathbf{u}_\delta^n\|^2_{\mathbf{H}^1(\Omega_{t_n,\delta})} \right)^{1/2}$$

and

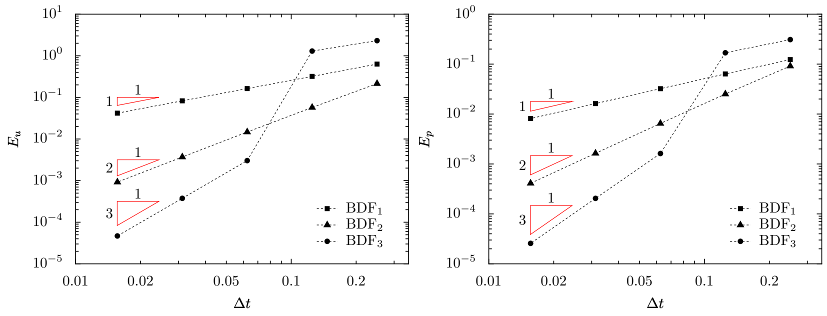
$$E_p = \left(\Delta t \sum_{n=0}^{N_T} \|\mathbf{p}(t_n) - p_\delta^n\|^2_{L^2(\Omega_{t_n,\delta})} \right)^{1/2}.$$

4.4.1 Using a direct method

In Figure 4.12 we plot the error quantities E_u and E_p for several choices of approximation spaces for velocity and pressure, different values of N_{geo} and different integration in time strategies. We considered for this test $h = 0.5$, $\nu = 10^{-3}$, $\rho = 1$. We highlight



(a) $N = 2, M = 1, N_{\text{geo}} = 1$.



(b) $N = 4, M = 2, N_{\text{geo}} = 2$.

Figure 4.12: Plot of the errors E_u and E_p for different choices of velocity-pressure spaces, geometrical elements and BDFq schemes.

that the flow is convection dominated (without the stabilization term, the method would not converge) and we have stabilized the equations by the interior penalty term. We took $\gamma = 0.1$. These results were obtained by solving directly the linear system (4.25) with a LU factorization. We highlight that the preconditioner proposed in the previous chapter could have been used, but the main goal here was to study the numerical properties of the methods in terms of accuracy. Moreover, the size of the problems solved in these tests falls in the range where the LU factorization performs better than the block preconditioner proposed in chapter 3. Moreover, we reuse the LU factorization as preconditioner until the number of iterations needed to solve the linear system is equal to 10. Once this value is attained, the LU factorization is recalculated. A better strategy to determine when to recalculate the preconditioner is described in [92].

From Figure 4.12 we confirm the expected convergence order for the proposed methods in time. Using a BDFq time integrator, a linear extrapolation of the convective term of the same order and an approximation of the mesh velocity also with a BDFq formula, the error, in Δt is of the order of Δt^q, $q = 1, 2, 3, 4$. The convergence order of each scheme is seen in Figure 4.12 through the slope of each curve.

Due to stability constraints of the time integration technique used, when we increase the polynomial degree, the stability regions of the *BDF4* and *BDF3* start not to be big enough to handle the stiffness of the problem and we only get the schemes to give acceptable results when we decrease Δt. In Figure 4.12(b), we do not even plot the results for BDF*4* because the method was not stable for the range of Δt we considered. On the other hand, the BDF*1* and BDF*2* schemes remain stable.

We remark that in Figures 4.12(a) and 4.12(b), the numerical schemes used describe the solution of the problem exactly in space, though not the geometry. We also tested the above numerical schemes using a fourth order geometry. In this case, the orders of convergence in Δt are the same as the ones reported for the cases in Figures 4.12(a) and 4.12(b), although the stability limitations on Δt are more severe. Again, in this case, the BDF*1* and BDF*2* schemes remain stable.

4.4.2 Using the Yosida-q schemes

In the benchmark using Yosida-q schemes, we considered $h = 0.5$, $\nu = 0.05$, $\rho = 1$ and $\gamma = 0$. The error quantities E_u and E_p are plotted in Figure 4.13. We notice that the

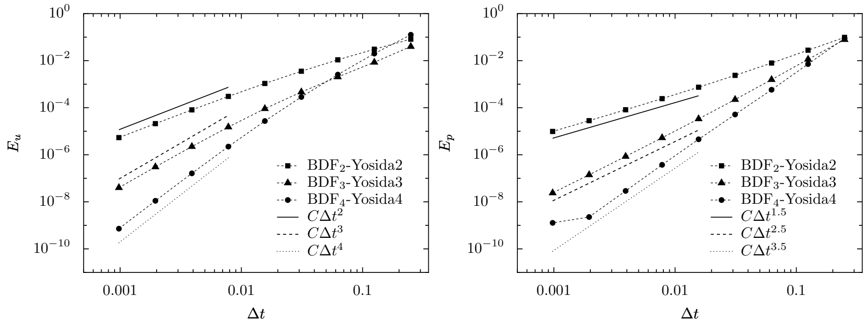


Figure 4.13: Plot of the errors E_u and E_p for $N = 2, M = 1, N_{\text{geo}} = 1$ and different BDFq and Yosida-q schemes.

convergence orders for the BDFq-Yosida-q are the same as the ones reported in section 3.4.

Similar tests were conducted using $\nu = 10^{-3}$. The convergence orders for the pressure were slightly better in this case, for the range of Δt considered. This was due to the fact that the splitting error introduced by the Yosida-q schemes was much smaller than the error introduced by the corresponding BDF method. Regarding the stability of the methods, we did not observe considerable differences between using BDFq and Yosida-q schemes, for $q = 2, 3, 4$.

4.5 Conclusion

In this chapter, we extended the Navier-Stokes solver from chapter 3 to the ALE framework. If the mesh velocity is approximated with the same BDF scheme as the velocity and the extrapolation formula, we showed that the same order of accuracy as for the fixed domain version is maintained. Similar tests were conducted with the Yosida versions of the solver and the results are the same as for the fixed domain version.

We highlight that the ALE map construction described in this chapter can be extended easily to more quadrangular meshes and account for more than vertical displacements only. The map's approximation properties in the moving boundary of the domain are of order $\mathcal{O}(N_{\text{geo}} + 1)$ in the $L^2(\Omega)$-norm.

The extension of the present ALE framework to 3D is, in general, straightforward. Regarding the approximation spaces, these are already available in the code used to produce all the simulations. Also, the Yosida-q and BDFq schemes do not depend on any geometrical quantities, but only on blocks of matrices or previous time step solutions. The tricky point here is to define a 3D high order map. This construction is easy if we consider first order finite elements. In the high order case, right now, the map's definition relies on Gordon-Hall transformations. The use of these transformations should be avoided in 3D due to their complicated expressions plus orientations issues that might appear to the programmer. Instead, a more general procedure should be devised to produce the high order nodes for the boundary elements.

Chapter 5

Application to Fluid-Structure Interaction in Hemodynamics

The goal of this chapter is to apply the algorithms developed in the previous chapters to simulate blood flow in large arteries. We restrict ourselves to a two dimensional setting and consider the fluid-structure interaction problem consisting of a thin elastic tube filled with an incompressible fluid.

The chapter is organized as follows. First, we introduce the system of equations that model the fluid-structure interaction (FSI) between the solid material and the fluid. This system is obtained from the coupling of an elasticity model, the incompressible Navier--Stokes equations in their ALE form, and suitable coupling conditions. Second, we present the numerical methods used to solve this coupled problem. We consider two approaches: an implicit fully coupled (FC) approach, where the unknowns related with the geometry are treated implicitly, and a semi-implicit (SI) one, where these quantities are treated in an explicit way. Both these methods rely on fixed point iterations. Finally, we present some results regarding the number of fix point iterations used by the FC and SI methods. We study the dependence of this quantity with respect to different BDF schemes for the structure and fluid solvers, different geometrical elements, polynomial degree to describe the structure displacement and velocity and pairs of spaces used to solve the fluid equations.

5.1 Model description

Let us consider a portion of an artery that, at time t, occupies the region Ω_t and its wall Σ_t. The interface between Σ_t and the blood vessel Ω_t, that we denote by Γ_t^w, together with the section S_1 and S_2 that connect the vessel with the rest of the cardiovascular system, form the boundary of Ω_t. Typically, the blood enters from S_1 and leaves the artery through S_2. In Figure 5.1 we present a simplified 2D model that describes schematically this situation.

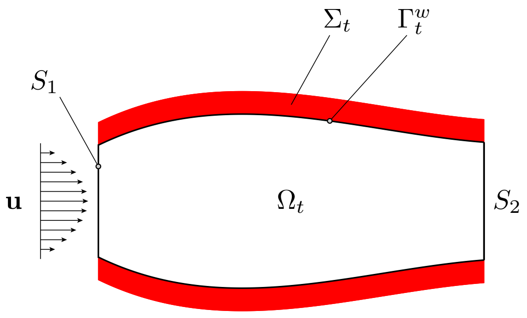


Figure 5.1: 2D simplified compliant tube.

5.1.1 Blood flow (fluid) model

In large arteries, blood can be approximated as a Newtonian fluid. We describe the flow through the unsteady Navier-Stokes equations, as presented in section 3.2. Let $t \in (0, T)$ and $\mathbf{x} \in \Omega_t$. We recall that if $\mathbf{u}(\mathbf{x}, t)$ is the fluid velocity and $p(\mathbf{x}, t)$ the pressure field then the unsteady incompressible Navier-Stokes equations in Ω_t read as

$$\rho \frac{\partial \mathbf{u}}{\partial t} + \rho(\mathbf{u} \cdot \boldsymbol{\nabla}_{\mathbf{x}})\mathbf{u} - \mathbf{div}_{\mathbf{x}}\left(\mathbf{T}(\mathbf{u}, p)\right) = \mathbf{f}, \quad \text{in } \Omega_t, \ t \in (0, T) \tag{5.1}$$

$$\text{div}_{\mathbf{x}}(\mathbf{u}) = 0, \quad \text{in } \Omega_t, \ t \in (0, T) \tag{5.2}$$

$\mathbf{T}$ is the *Cauchy stress tensor* and it is defined as

$$\mathbf{T}(\mathbf{u}, p) = -p\mathbf{I} + 2\nu \mathbf{D}_{\mathbf{x}}(\mathbf{u}) \tag{5.3}$$

Remark 5.1.1. *In smaller arteries, as for instance capillaries, the blood behavior is no longer Newtonian and a different constitutive law should be taken into account in* (5.3)*, see Chapter 6 in Formaggia, Quarteroni and Veneziani [31].*

System (5.1)-(5.2) has to be completed with suitable boundary conditions. We will consider Dirichlet or Neumann boundary conditions at the inlet,

$$\mathbf{u} = \mathbf{g} \quad \text{or} \quad \mathbf{T}\mathbf{n} = \mathbf{g}, \quad \text{on } S_1$$

and homogeneous Neumann boundary conditions at the outlet

$$\mathbf{T}\mathbf{n} = \mathbf{0}, \quad \text{on } S_2.$$

We highlight that this last condition is not physical. Indeed, since there are traveling waves in the compliant tube, this boundary condition introduces spurious reflections that have no physical meaning. It has been proposed in the literature to couple the FSI problem with one-dimensional models that act as absorbing devices as the wave exits the tube, see Nobile [57] and Quarteroni and Formaggia [67].

