Ph.D. program in Computational Mathematics and Decision Sciences
UNIVERSITÀ DEGLI STUDI DI PAVIA
UNIVERSITÀ DELLA SVIZZERA ITALIANA

Ph.D. program in Applied Mathematics and Computational Sciences
KING ABDULLAH UNIVERSITY OF SCIENCE AND TECHNOLOGY

Numerical approximation of PDEs on complex geometries: fluid–structure interaction problems and virtual element methods

---

Ph.D. dissertation

Advisors:
Silvia Bertoluzza
Daniele Boffi

Candidate:
Fabio Credali

Abstract

The studies in numerical approximation of partial differential equations (PDEs) are characterized by the necessity of managing complex geometries and their discretization. We focus our attention on two different fields where complex geometries are very common: the mathematical modeling of fluid-structure interaction problems and the family of virtual element methods.

We first consider the finite element approximation of fluid-structure interaction (FSI) problems described by a distributed Lagrange multiplier formulation. The method is based on a fixed mesh for the fluid domain which is extended to include the region occupied by the solid, in the spirit of the fictitious domain approach. The reference configuration of the solid is discretized with a fixed mesh which is mapped, at each time step, to the actual position of the body.

A crucial aspect of the method is the assembly of the finite element matrix describing the coupling between fluid and structure. This procedure consists in integrating over the solid domain both solid and fluid basis functions, which are defined on two different non-matching grids. This can be done in two ways: the *exact approach* is based on a composite quadrature rule defined on the intersection between the two involved meshes, whereas the *approximate approach* is carried out roughly integrating on each solid element. We discuss and compare these two approaches both from the theoretical and computational point of view. We present quadrature error estimates for the approximate case accompanied by a wide range of numerical tests, showing the different features of the considered techniques. Moreover, we provide the proof that the problem is well-posed when the coupling term is inexactly assembled.

From the computational point of view, another aspect that cannot be overlooked is the cost, in terms of time and resources, required to carry out the simulation of a fluid-structure interaction problem. In particular, several difficulties

may occur while performing this task. For instance, the problem arises from nonlinear models, the coupling terms need to be updated at each time step, the linear system is ill-conditioned due to the high resolution. Therefore, for these reasons, accuracy and efficiency should be balanced and the use of a parallel solver becomes mandatory in this sense. We present a preliminary study on a parallel solver for our fictitious domain formulation. As a first step towards the design of an effective solver, we consider block preconditioners, which are tested on two simplified academic problems and compared in terms of optimality, weak and strong scalability.

Virtual element methods (VEM) are known to tackle complex geometries without limitations on the degree of the polynomial that partially contributes to the approximated solution. An important aspect of this method is that we are not required to explicitly compute the basis functions of the VEM space since these are solutions of PDEs. Due to this fact, several quantities are not computed exactly. Two of these are the bilinear form, for which the action on the nonpolynomial part is handled by a stabilization term, and the error, where the contribution of the nonpolynomial part is neglected.

In certain cases, such as when anisotropic problems are considered, the method may show poor performance if endowed with standard stabilization terms, which have an isotropic structure. We propose a model order reduction technique constructed by means of the reduced basis method (RB) for efficiently solving the equation associated to each virtual basis function. The idea is to replace the stabilization term with an actual approximation of the nonpolynomial contribution. We show that this operation produces good results even if done in a very rough way, so that the *virtual* nature of the method is preserved.

In post-processing framework, it is well known that, when a PDE is solved with VEM, the degrees of freedom of the discrete solution allow only the computation of projections onto discontinuous polynomial spaces, so that the solution is not conforming. The RB approximation of the virtual functions can also be exploited for reconstructing conforming solution in the VEM space. This task can be useful to carry out operations such as visualization, reconstruction in subdomains, pointwise evaluation and evaluation of the conforming error when benchmarking the method.

The proposed RB approach is validated trough the comparison with standard VEM techniques in terms of accuracy and efficiency and the mentioned applications are discussed with several numerical tests.

Sommario

Gli studi riguardanti l'approssimazione numerica di equazioni differenziali alle derivate parziali (PDEs) sono caratterizzati dalla necessità di gestire geometrie complesse e la loro discretizzazione. Focalizziamo la nostra attenzione su due differenti campi dove la presenza di geometrie complesse è molto comune: lo studio di modelli matematici per problemi di interazione fluido-struttura e la famiglia dei metodi agli elementi virtuali.

Consideriamo l'approssimazione tramite elementi finiti di problemi di interazione fluido-struttura (FSI) descritti tramite una formulazione con moltiplicatore di Lagrange distribuito. Il metodo è basato su una mesh fissa per il dominio fluido che viene estesa affinchè includa anche la regione occupata dal solido, nello spirito dell'approccio a dominio fittizio. La configurazione di riferimento per il solido è discretizzata con una mesh fissa che viene mappata, ad ogni istante di tempo, nella posizione effettiva dell'oggetto.

Un aspetto cruciale del metodo riguarda l'assemblaggio della matrice ad elementi finiti che descrive l'interazione tra il fluido e la struttura. Questa procedura consiste nell'integrare sul dominio solido le funzioni di base sia del fluido che del solido, che sono definite su due mesh non allineate. Questa operazione può essere effettuata in due modi: l'*approccio esatto* è basato sull'uso di una formula di quadratura composita che viene costruita sull'intersezione delle mesh coinvolte, mentre l'*approccio approssimato* consiste nell'intergare grossolanamente su ogni elemento della mesh solida. Discutiamo e confrontiamo questi due approcci sia dal punto di vista teorico che computazionale. Presentiamo stime per l'errore di quadratura nel caso approssimato, corredate da una vasta gamma di test numerici che mostrano le diverse caratteristiche delle tecniche considerate. Inoltre, dimostriamo che il problema discreto è ben posto anche quando il termine di accoppiamento viene costruito in maniera approssimativa.