We address now the boundary conditions considered for Γ_t^w. Normally, the position of the interface Γ_t^w between the artery wall and the blood vessel is given by a displacement variable η with respect to a reference configuration of the artery, see Figure 5.2. Therefore,

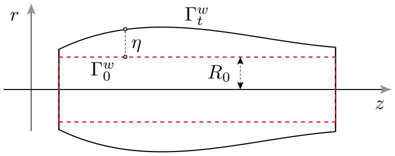


Figure 5.2: The displacement of the boundary w.r.t the reference configuration.

the condition imposed at Γ_t^w is the continuity of the velocity, i.e.,

$$\mathbf{u} = \dot{\eta}\mathbf{e}_r, \qquad \text{on } \Gamma_t^w \tag{5.4}$$

with some abuse in the notation, see remark 5.1.2.

If the displacement η is provided *a priori*, then we fall back into the problem already presented and analyzed in chapter 4. However, in fluid-structure interaction problems, as it is the case of the blood flowing through an artery, that is not the case. The position occupied by the wall, that we will refer as structure, is modeled by a differential equation.

5.1.2 Arterial wall (structure) model

We start by defining the set of points that describes the upper boundary of Figure 5.2,

$$\Gamma_0^w = \left\{(z,r) \in \mathbb{R}^2 :\; r = R_0,\; 0 \leqslant z \leqslant L\right\}.$$

Let $\eta : [0,L] \times \mathbb{R}_+ \longrightarrow \mathbb{R}$ be the vertical displacement of Γ_0^w and $\boldsymbol{\varphi}_\eta : [0,L] \times \mathbb{R}_+ \longrightarrow \mathbb{R}^2$ be defined as

$$\boldsymbol{\varphi}_\eta(z,t) = (z, R_0 + \eta(z,t)).$$

Then

$$\Gamma_t^w = \left\{(z,r) \in \mathbb{R}^2 :\; (z,r) = \boldsymbol{\varphi}_\eta(z,t),\; 0 \leqslant z \leqslant L,\; t \in I\right\}.$$

Remark 5.1.2. *With these new notations, the boundary condition at the wall of the vessel is*

$$\mathbf{u} = (\dot{\eta} \circ \boldsymbol{\varphi}_\eta^{-1})\mathbf{e}_r, \qquad \textit{on } \Gamma_t^w. \tag{5.5}$$

In this work, we will limit ourselves to consider a one dimensional model for the structure, the *generalized string model*, see Quarteroni, Formaggia and Veneziani [31]. We suppose that the *long wall assumption* holds, i.e.,

$$\frac{\partial \eta}{\partial z} \ll 1.$$

Note that without this assumption, the wall curvature would enter the equations in the form of

$$\sqrt{1 + \left(\frac{\partial \eta}{\partial z}\right)^2}$$

making them nonlinear. In this case, the displacement η is modeled by the following differential equation

$$\rho_w h \frac{\partial^2 \eta}{\partial t^2} - kGh \frac{\partial^2 \eta}{\partial z^2} + \frac{Eh}{1-\mu^2} \frac{\eta}{R_0^2} - \gamma_v \frac{\partial^3 \eta}{\partial z^2 \partial t} = f. \tag{5.6}$$

Here, h is the thickness of the structure, R_0 is the arterial reference radius at rest, k is the *Timoshenko shear correction factor*, G is the *shear modulus*, E is the *Young modulus*, μ is the *Poisson ratio*, ρ_w is the wall volumetric mass, γ_v is a viscoelastic parameter and f is an external force being applied to the structure, to be specified later in the context of the fluid-structure interaction.

5.1.3 Fluid-structure interaction problem

The coupling between the fluid equations and the structural equations is enforced at Γ_t^w through the continuity of the normal stress at the interface and the adherence of the fluid particles to the structure. The first coupling conditions implies that the force f is given by the radial component Φ_r of the normal stress exerted by the fluid, i.e.,

$$\Phi_r = -(\mathbf{Tn}) \cdot \mathbf{e}_r \circ \boldsymbol{\varphi}_\eta$$

while the second coupling condition is given by equation (5.5).

The final system of equations for the fluid-structure interaction problem reads as

$$\begin{aligned}
\rho \left.\frac{\partial \mathbf{u}}{\partial t}\right|_{\mathbf{Y}} + \rho((\mathbf{u}-\mathbf{w})\cdot\boldsymbol{\nabla}_{\mathbf{x}})\mathbf{u} - 2\nu\mathbf{D}_{\mathbf{x}}(\mathbf{u}) + \nabla p &= \mathbf{f}, \quad \text{in } \Omega_t,\ t \in I \\
\operatorname{div}_{\mathbf{x}}(\mathbf{u}) &= 0, \quad \text{in } \Omega_t,\ t \in I \\
\rho_w h \frac{\partial^2 \eta}{\partial t^2} - kGh \frac{\partial^2 \eta}{\partial z^2} + \frac{Eh}{1-\mu^2} \frac{\eta}{R_0^2} - \gamma_v \frac{\partial^3 \eta}{\partial z^2 \partial t} &= \Phi_r \quad \text{in } (0,L),\ t \in I \\
\mathbf{u} &= (\dot{\eta} \circ \boldsymbol{\varphi}_\eta^{-1})\mathbf{e}_r, \quad \text{on } \Gamma_t^w \\
-((\mathbf{Tn})\cdot\mathbf{e}_r) \circ \boldsymbol{\varphi}_\eta &= \Phi_r
\end{aligned}$$

supplied with initial boundary conditions

$$\begin{aligned} \mathbf{u} &= \mathbf{u}_0, \quad \text{for } t = t_0, \text{ in } \Omega_{t_0} \\ \eta = \eta_0, \left(\dot{\eta} \circ \boldsymbol{\varphi}_\eta^{-1}\right) \mathbf{e}_r &= \mathbf{u}_0, \quad \text{for } t = t_0, \text{ on } \Gamma_0^w. \end{aligned}$$

For this problem, we considered two types of boundary conditions for the inflow/outflow sections of the fluid and structure. The first set of boundary conditions reads as

$$\mathbf{Tn} = \mathbf{h}, \text{ on } S_1, \qquad \frac{\partial \eta}{\partial t} - \sqrt{\frac{kG}{\rho_w}} \frac{\partial \eta}{\partial z} = 0, \text{ at } z = 0 \tag{5.7}$$

$$\mathbf{Tn} = \mathbf{0}, \text{ on } S_2, \qquad \frac{\partial \eta}{\partial t} + \sqrt{\frac{kG}{\rho_w}} \frac{\partial \eta}{\partial z} = 0, \text{ at } z = L \tag{5.8}$$

In this case, the inflow profile is imposed as a Neumann boundary condition and the boundary conditions for the structure allow for the boundary of the structure to evolve with the movement of the fluid.

The other set of boundary conditions considered is

$$\mathbf{u} = \mathbf{h}, \text{ on } S_1 \qquad \mathbf{Tn} = \mathbf{0}, \text{ on } S_2 \qquad \eta = 0, \text{ at } z = 0, L.$$

In this case, the velocity is imposed as a Dirichlet boundary condition at the inflow and the boundary points of the structure are fixed.

Remark 5.1.3. *The spaces where we look for the velocity and pressure fields are* $\mathbf{V}^F(\Omega_t) = \mathbf{H}^1(\Omega_t)$ *and* $Q(\Omega_t) = L^2(\Omega_t)$. *On the other hand, the displacement* η *is sought in* $V^S(0, L) = H_0^1(0, L) \cap W^{1,\infty}(0, L)$.

The system of equations is non-linear, not only at the level of the fluid equations, but also because of the coupling.

Remark 5.1.4. *We highlight that the following variational formulation stands for the first set of boundary conditions presented, see equations* (5.7) *and* (5.8). *The case with Dirichlet boundary conditions is dealt as in the previous chapters.*

Following Nobile [57], we introduce a formal decoupling for this problem (already in the weak formulation) to solve separately the fluid and structure parts. The fluid sub-problem is

Problem 5.1.1. *find* $\mathbf{u}(t) \in \mathbf{V}^F(\Omega_t)$, $p(t) \in Q(\Omega_t)$ *such that*

$$\begin{aligned} &\rho \int_{\Omega_t} \left.\frac{\partial \mathbf{u}}{\partial t}\right|_{\mathbf{Y}} \cdot \mathbf{v} \, dx + \rho \int_{\Omega_t} \left[(\mathbf{u} - \mathbf{w}) \cdot \boldsymbol{\nabla}_{\mathbf{x}}\right] \mathbf{u} \cdot \mathbf{v} \, dx + 2\nu \int_{\Omega_t} \mathbf{D}_{\mathbf{x}}(\mathbf{u}) : \boldsymbol{\nabla}_{\mathbf{x}} \mathbf{v} \, dx \\ &+ \int_{\Omega_t} \mathrm{div}_{\mathbf{x}}(\mathbf{v}) \, p \, dx = \int_{\Omega_t} \mathbf{f} \cdot \mathbf{v} \, dx + \int_{S_1} \mathbf{h} \cdot \mathbf{v} \, ds, \qquad \forall \mathbf{v} \in \mathbf{V}(\Omega_t) \\ &\int_{\Omega_t} \mathrm{div}_{\mathbf{x}}(\mathbf{u}) \, q \, dx = 0, \qquad \forall q \in Q(\Omega_t) \\ &\mathbf{u} = \left(\dot{\eta} \circ \boldsymbol{\varphi}_\eta^{-1}\right) \mathbf{e}_r, \qquad \textit{on } \Gamma_t^w \end{aligned} \tag{5.9}$$

and structure sub-problem is

Problem 5.1.2. *find $\eta(t) \in V^S(0,L)$ such that*

$$\begin{aligned}
&\int_0^L \rho_w h \frac{\partial^2 \eta}{\partial t^2}\xi dz + \int_0^L \frac{Eh}{(1-\mu^2)R_0^2}\eta\xi dz \\
&+ \int_0^L \left(-kGh\frac{\partial \eta}{\partial z} - \gamma_v \frac{\partial^2 \eta}{\partial z \partial t}\right)\frac{\partial \xi}{\partial z} dz = \int_0^L \Phi_r \xi, \quad \forall \xi \in V^S(0,L).
\end{aligned} \tag{5.10}$$

For more details on these decoupling strategies, we refer the reader to Nobile [57].

5.2 Discretization

The numerical schemes that we are going to use to solve the FSI problem are based on the Arbitrary Lagrangian-Eulerian Navier-Stokes solver developed in the previous chapter and the schemes proposed in Nobile [57] and Deparis [18]. The idea is to use the decoupling presented in the previous section and iterate between the solution of the fluid problem 5.1.1 and the structure problem 5.1.2.

As we mentioned before, we consider two approaches: the implicit fully coupled, see section 5.2.3 and the semi-implicit, see section 5.2.4.

We detail now the numerical schemes used to solve the structure/fluid equations separately.

5.2.1 Structure solver

Let us suppose that $(\mathbf{u}, p)$ are known at a certain time step. With the velocity and pressure fields, we can calculate explicitly the normal stress at the moving boundary, ie,

$$\Phi_r = -(\mathbf{Tn}) \cdot \mathbf{e}_r \circ \boldsymbol{\varphi}_\eta, \quad \text{on } (0,L). \tag{5.11}$$

Introducing the notation

$$\dot{\eta} = \frac{\partial \eta}{\partial t}$$

we can rewrite equation (5.10) as

$$\begin{aligned}
&\int_0^L \rho_w h \frac{\partial \dot{\eta}}{\partial t}\xi dz + \int_0^L \frac{Eh}{(1-\mu^2)R_0^2}\eta\xi dz + \int_0^L \left(-kGh\frac{\partial \eta}{\partial z} - \gamma_v \frac{\partial \dot{\eta}}{\partial z}\right)\frac{\partial \xi}{\partial z} dz \\
&= \int_0^L \Phi_r \xi, && \forall \xi \in V^S(0,L) \\
&\int_0^L \dot{\eta}\zeta dz = \int_0^L \frac{\partial \eta}{\partial t}\zeta dz, && \forall \zeta \in V^S(0,L).
\end{aligned}$$

Applying the BDFq schemes to the previous system of equations, we obtain

$$\begin{aligned}
&\int_0^L \rho_w h \frac{\beta_{-1}}{\Delta t}\dot{\eta}^{n+1}\xi dz + \int_0^L \frac{Eh}{(1-\mu^2)R_0^2}\eta^{n+1}\xi dz \\
&+\int_0^L \left(-kGh\frac{\partial \eta^{n+1}}{\partial z} - \gamma_v \frac{\partial \dot{\eta}^{n+1}}{\partial z}\right)\frac{\partial \xi}{\partial z} dz = \int_0^L f^{n+1}\xi, \quad \forall \xi \in V^S(0,L) \\
&\int_0^L \dot{\eta}^{n+1}\zeta dz = \int_0^L \beta_{-1}\eta^{n+1}\zeta dz + \int_0^L g^{n+1}\zeta dz, \qquad \forall \zeta \in V^S(0,L).
\end{aligned} \tag{5.12}$$

where

$$f^{n+1} = \Phi_r^{n+1} + \sum_{j=0}^{q-1}\frac{\beta_j}{\Delta t}\dot{\eta}^{n-j}, \Phi_r^{n+1} = \Phi_r(t_{n+1}) \quad \text{and} \quad g^{n+1} = \dot{\eta}^{n+1} + \sum_{j=0}^{q-1}\frac{\beta_j}{\Delta t}\eta^{n-j}.$$

In these equations, the quantity f^{n+1} is the force function calculated with the fluid solution.

To have a fully discrete scheme, we can now replace the continuous space $V^S(0,L)$ by the spectral element space $\mathcal{F}_{N_s}(\mathcal{T}_h)$ (N_s is the polynomial degree used in discretizing the displacement and $\mathcal{T}_h$ is a mesh of the domain $[0,L]$). This induces a linear system that can be solved by standard techniques.

5.2.2 Fluid solver

Regarding the numerical method to solve the fluid equations (5.9), if we assume that the displacement η and its derivative $\dot{\eta}$ are known at time $n+1$, then we use the solver developed in the previous chapter. We recall the weak formulation of the fluid problem

Problem 5.2.1. *find the solution* $(\mathbf{u}_\delta^{n+1}, p_\delta^{n+1}) \in \mathbf{V}_\delta(\Omega_{t_{n+1},\delta}) \times Q_\delta(\Omega_{t_{n+1},\delta})$, *such that*

$$\begin{aligned}
&\rho\int_{\Omega_{t_{n+1}}} \frac{\beta_{-1}}{\Delta t}\mathbf{u}_\delta^{n+1}\cdot \mathbf{v}\ dx + \rho\int_{\Omega_{t_{n+1}}} \left[(\mathbf{u}_\delta^* - \mathbf{w}_\delta^*)\cdot \boldsymbol{\nabla}_\mathbf{x}\right]\mathbf{u}_\delta^{n+1}\cdot \mathbf{v}\ dx \\
&+\frac{\rho}{2}\int_{\Omega_{t_{n+1}}} \mathrm{div}_\mathbf{x}(\mathbf{u}_\delta^*)\mathbf{u}_\delta^{n+1}\cdot \mathbf{v}\ dx + 2\nu\int_{\Omega_{t_{n+1}}} \mathbf{D}_\mathbf{x}(\mathbf{u}_\delta^{n+1}) : \boldsymbol{\nabla}_\mathbf{x}\mathbf{v}\ dx + \\
&\int_{\Omega_{t_{n+1}}} \mathbf{div}_\mathbf{x}(\mathbf{v})\ p_\delta^{n+1}\ dx = \int_{\Omega_{t_{n+1}}} \tilde{\mathbf{f}}_\delta^{n+1}\cdot \mathbf{v}\ dx + \int_{S_1} \boldsymbol{h}\cdot \mathbf{v}\ ds, \qquad \forall \mathbf{v} \in \mathbf{V}_\delta^0(\Omega_{t_{n+1},\delta}) \\
&\int_{\Omega_{t_{n+1}}} \mathbf{div}_\mathbf{x}(\mathbf{u}_\delta^{n+1})\ q\ dx = 0, \qquad \forall q \in Q_\delta(\Omega_{t_{n+1},\delta})
\end{aligned} \tag{5.13}$$

where

$$\tilde{\mathbf{f}}_\delta^{n+1} = \mathbf{f}^{n+1} + \sum_{j=0}^{q-1}\frac{\beta_j}{\Delta t}\mathbf{u}_\delta^{n-j}$$

and the spaces $\mathbf{V}_\delta(\Omega_{t_{n+1},\delta})$, $\mathbf{V}_\delta^0(\Omega_{t_{n+1},\delta})$ *and* $Q_\delta(\Omega_{t_{n+1},\delta})$ *are defined as*

$$\mathbf{V}_\delta(\Omega_{t_{n+1},\delta}) = \left(\mathcal{F}_N(\mathcal{T}_{t_{n+1},\delta})\right)^2 \cap \left\{\mathbf{v} \in \mathbf{H}^1(\Omega_{t_{n+1},\delta}) :\ \mathbf{v}_{\Gamma_t^w} = \left(\dot{\eta}\circ \boldsymbol{\varphi}_\eta^{-1}\right)\mathbf{e}_r\right\},$$

$$\mathbf{V}^0_\delta(\Omega_{t_{n+1},\delta}) = \big(\mathcal{F}_N(\mathcal{T}_{t_{n+1},\delta})\big)^2 \cap \mathbf{H}^1_{\Gamma^w_t}(\Omega_{t_{n+1},\delta})$$

and

$$Q_\delta(\Omega_{t_{n+1},\delta}) = \mathcal{F}_M(\mathcal{T}_{t_{n+1},\delta}).$$

The quantity $\mathbf{w}^*_\delta$ *will be specified for each one of the two methods we are going to present.* $\mathbf{u}^*_\delta$ *are given according to* (3.34).

To use the interior penalty method we just need to add the stabilization term to the formulation. We highlight that the shape of the domain is known since η is known.

5.2.3 Implicit fully coupled (FC) FSI solver

The sub-structuring iteration algorithm, see Nobile [57], reads as follows:

Problem 5.2.2. *for each* t_{n+1}

(i) extrapolate the structure displacement: solve equations (5.12) *for structure displacement,* $\eta^{n+1}_{(0)}$ *(and* $\dot\eta^{n+1}_{(0)}$*) using the normal stress* Φ^n_r *from the previous time step.*

(ii) for $j = 1, \dots$ *(fix point iterations)*

  (a) given $\eta^{n+1}_{(j-1)}$*, calculate the ALE map, the computational domain and* $\mathbf{w}^{n+1}_\delta$

  (b) solve the Navier-Stokes equations (5.13) *with* $\mathbf{w}^*_\delta = \mathbf{w}^{n+1}_\delta$

  (c) calculate normal stress, $\Phi^{n+1}_{r,(j)}$*, in the moving boundary of the fluid*

  (d) solve equations (5.12) *for structure displacement,* $\eta^{n+1,*}$ *(and* $\dot\eta^{n+1,*}$*)*

  (e) if

$$\frac{\left\|\eta^{n+1,*} - \eta^{n+1}_{(j-1)}\right\|_{L_2}}{\left\|\eta^{n+1}_{(j-1)}\right\|_{L_2}} + \frac{\left\|\dot\eta^{n+1,*} - \dot\eta^{n+1}_{(j-1)}\right\|_{L_2}}{\left\|\dot\eta^{n+1}_{(j-1)}\right\|_{L_2}} < tolerance$$

  advance for the next time step with $\eta^{n+1} = \eta^{n+1,*}$ *and* $\dot\eta^{n+1} = \dot\eta^{n+1,*}$

  (f) otherwise (relaxation)

$$\eta^{n+1}_{(j)} = \theta\eta^{n+1,*} + (1-\theta)\eta^{n+1}_{(j-1)}$$

$$\dot\eta^{n+1}_{(j)} = \theta\dot\eta^{n+1,*} + (1-\theta)\dot\eta^{n+1}_{(j-1)}$$

For more details regarding this fix point type algorithm, called *Dirichlet-Neumann method*, we refer the reader to [57, 15, 19]. A full justification of the equivalence of solving the coupled FSI problem and the fix point problem is presented therein.

We highlight, as it is done in Nobile [57], that the relaxation parameter θ has to be chosen in an appropriate way for the fix point iteration to converge. In order to accelerate the convergence of this algorithm and not choose a priori a fixed relaxation parameter, we considered a strategy described in Deparis [18] using Aitken extrapolation. In this way, the parameter θ is calculated at each fix point iteration.

5.2.4 Semi-implicit (SI) FSI solver

A scheme of this type had already been proposed in Nobile [57]. The idea is, at each time step, freeze the domain where the fluid equations are solved in the first fix point iterations. Then, we iterate between solving the fluid and the structure, but passing to the fluid the displacement's velocity as boundary condition on the moving boundary.

The sub-structuring iteration algorithm reads as follows:

Problem 5.2.3. *for each t_{n+1}*

(i) $\eta_{(0)}^{n+1}$ and $\dot\eta_{(0)}^{n+1}$ are taken from the previous solution of the structure solver

(ii) calculate ALE map, fix computational domain and calculate $\mathbf{w}_\delta^n$

(iii) for $j = 1, \ldots$

(a) solve Navier-Stokes equations using $\dot\eta_{(j-1)}^{n+1}\mathbf{e}_2$ as boundary condition (on Σ_t) with $\mathbf{w}_\delta^* = \mathbf{w}_\delta^n$

(b) calculate normal stress in the moving boundary of the fluid

(c) solve for structure displacement, $\eta^{n+1,*}$ (and $\dot\eta^{n+1,*}$)

(d) if

$$\frac{\left\|\eta^{n+1,*} - \eta_{(j-1)}^{n+1}\right\|_{L_2}}{\left\|\eta_{(j-1)}^{n+1}\right\|_{L_2}} + \frac{\left\|\dot\eta^{n+1,*} - \dot\eta_{(j-1)}^{n+1}\right\|_{L_2}}{\left\|\dot\eta_{(j-1)}^{n+1}\right\|_{L_2}} < \textit{tolerance}$$

advance for the next time step with $\eta^{n+1} = \eta^{n+1,*}$ and $\dot\eta^{n+1} = \dot\eta^{n+1,*}$

(e) otherwise (relaxation)

$$\eta_{(j)}^{n+1} = \theta\eta^{n+1,*} + (1-\theta)\eta_{(j-1)}^{n+1}$$

$$\dot\eta_{(j)}^{n+1} = \theta\dot\eta^{n+1,*} + (1-\theta)\dot\eta_{(j-1)}^{n+1}$$

A few remarks are in order:

Remark 5.2.1. *The parameter θ in the semi-implicit method is also chosen through Aitken extrapolation.*

Remark 5.2.2. *Since the computational domain is fixed for each time step, in each iterations of the fix point method, the fluid equations have to be solved always in the same geometry. Therefore, the matrices used to solve the Navier-Stokes equations have to be assembled only once per time step. This reduces considerably the computational cost of this method, compared with the FC method.*

Remark 5.2.3. *A particularity of both FC and SI method is that, if different BDF schemes are used to approximate the time derivatives in the fluid and structure equations, the displacement's velocity approximation passed from the structure to the fluid solver is always of the order of the fluid's time approximations. This means that, for instance, if we take BDF3 to approximate the time derivatives in the structure and BDF1 for the fluid, the displacement's velocity* $\dot{\eta}^{n+1}_{(j-1)}\mathbf{e}_2$ *passed to the fluid solver will actually be a first order in time approximation, and not third order. This is a particularity of how the solver was built and is valid both for the FC and SI methods. The impact of this choice in the stability of the methods is unknown.*

5.3 Numerical results

As mentioned before, we performed several numerical tests with the fluid-structure solvers. For the FC method (which was the first one to be implemented) we considered two types of boundary conditions at the inflow, Dirichlet and Neumann. For the SI method, we only considered the latter.

We considered the following problem. The initial domain is a 2D rectangle of height $D = 1cm$ and length $L = 6cm$. The upper and lower boundaries of the domain are deformable in the vertical direction according to equation (5.6). The fluid and structure are at rest at time $t = 0$.

The parameters used for the fluid problem are $\nu = 0.035 poise$ and $\rho = 1gr/cm^3$. As for the structure we considered: density $\rho_w = 2gr/cm^3$, thickness $h = 0.1cm$, Young modulus $E = 0.75 \cdot 10^6 dyne/cm^2$, Poisson coefficient $\mu = 0.5$, reference radius $R_0 = 0.5$, shear modulus $G = 10^5$, Timoshenko shear factor $k = 2.5$ and viscoelastic parameter $\gamma_v = 0.01$. This problem is the same considered in Nobile [57] and Deparis [18], except for the wall density value. In our case, we consider a denser wall to avoid problems with the added mass effect. We simulated the pulse for $10ms$.

5.3.1 Implicit fully coupled (FC) solver

We describe now the results obtained with the FC method, both for Dirichlet and Neumann boundary conditions at the inflow, as well as several other tests.

Using inflow Neumann boundary conditions

In this case, we impose the following Neumann condition at the inflow boundary

$$\mathbf{Tn} = -10^4 \left[1 - \cos\left(\frac{\pi t}{2.5ms}\right)\right] \mathbf{n}.$$

In our numerical tests we always obtained instabilities using a Neumann boundary condition, except in the cases where the viscosity ν was big or the magnitude of pressure pulse imposed at the inlet was at least one order of magnitude smaller than 10^4. We noticed that these instabilities were formed at the inlet boundary where the Neumann condition was imposed and after the pressure wave entered the tube. We believe that this behavior is caused by the lack of control of the velocity field at the inlet; since the boundary condition is of Neumann type, the degrees of freedom at the inlet should also be taken into account in the stabilization. This does not happen because the interior penalty stabilization only takes care of the interior faces of the domain.

Once we enforced a similar pulse at the inlet as a Dirichlet boundary condition (second test case), the instabilities disappeared and we successfully simulated the fluid-structure interaction.

Using inflow Dirichlet boundary conditions

We changed the original problem only by imposing a Dirichlet velocity profile at the inlet given by:

$$\begin{aligned}\mathbf{u}(0,y,t) &= 343.99(0.25-y)^2(-1357t^9+7443t^8-17099t^7+21255t^6\\ &\quad -15356t^5+6379t^4-1368t^3+97t^2+6t)\mathbf{e}_1, \text{ with } \mathbf{e}_1=(1,0)^T.\end{aligned} \tag{5.14}$$

This initial condition comes from realistic data measured in a pig. The velocity was scaled to have the same magnitude as the velocity profile that was obtained using Neumann boundary conditions at the inflow.

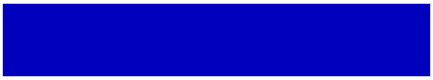
We performed successfully several simulations, with different time/space discretizations, namely, $\mathbb{P}_N-\mathbb{P}_{N-2}$ finite elements ($N=3,4,5$) for the fluid, $\mathbb{P}_1$ and $\mathbb{P}_2$ geometries and BDFq, $q=1,2,3,4$ as time integrator. The structure model was discretized with $\mathbb{P}_N$ elements, $N=1,2$. In Figure 5.3 we plot the pressure field associated with the inlet profile (5.14) at several time steps. We can observe the pressure wave propagating through the pipe due to the elastic behavior of the walls.

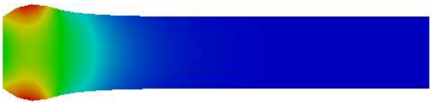
The solver was tested by varying some of the parameters at the discretization level. In the following, we present the results obtained.

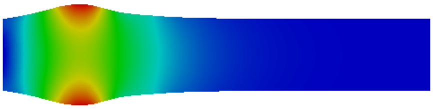
Varying N_s and N_{geo}. In Figure 5.4 we plot the residual

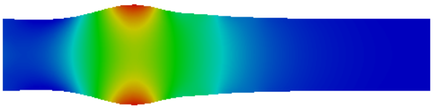
$$\frac{\left\|\eta^{n+1,*}-\eta^{n+1}_{(j-1)}\right\|_{L_2}}{\left\|\eta^{n+1}_{(j-1)}\right\|_{L_2}}+\frac{\left\|\dot\eta^{n+1,*}-\dot\eta^{n+1}_{(j-1)}\right\|_{L_2}}{\left\|\dot\eta^{n+1}_{(j-1)}\right\|_{L_2}}$$

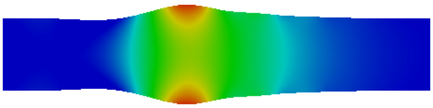
of the fix point method for the first time step of the simulation. We considered the following spaces/parameters: fluid discretized with BDF*1* and $\mathbb{P}_4-\mathbb{P}_2$ elements, $h=0.2$, no stabilization and $tol=10^{-6}$; for the structure we used BDF*1*. We observe that the more we increase the polynomial degree associated with the geometry and/or the structure, the convergence of the fix point algorithm becomes slower (and in some cases extremely slow). When using second order geometrical elements for the fluid and second degree polynomials for the structure's displacement, the residual eventually hits the tolerance prescribed, but

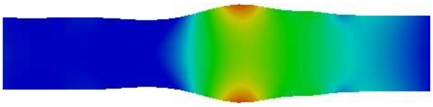

(a) $t = 0ms$

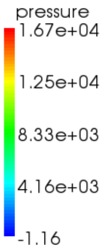

(b) $t = 2ms$


(c) $t = 4ms$


(d) $t = 6ms$

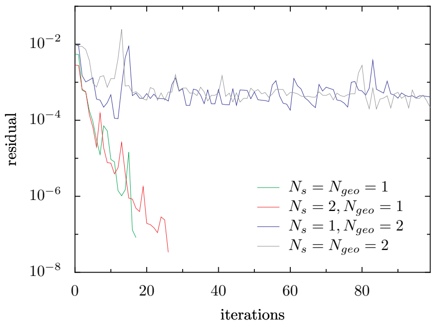

(e) $t = 8ms$


(f) $t = 10ms$

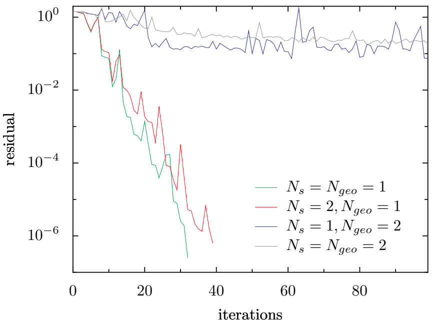


(g) Legend

Figure 5.3: Pressure pulse propagating through the pipe. Fluid discretized with BDF*3* and $\mathbb{P}_4 - \mathbb{P}_2$ elements, $h = 0.2$, $\Delta t = 10^{-4}$, $\gamma = 0.01$ and $tol = 10^{-6}$. For the structure we used BDF*3* and $\mathbb{P}_1$ elements. The displacement is magnified five times.



(a) $t_n = 0.1ms$



(b) $t_n = 1ms$

Figure 5.4: Residual of the fix point iteration.

at around 10000 iterations. As we can see from Figure 5.4, not even reducing the time step (in one order of magnitude) changes this behavior.

This happens due to the added mass effect [14] and the increasing stiffness of the problem using higher degree polynomials for the interface between structure and fluid.

We also ran the same tests using $\rho_w = 1.1$. This problem is the same as the one solved in Nobile [57] and Deparis [18]. The results obtained are slightly different from the above regarding the convergence of the $(N_s, N_{\text{geo}}) = (2, 1)$ case. While in the above case, the convergence of $N_s = N_{\text{geo}} = 1$ and $(N_s, N_{\text{geo}}) = (2, 1)$ are similar, with the denser wall, the method using $(N_s, N_{\text{geo}}) = (2, 1)$ takes much more iterations to converge. This is again a sign of the added mass effect.

Remark 5.3.1. *We highlight that the simulation using geometrical elements of order one and second order elements for the structure displacement's discretization was conducted until $T = 10ms$ as all the simulations in this chapter. For each time step, the fix point method converged to the established tolerance.*

Varying BDF*q* for fluid and structure. As a second test, we fixed the polynomial spaces used for the spatial discretization and varied the order of the integration methods for the fluid and structure discretization. In Figure 5.5 we show the evolution of the fix point iterations over time using the BDF*1*, BDF*2* and BDF*3* methods in the fluid discretization. The choice of parameters is the same as in the previous case. The results show that for a fixed time integration order for the fluid, increasing the integration order for the structure solver decreases the number of iterations for the fix point method. The same behavior is observed using BDF*4* to integrate in time the fluid equations. We conducted the same tests using the $\mathbb{P}_2$-$\mathbb{P}_1$ and the results are similar.

We plot in Figure 5.6 the average number of iterations for the $\mathbb{P}_2$-$\mathbb{P}_1$ and $\mathbb{P}_4$-$\mathbb{P}_2$ methods. Each set of columns represents a choice for the time integrator for the fluid and in each set,

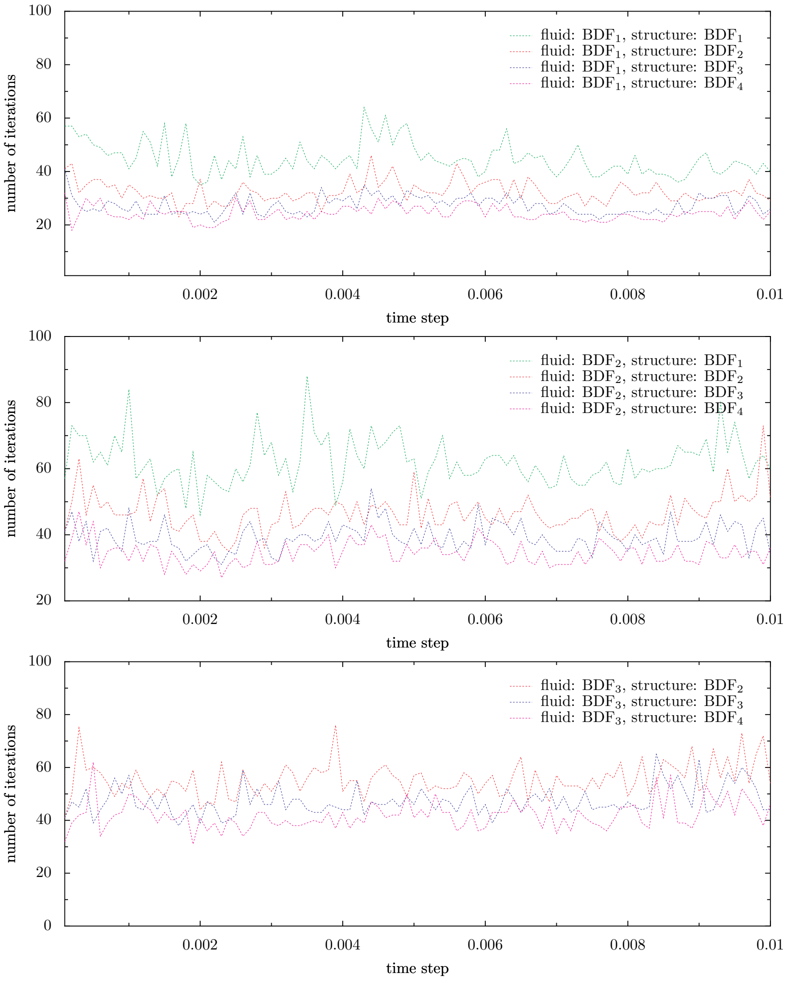
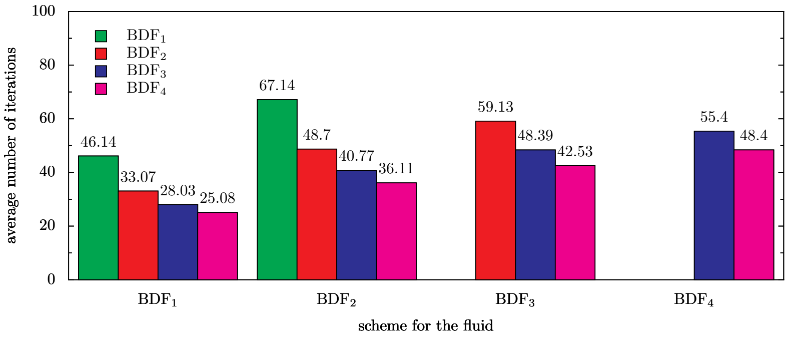
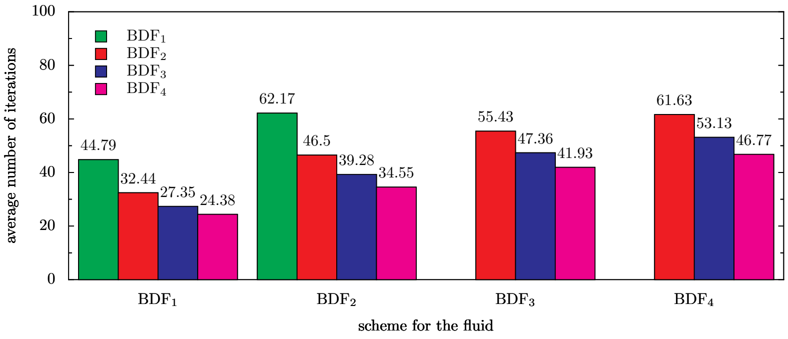


Figure 5.5: Evolution of the number of iterations of the fix point method throughout a simulation using the $\mathbb{P}_4$-$\mathbb{P}_2$.

from right to left, it represents the choice of the BDF*4*, BDF*3*, BDF*2* and BDF*1* methods for the structure solver. The colors chosen are consistent with Figure 5.5. In these figures,



(a) $\mathbb{P}_2$-$\mathbb{P}_1$ method



(b) $\mathbb{P}_4$-$\mathbb{P}_2$ method

Figure 5.6: Average number of iterations for the fix point method, for several time integration methods for the fluid.

we confirm what we mentioned before regarding the decrease of fix point iterations when increasing the order of the BDF*q* method for the structure.

On the other side, if we fix the order of the structure time integrator and vary the order of the fluid's BDF*q* scheme, in any of the cases, the number of iterations increases, in average. When using the BDF*1* method for the structure, if we choose BDF*3* or BDF*4* for the fluid discretization, the fix point method stops converging to the prescribed tolerance.

We highlight an interesting fact that can be directly observed from Figures 5.6(a) and 5.6(b). If we look at the number of fix point iterations taken by schemes that use the same order in time for fluid and structure, that is, schemes where BDF*q* is used for fluid and structure, the average number of iterations is more or less the same.

Varying the polynomial order for the velocity and pressure spaces. The number of iterations used by the fix point method is also reported when considering several discretization pairs of spaces for velocity and pressure. In this test, we took the $\mathbb{P}_N$-$\mathbb{P}_{N-1}$ and $\mathbb{P}_N$-$\mathbb{P}_{N-2}$ methods for N ranging from 2 to 5 in the first case, and from 3 to 5 in the second. The fluid was discretized using BDF*1* and the structure with BDF*4*. The results are displayed in Figure 5.7. We conclude that the number of iterations isn't affected by

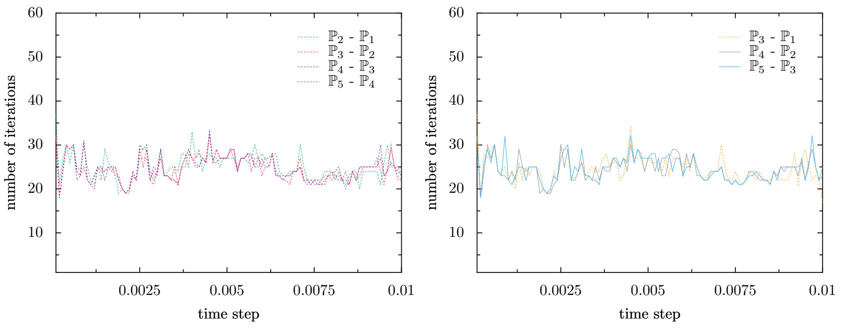


Figure 5.7: Average number of iterations for the fix point method, for several $\mathbb{P}_N$-$\mathbb{P}_{N-1}$ (left) and $\mathbb{P}_N$-$\mathbb{P}_{N-2}$ (right) methods.

the discretization spaces, at least for the polynomial orders considered.

5.3.2 Semi-implicit (SI) solver

We performed the same batch of tests to the semi-implicit solver. We start by analyzing the dependency of the number of iterations of the fix point method with respect to the order of the geometrical elements of the fluid mesh and the polynomial degree of the approximation of the displacement.

Varying N_s and N_{geo}. The behavior of the semi-implicit solver in terms of these quantities is similar to the FC solver. Regarding the choices all the choices but $N_{\text{geo}} = N_s = 2$, the solvers perform more or less the same (apart from an increase in the number of iterations that the SI solver needs to achieve the prescribed tolerance). However, when using second order geometrical elements for the fluid and the $\mathbb{P}_2$ to approximate the displacement and the velocity's displacement of the structure, the SI solver converges not only in the first time step, but throughout the whole test simulation. We plot in Figure 5.8 the residual for $t_n = 1ms$.

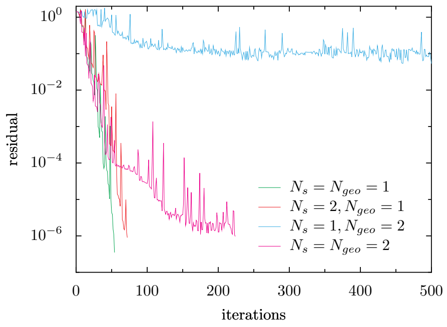


Figure 5.8: Residual of the fix point iteration for $t_n = 1ms$.

We show some results with the SI method and second order geometrical elements (with the fixed set of parameters chosen) in Figure 5.9. In this figure, we present a few snapshots of the pressure evolution in time. The rest of the simulation is as in Figure 5.3, apart from the dissipative behavior of BDF*1*. There are some artifacts in Figure 5.9, but this is due

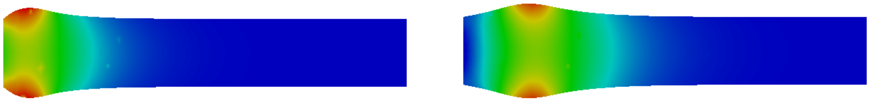


Figure 5.9: Pressure pulse propagating through the pipe. Fluid discretized with BDF*1* and $\mathbb{P}_4 - \mathbb{P}_2$ elements, $h = 0.2$, $\Delta t = 10^{-4}$, $\gamma = 0.01$ and $tol = 10^{-4}$. For the structure we used BDF*1* and $\mathbb{P}_2$ elements. The displacement is magnified five times. The legend for these figures is the same as in Figure 5.3.

to the routine that exports the solutions and not an error of computation of the method.

Varying the polynomial order for the velocity and pressure spaces. As it happened with the FC method, the choice of different pairs of approximation spaces for velocity and pressure does not influence much the number of iterations. We plot in Figure 5.10 the number of fix point iterations necessary to achieve the usual tolerance during a simulation. The results are twice as big as the ones obtained for the FC method, see Figure 5.7.

Varying BDF*q* for fluid and structure. Notice that in average, the SI method takes more iterations to converge than the FC method. Looking at the average number of iterations of the methods that use BDF*q* for fluid and structure, we remark that also the SI method maintains the number of iterations fixed. The average number of iterations is about three times bigger for the SI method than for the FC method.

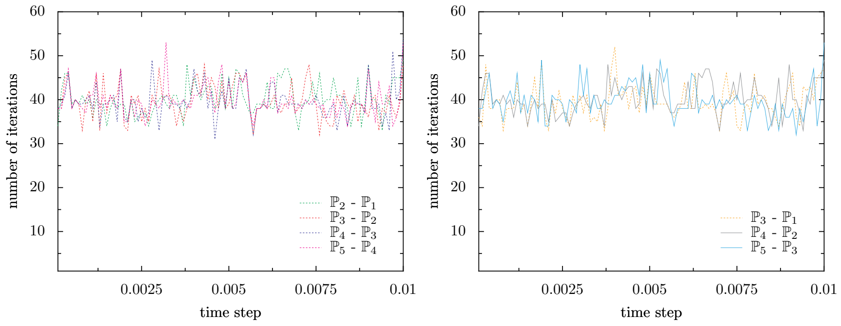
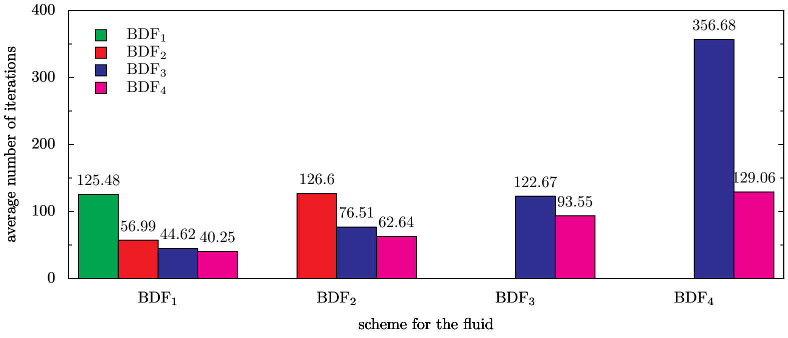
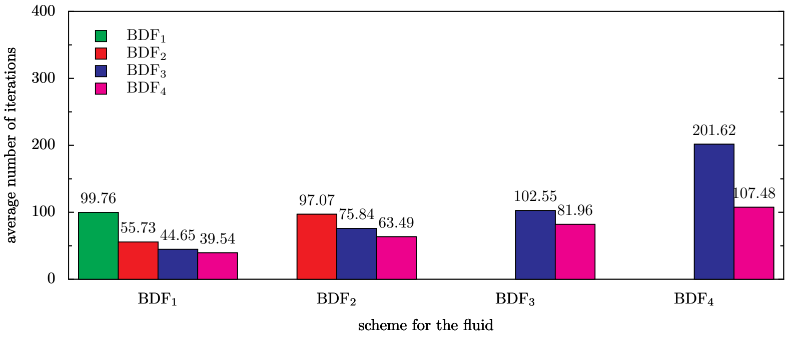


Figure 5.10: Average number of iterations for the fix point method, for several $\mathbb{P}_N$-$\mathbb{P}_{N-1}$ (left) and $\mathbb{P}_N$-$\mathbb{P}_{N-2}$ (right) methods.

From Figures 5.11(a) and 5.11(b) we stress that the gap of iterations that exists between using the SI method combined the $\mathbb{P}_2$-$\mathbb{P}_1$ method and the $\mathbb{P}_4$-$\mathbb{P}_2$ method is bigger than in the FC method. There is a reduction in the number of iterations that is more significant than in the results shown in Figures 5.6(a) and 5.6(b).



(a) $\mathbb{P}_2$-$\mathbb{P}_1$ method



(b) $\mathbb{P}_4$-$\mathbb{P}_2$ method

Figure 5.11: Average number of iterations for the fix point method, for several time integration methods for the fluid.

Chapter 6

Conclusions

In this thesis we addressed the numerical approximation of the incompressible Navier--Stokes equations evolving in a moving domain with the spectral element method. As an application, we solved a simple fluid-structure problem in the context of hemodynamics.

First, we described the polynomial framework behind the spectral element method. One crucial aspect in this construction was the choice of points to differentiate the polynomial bases and the conditioning of the generalized Vandermonde matrix associated with this issue. A numerical study was conducted to assess which points should be used in order that the methodology was numerically stable. We concluded that the Fekete points were the best choice in 1D and 2D and the Warpblend points in 3D. We highlight that these results were limited to the point sets taken for comparison.

The multidomain case was also addressed and we showed how to construct a map that relates the local degrees of freedom with the global ones. This construction was made in nD, $n = 1, 2, 3$, for all the reference elements considered, and in the case of modal and nodal bases. A Poisson test problem was considered as benchmark to determine the growth of the eigenvalues or the conditioning of the stiffness matrix for the different Lagrange based bases. Again, the best results were obtained with the Lagrange basis associated with either Fekete or Warpblend points, in 1D and 2D, and 3D, respectively.

Second, we introduced the incompressible Navier-Stokes equations in a fixed domain and its approximation by the spectral element method. We considered inf-sup stable, arbitrary order polynomial bases to discretize velocity and pressure. The geometry of the domain was considered to be approximated with curved elements. In the test case we presented based on the Kovasznay solution, we highlight the importance of using high order geometrical elements, as being the only way to, in curved geometries, achieve spectral accuracy.

The discrete problem that was obtained after time-space discretization (or only space discretization, in the steady case) of these equations is quite difficult to solve. We proposed two strategies to deal with the solution of this system: a block type preconditioner and the Yosida-q methods. It was shown that the preconditioner is optimal (regarding the number of iterations used by the GMRES method) w.r.t. the polynomial order for the velocity/pressure and mesh size, in the steady Navier-Stokes frame. In the unsteady case, optimality was lost regarding the polynomial order. The dependence was mild w.r.t. the viscosity.

We compared the block type preconditioner with other strategies and showed that for the range of problems that we considered, a LU factorization of the matrix is a faster method of solution. Regarding the Yosida-q methods, we showed the order of convergence in time for velocity and pressure. The results obtained are in agreement with the ones in the literature for this type of methods.

Next, we addressed the incompressible Navier-Stokes equations evolving in a moving domain. We generalized the discretization method used in the fixed domain case to the Arbitrary Lagrangian Eulerian framework. The construction of a high order ALE map was explained in detail. After space-time discretization of the equations, the linear system obtained is solved using a LU factorization or the Yosida-q methods. The order of convergence of both approaches, in time, using both methods, was the same as the fixed domain version.

Finally, we considered a 2D fluid-structure interaction problem with a one dimensional structure. The strategies to solve the coupled problem were an implicit fully coupled method and a semi-implicit method. We showed for the FC method that the residual of the fix point method converges to a prescribed tolerance when the geometrical elements are straight and the structure is discretized with first or second degree polynomials. Increasing the order of the geometrical elements for the fluid makes the fix point method converge very slowly, making this approach unfeasible to use in practice. We were able to circumvent this drawback of the FC method with the SI scheme. Although in general the semi-implicit method takes more iterations to converge and has more severe stability restrictions, its behavior regarding second order elements is satisfactory, at least compared with the FC method.

Summarizing, the simulation of fluid-structure interaction problems using high order in time and space methods is far from being complete. Regarding algorithms to solve the coupled problem, we conclude that the fix point method, combined with an implicit or semi-implicit approach, takes too much time to convergence to some prescribed tolerance. We do not provide timings with this respect (only the number of iterations), but taking into account that in the FC method the matrix solving the fluid equations has to be reassembled, makes it extremely expensive. On the other hand, the semi-implicit approach does not have this drawback, but it has more severe time stability restrictions. In the future, other semi-implicit approaches or even monolithic methods (using the Newton or quasi-Newton methods) might be of interest to test in this framework to go around the slow convergence of the fix point method, see Chapter 9 of Formaggia, Quarteroni and Veneziani [31].

Another crucial point is the extension of the framework to 3D geometries. This is strongly dependent on two things: first, the construction of a high order ALE map in 3D; and second, the solution algorithms for the fluid solver (or in the case of a non modular approach, the whole fluid-structure system). These are issues that are not considered in this thesis, but that should be looked at in the future.

Bibliography

[1] M. Ainsworth and P. Coggins. A uniformly stable family of mixed hp-finite elements with continuous pressures for incompressible flow. *IMA Journal of Numerical Analysis*, 22(2):307–327, April 2002.

[2] C. Bernardi and Y. Maday. Uniform inf-sup conditions for the spectral discretization of the Stokes problem. *Math. Models Meth. Appl. Sci.*, 9(3):395–414, 1999.

[3] L. Bos, M. A. Taylor, and B. A. Wingate. Tensor product Gauss-Lobatto points are Fekete points for the cube. *Math. Comput.*, 70(236):1543–1547, 2001.

[4] R. Bouffanais. *Simulation of shear-driven flows: transition with a free surface and confined turbulence.* PhD thesis, n°3837, EPF Lausanne, 2007.

[5] F. Brezzi. On the existence, uniqueness and approximation of saddle-point problems arising from lagrangian multipliers. *R.A.I.R.O.*, 8:129–151, 1974.

[6] F. Brezzi and R. S. Falk. Stability of higher-order Hood-Taylor methods. *SIAM J. Numer. Anal.*, 28(3):581–590, 1991.

[7] E. Burman and A. Ern. Continuous interior penalty hp-finite element methods for advection and advection-diffusion equations. *Math. Comp*, 76(259):119–1140, 2007. EPFL-IACS report 02.2005.

[8] E. Burman and M. A. Fernandez. Continuous interior penalty finite element method for the time-dependent Navier-Stokes equations: space discretization and convergence. *Numer. Math.*, 107(1):39–77, 2007.

[9] J. Cahouet and J. P. Chabard. Some fast 3D finite element solvers for the generalized Stokes problem. *Int. J. Numer. Methods Fluids*, 8(869), 1988.

[10] C. Canuto, M. Y. Hussani, A. Quarteroni, and T. A. Zang. *Spectral Methods in Fluid Dynamics.* Springer-Verlag, New York and Berlin, 2nd printing edition, 1988.

[11] C. Canuto, M. Y. Hussani, A. Quarteroni, and T. A. Zang. *Spectral Methods: Evolution to Complex Geometries and Applications to Fluid Dynamics.* Springer-Verlag, New York and Berlin, 2006.

[12] C. Canuto, M. Y. Hussani, A. Quarteroni, and T. A. Zang. *Spectral Methods: Fundamentals in Single Domains.* Springer-Verlag, New York and Berlin, 2006.

[13] M. Casarin. Schwarz preconditioners for the spectral element discretization of the steady stokes and navier-stokes equations. *Numer. Math.*, 2001.

[14] P. Causin, J. F. Gerbeau, and F. Nobile. Added-mass effect in the design of partitioned algorithms for fluid-structure problems. *Comput. Methods Appl. Mech. Eng.*, 194:4506–4527, 2005.

[15] M. Cervera, R. Codina, and M. Galindo. *On the Computational Efficiency and Implementation of Block-Iterative Algorithms for Nonlinear Coupled Problems.* CIMNE (International Center For Numerical Methods in Engineering, Barcelona, 1993.

[16] P. G. Ciarlet. *The Finite Element Method for Elliptic Problems.* SIAM, 2002.

[17] T. A. Davis. *Direct Methods for Sparse Linear Systems (Fundamentals of Algorithms 2).* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2006.

[18] S. Deparis. *Numerical Analysis of Axisymmetric Flows and Methods for Fluid Structure Interaction Arising in Blood Flow Simulation.* PhD thesis, n°2965, EPF Lausanne, 2004.

[19] S. Deparis, M. Fernández, and L. Formaggia. Acceleration of a fixed point algorithm for fluid-structure interaction using transpiration conditions. *Mathematical Modelling and Numerical Analysis*, 37(4):601–16, 2003.

[20] M. Deville, P. F. Fischer, and E. H. Mund. *High-Order Methods for Incompressible Fluid Flow.* Cambridge University Press, Cambridge, 2002.

[21] V. Dolean, R. Pasquetti, and F. Rapetti. p-Multigrid for Fekete spectral element method. In *"Domain Decomposition Methods in Science and Engineering", LNCSE*, page to appear. Springer Verlag, 2007.

[22] J. Donea, S. Giuliani, and J. P. Halleux. An arbitrary Lagrangian-Eulerian finite element method for transient dynamic fluid-structure interactions. *Computer Methods in Applied Mechanics and Engineering*, 33(1-3):689–723, 1982.

[23] D. L. Donoho, A. Maleki, I. U. Rahman, M. Shahram, and V. Stodden. Reproducible research in computational harmonic analysis. *Computing in Science and Engineering*, 11(1):8–18, 2009.

[24] M. Dubiner. Spectral Methods on triangles and other domain. *J. Sci. Comput.*, 6(4):345–390, 1991.

[25] H. Elman and D. Silvester. Fast Nonsymmetric Iterations and Preconditioning for Navier–Stokes Equations. *SIAM J. Sci. Comput.*, 17(1):33–46, 1996.

[26] A. Ern and J. Guermond. *Theory and practice of finite elements.* Springer, 2004.

[27] C. Farhat, P. Geuzaine, and C. Grandmont. The discrete geometric conservation law and the nonlinear stability of ALE schemes for the solution of flow problems on moving grids. *J. Comput. Phys.*, 174(2):669–694, 2001.

[28] L. Fejeér. Lagrangesche interpolation und die zugehörigen kongjugierten punkte. *Math. Ann*, 106:1–55, 1932.

[29] M. A. Fernández, 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(4):794 – 821, 2006.

[30] P. F. Fischer and J. W. Lottes. Hybrid Schwarz-Multigrid methods for the spectral element method: Extensions to Navier-Stokes. In *In Proceedings of the Fifteenth International Conference on Domain Decomposition Methods*, pages 35–49, 2003.

[31] L. Formaggia, A. Quarteroni, and A. Veneziani, editors. *Cardiovascular Mathematics: Modeling and simulation of the circulatory system*, volume 1 of *MS & A*. Springer, 2009.

[32] A. Gauthier, F. Saleri, and A. Veneziani. A fast preconditioner for the incompressible Navier-Stokes equations. *Comput. Vis. Sci.*, 6(2-3):105–112, 2004.

[33] P. Gervasio. Convergence Analysis of High Order Algebraic Fractional Step Schemes for Time-Dependent Stokes Equations. *SIAM J. Numer. Anal.*, 46(4):1682–1703, 2008.

[34] P. Gervasio, F. Saleri, and A. Veneziani. Algebraic fractional-step schemes with spectral methods for the incompressible Navier–Stokes equations. *J. Comput. Phys.*, 214(1):347–365, 2006.

[35] C. Geuzaine and J.-F. Remacle. Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities. *International Journal for Numerical Methods in Engineering*, 79(11):1309–1331, 2009.

[36] W. J. Gordon and C. A. Hall. Construction of curvilinear coordinate systems and their applications to mesh generation. *Int. J. Numer. Meth. Eng.*, 7:461–477, 1973.

[37] W. J. Gordon and C. A. Hall. Transfinite element methods: blending-function interpolation over arbitrary curved element domains. *Numer. Math.*, 21:109–129, 1973.

[38] D. Gottlieb and S. A. Orszag. Numerical analysis of spectral methods: theory and applications. *SIAM-CMBS*, 1977.

[39] J. L. Guermond, P. Minev, and J. Shen. Error analysis of pressure-correction schemes for the time-dependent stokes equations with open boundary conditions. *SIAM J. Numer. Anal.*, 43(1):239–258, 2005.

[40] J. L. Guermond and J. Shen. A new class of truly consistent splitting schemes for incompressible flows. *J. Comput. Phys.*, 192(1):262–276, 2003.

[41] J. L. Guermond and J. Shen. Velocity-correction projection methods for incompressible flows. *SIAM J. Numer. Anal.*, 41(1):112–134, 2003.

[42] M. Heil. An efficient solver to the fully coupled solution of large displacement fluid-structure interaction problems. *Comput. Methods Appl. Mech. Engrg.*, 193:1–23, 2004.

[43] W. Heinrichs. Improved Lebesgue constants on the triangle. *J. Comput. Phys.*, 207(2):625–638, 2005.

[44] J. S. Hesthaven and C. H. Teng. Stable spectral methods on tetrahedral elements. *SIAM J. Sci. Comput.*, 21(6):2352–2380 (electronic), 2000.

[45] L. W. Ho and E. M. Rønquist. Spectral element solution of steady incompressible viscous free-surface flows. *Finite Elem. Anal. Des.*, 16(3-4):207–227, 1994.

[46] P. Houston, C. Schwab, and E. Süli. Discontinuous hp-finite element methods for advection-diffusion-reaction problems. *SIAM J. Numer. Anal.*, 39(6):2133–2163, 2001.

[47] T. J. R. Hughes, W. K. Liu, and T. K. Zimmermann. Lagrangian-Eulerian finite element formulation for incompressible viscous flows. *Computer Methods in Applied Mechanics and Engineering*, 29(3):329–349, 1981.

[48] G. E. Karniadakis and S. J. Sherwin. *Spectral/hp element methods for computational fluid dynamics.* Oxford Universtity Press, Oxford, 2nd ed. edition, 2004.

[49] G. E. Karniadakis and S. J. Sherwin. *Spectral/hp element methods for computational fluid dynamics*, chapter Multi-dimensional formulation, pages 222–234. Oxford Universtity Press, Oxford, 2nd ed. edition, 2004.

[50] D. Kay, D. Loghin, and A. Wathen. A Preconditioner for the Steady–State Navier–Stokes Equations. *SIAM J. Sci. Comput.*, 24(1):237–256, 2002.

[51] D. Kay and E. Lungu. A Block Preconditioner for High–Order Mixed Finite Element Approximations to the Navier–Stokes Equations. *SIAM J. Sci. Comput.*, 27(6):1867–1880, 2006.

[52] L. I. G. Kovasznay. Laminar flow behind a two-dimensional grid. *Proc. Camb. Philos. Soc.*, 44:58–62, 1948.

[53] M. Lesoinne and C. Farhat. Geometric conservation laws for aeroelastic computations using unstructured dynamics meshes. AIAA-95-1709, Presented at the 12th AIAA Computational Fluid Dynamics Conference, San Diego, june 1995.

[54] D. J. Mavriplis and Z. Yang. Construction of the discrete geometric conservation law for high-order time-accurate simulations on dynamic meshes. *J. Comput. Phys.*, 213(2):557–573, 2006.

[55] J. M. Melenk. On condition numbers in hp-FEM with Gauss-Lobatto-based shape functions. *J. Comput. Appl. Math.*, 139(1):21–48, 2002.

[56] M. F. Murphy, G. H. Golub, and A. J. Wathen. A Note on Preconditioning for Indefinite Linear Systems. *SIAM J. Sci. Comput.*, 21(6):1969–1972, 1999.

[57] F. Nobile. *Numerical approximation of fluid-structure interaction problems with application to haemodynamics.* PhD thesis, n°2458, EPF Lausanne, 2001.

[58] E. T. Olsen and J. Jim Douglas. Bounds on spectral condition numbers of matrices arising in the p-version of the finite element method. *Numer. Math.*, 69(3):333–352, 1995.

[59] R. Pasquetti, L. Pavarino, F. Rapetti, and E. Zampieri. *Domain Decomposition Methods in Science and Engineering XVI*, volume 55 of *Lecture Notes in Computational Science and Engineering*, chapter Overlapping Schwarz preconditioners for Fekete spectral elements. Springer Berlin Heidelberg, 2007.

[60] R. Pasquetti and F. Rapetti. Spectral element methods on triangles and quadrilaterals: comparisons and applications. *J. Comput. Phys.*, 198:349–462, 2004.

[61] A. T. Patera. A spectral element methods for fluid dynamics: laminar flow in a channel expansion. *J. Comput. Phys.*, 54:468–488, 1984.

[62] L. F. Pavarino. Indefinite overlapping Schwarz methods for time-dependent Stokes problems. *Comput. Meth. Appl. Mech. Engrg.*, 187(1-2):35–51, 2000.

[63] L. F. Pavarino and A. Klawonn. A comparison of overlapping Schwarz methods and block preconditioners for saddle point problems. *Numer. Linear Algebra Appl.*, 7(1):1–25, 2000.

[64] C. Prud'homme. A domain specific embedded language in C++ for automatic differentiation, projection, integration and variational formulations. *Scientific Programming*, 14(2):81–110, 2006. `http://iospress.metapress.com/link.asp?id=8xwd8r59hg1hmlcl`.

[65] C. Prud'homme. Life: Overview of a unified C++ implementation of the finite and spectral element methods in 1D, 2D and 3D. In *Workshop On State-Of-The-Art In Scientific And Parallel Computing*, Lecture Notes in Computer Science, page 10. Springer-Verlag, 2006. Accepted.

[66] C. Prud'homme. Life: A modern and unified c++ implementation of finite-element and spectral-elements methods in 1d, 2d and 3d: overview and applications. In *ICIAM*, 2007. accepted.

[67] A. Quarteroni and L. Formaggia. Mathematical Modelling and Numerical Simulation of the Cardiovascular System. In *Modelling of Living Systems*, Handbook of Numerical Analysis Series. 2003.

[68] A. Quarteroni, F. Saleri, and A. Veneziani. Analysis of the Yosida Method for the Incompressible Navier-Stokes Equations. *Journal de Mathématiques Pures et Appliquées*, 78(5):473–503, 1999.

[69] A. Quarteroni, F. Saleri, and A. Veneziani. Factorization methods for the numerical approximation of Navier-Stokes equations. *Comput. Methods Appl. Mech. Eng.*, 188(1-3):505–526, 2000.

[70] A. Quarteroni and A. Valli. *Numerical Approximation of Partial Differential Equations.* Springer-Verlag, Berlin, Heidelberg, New York, 1994.

[71] B. Ramaswamy and M. Kawahara. Arbitrary Lagrangian-Eulerian finite element method for unsteady, convective, incompressible viscous free surface fluid flow. *International Journal for Numerical Methods in Fluids*, 7(10):1053–1075, 1987.

[72] E. M. Rønquist. *Optimal spectral element methods for the unsteady three-dimensional incompressible Navier-Stokes equations.* PhD thesis, Massachusetts Institute of Technology, 1988.

[73] M. J. Roth. *Nodal configurations and voronoi tessellations for triangular spectral elements.* PhD thesis, Victoria, B.C., Canada, Canada, 2005.

[74] M. Sala and M. Heroux. Robust algebraic preconditioners with IFPACK 3.0. Technical Report SAND-0662, Sandia National Laboratories, 2005.

[75] F. Saleri and A. Veneziani. Pressure correction algebraic splitting methods for the incompressible Navier–Stokes equations. *SIAM J. Numer. Anal.*, 43(1):174–194, 2005.

[76] C. Schwab and M. Suri. Mixed hp finite element methods for Stokes and non-Newtonian flow. *Comput. Methods Appl. Mech. Engrg.*, 175:217–241, 1999.

[77] S. Sherwin and G. Karniadakis. A triangular spectral element method; applications to the incompressible navier-stokes equations. *Comput. Meth. Appl. Mech. Eng.*, 124:189–229, 1995.

[78] D. Silvester, H. Elman, D. Kay, and A. Wathen. Efficient Preconditioning of the Linearized Navier–Stokes Equations for Incompressible Flow. *J. Comput. Appl. Math.*, 128(1-2):261–279, 2001.

[79] P. Solin, K. Segeth, and I. Dolezel. *Higher-Order Finite Element Methods.* Chapman and Hall/CRC, Boca Raton, London, New York, 2004.

[80] K. Stanley. Klu: a "clark kent" sparse lu factorization algorithm for circuit matrices. In *2004 SIAM Conference on Parallel Processing for Scientific Computing (PP04)*, 2004.

[81] R. Stenberg and M. Suri. Mixed hp finite element methods for problems in elasticity and Stokes flow. *Numer. Math.*, 72(3):367–389, 1996.

[82] V. L. Streeter, E. B. Wylie, and K. B. Bedford. *Fluid mechanics.* McGraw-Hill, 9th ed. edition, 1998.

[83] S. Sy and C. Murea. A stable time advancing scheme for solving fluid-structure interaction problem at small structural displacements. *Computer Methods in Applied Mechanics and Engineering*, August 2008.

[84] G. Szego. Orthogonal polynomials. *Am. Math. Soc.*, 23, 1939.

[85] M. A. Taylor, B. A. Wingate, and R. E. Vincent. An algorithm for computing Fekete points in the triangle. *SIAM J. Numer. Anal.*, 38(5):1707–1720 (electronic), 2000.

[86] R. Temam. *Navier-Stokes equations and nonlinear functional analysis.* Number CBMS-NSF 66. SIAM, Philadelphia, 2nd edition edition, 1995.

[87] L. N. Trefethen and J. A. C. Weideman. Two results on polynomial interpolation in equally spaced points. *J. Approx. Theory*, 65(3):247–260, 1991.

[88] A. Veneziani. *Mathematical and Numerical Modeling of Blood flow problems.* PhD thesis, Università degli Studi di Milano, 1998.

[89] T. Warburton. *Spectral/hp methods on polymorphic multi-domains : algorithms and applications.* PhD thesis, Brown University, RI, 1999.

[90] T. Warburton. An explicit construction for interpolation nodes on the simplex. *Journal of Engineering Mathematics*, 56(3):247–262, November 2006.

[91] T. Warburton, L. F. Pavarino, and J. S. Hesthaven. A Pseudo-spectral Scheme for the Incompressible Navier-Stokes Equations Using Unstructured Nodal Elements. *J. Comput. Phys.*, 164:1–21, 2000.

[92] C. Winkelmann. *Interior penalty finite element approximation of Navier-Stokes equations and application to free surface flows.* PhD thesis, Lausanne, 2007.

Curriculum Vitæ

Gonçalo Pena was born on May 19th, 1980 in Coimbra, Portugal. He attended high school in *Escola Secundária de Arganil* which he finished in 1998 with the final mark of 18 (out of 20). In the same year, he enrolled the University of Coimbra where he got a degree in *Mathematics - Specialization in Pure Mathematics* in 2002, with the final mark of 17 (out of 20). Afterwards, he started the master's program in the Department of Mathematics of the University of Coimbra under the supervision of Professor José Augusto Ferreira. He got his master degree in *Mathematics - Specialization in Applied Mathematics* in March 2005 with the classification of *Muito Bom*. The title of his dissertation was *Splitting methods and Implicit-explicit methods for convection-diffusion-reaction equations*. In October 2005 he enrolled in the Doctoral School of Mathematics in École Polytechnique Fédérale de Lausanne, Switzerland, where he started a PhD under the supervision of Professor Alfio Quarteroni. He now works as a teaching assistant in the Department of Mathematics of the University of Coimbra.

Scholarships and grants:

- Merit scholarship given by *Fundação Calouste Gulbenkian* under the program *Novos Talentos em Matemática*, 2000-2001
- Individual Doctoral grant from *Fundação para a Ciência e Tecnologia* to conduct doctoral studies in École Polytechnique Fédérale de Lausanne, Switzerland, 2005-2009
- Scholarship given by *Fundação Calouste Gulbenkian* under the program *Estímulo à Criatividade e à Qualidade na Actividade de Investigação*, 2009

List of publications:

- C. Prud'homme and G. Pena, *Construction of a High Order Fluid-Structure Interaction Solver*, Journal of Computational and Applied Mathematics, 2009, In Press, Accepted Manuscript

Projects:

- Center of Mathematics of the University of Coimbra (group of Numerical Analysis and Optimization), member, since 2003 to present

- LIFE, a unified C++ implementation of spectral and hp/finite element methods in 1D, 2D and 3D, developer, from January 2006 to present
- Reaction-Diffusion in Porous Media (UTAustin/MAT/0066/2008), researcher, from May 2009 to present

Invited seminars:

- BCAM-CIM, workshop on Applied Mathematics, Bilbao (Spain), 2nd July 2009, *Fluid-structure interaction: a spectral approach*

Oral presentations:

- CMUC internal seminar, Coimbra (Portugal), October 2003, *Some extension to non-linear evolutionary problems of the concept of stability*
- ACOMEN 2008, Liège (Belgium), 27th May 2008, Talk: *Towards the construction of a high order fluid-structure interaction solver*
- CMUC internal seminar, Coimbra (Portugal), 5th February 2009, Talk: *Construction of a high order fluid-structure interaction solver*

Participation in postgraduate courses:

- Course: High Performance Computing Methods, Doctoral course in Mathematics, EPFL, October-December 2006
- Course: Error Estimates for Stabilized Finite Element Methods, Doctoral course in Mathematics, EPFL, October - November 2007
- Course: Spectral Methods, Doctoral course in Mechanics, EPFL, November 2007 - January 2008

Conferences and summer schools:

- Workshop: Modeling and Simulation in Chemical Engineering, University of Coimbra, 30th June 2003
- Summer school: Mathematical models in Life Science: Theory and Simulation, Dobbiaco, 1st - 5th July 2005
- Workshop: Variational Multiscale Method and Stabilized Finite Elements, EPFL, 12-13 February 2007
- Workshop: Numerical Analysis of Stochastic PDEs, Eidgenössische Technische Hochschule, Zürich (ETHZ), 16-17 May 2008