Dal punto di vista computazionale, un altro aspetto che non può essere ignorato è il costo, in termini di tempo e risorse, necessario per simulare un problema di interazione fluido-struttura. Potrebbero infatti sorgere varie difficoltà. Per esempio, il problema potrebbe derivare da modelli non lineari, il termine di accoppiamento deve essere aggiornato ad ogni istante di tempo, il sistema lineare è mal condizionato a causa dell'alta risoluzione. Di conseguenza, per queste ragioni, accuratezza ed efficienza devono essere bilanciate e l'uso di un solutore parallelo diventa obbligatorio in questo senso. Presentiamo uno studio preliminare riguardante un primo solutore parallelo per la nostra formulazione a dominio fittizio. Come primo passo verso il design di un solutore efficace, consideriamo due precondizionatori a blocchi, che testiamo e confrontiamo su due problemi accademici in termini di ottimalità e scalabilità debole e forte.

I metodi agli elementi virtuali (VEM) sono noti per la capacità di gestire geometrie complesse senza limitazioni sul grado polinomiale, che contribuisce solo parzialmente alla soluzione approssimata. Un aspetto importante di questo metodo consiste nel fatto che non è necessario calcolare esplicitamente le funzioni di base dello spazio VEM, in quanto esse stesse soluzioni di PDEs. Di conseguenza, diverse quantità non sono calcolate in maniera esatta. Due di queste sono la forma bilineare, in cui il contributo non polinomiale viene gestito tramite un termine di stabilizzazione, e l'errore, dove il contributo non polinomiale viene ignorato.

In alcuni casi, come per esempio quando si considerano problemi anisotropi, il metodo potrebbe fornire risultati non ottimali se costruito tramite termini di stabilizzazione standard, che hanno invece struttura isotropa. Proponiamo un approccio ad ordine ridotto costruito tramite il metodo alle basi ridotte (RB) per risolvere in maniera efficiente l'equazione associata ad ogni funzione di base virtuale. L'idea è di sostituire il termine di stabilizzazione con una approssimazione effettiva del contributo non polinomiale. Mostriamo che questa operazione porta a buoni risultati anche se effettuata in maniera grossolana, in modo tale che la natura virtuale del metodo sia preservata.

Nel caso del post-processing, è ben noto che, quando una PDE viene risolta tramite VEM, i gradi di libertà della soluzione discreta permettono di calcolare solo proiezioni in spazi polinomiali discontinui, per cui la soluzione risulta non conforme. L'approssimazione a basi ridotte delle funzioni virtuali può essere sfruttata anche per ricostruire soluzioni che siano conformi nello spazio VEM. Questa

tecnica può essere impiegata per effettuare operazioni come visualizzazione, ricostruzione in sotto-domini, valutazione puntuale e valutazione dell'errore conforme quando si effettua l'analisi comparativa del metodo.

Il metodo a basi ridotte che proponiamo viene validato, in termini di accuratezza ed efficienza, tramite il confronto con tecniche VEM standard e le applicazioni che abbiamo menzionato vengono discusse tramite diversi test numerici.

Ph.D. program in Computational Mathematics and Decision Sciences
Università degli Studi di Pavia, Italy
Università della Svizzera Italiana, Switzerland
Ph.D. program Coordinator: Luca Pavarino

Ph.D. program in Applied Mathematics and Computational Sciences
Computer, Electrical and Mathematical Sciences and Engineering Division,
King Abdullah University of Science and Technology, Saudi Arabia

Reviewers:

Luca Heltai	Scuola Internazionale Superiore di Studi Avanzati, Italy
Stefano Berrone	Politecnico di Torino, Italy

Committee members:

Luca Pavarino	Università degli Studi di Pavia, Italy
Daniele Boffi	King Abdullah University of Science and Technology, Saudi Arabia
Silvia Bertoluzza	IMATI 'E. Magenes', CNR Pavia, Italy
Stefano Berrone	Politecnico di Torino, Italy
Luca Heltai	Scuola Internazionale Superiore di Studi Avanzati, Italy
Matteo Parsani	King Abdullah University of Science and Technology, Saudi Arabia
Rolf Krause	Università della Svizzera Italiana, Switzerland

Pavia, 28 November 2023

Acknowledgements

The work presented in this Ph.D. thesis has been developed during a three years journey between Europe and Middle East: I thank the three universities that made this unique experience possible.

I would like to thank my mentors, Silvia Bertoluzza and Daniele Boffi, for their availability, kindness, and enthusiasm in guiding my work. I couldn't have asked for better role models.

Many thanks to Lucia Gastaldi, Simone Scacchi and Daniele Prada for the collaboration we established during these years, which not only contributed to the scientific results, but also to my academic growth.

Daniele B. deserves also my gratitude for giving me the opportunity of living and studying in Saudi Arabia, taking me out of the "tiny world" of Ferrera and Pavia. Life in KAUST would not have been the same without my traveling companions: Simone, Clarissa and Stefanos ("the committee"), Linda and Najwa, Luca and Roberto, Paolo, Umberto. I will miss the moments of brain storming and relax in the office, the dinners together, the time we spent in IRC. I will never forget our trips to Al Ula and Egypt, the conferences. I hope our friendship remains, even though our paths are somehow diverging.

Moving to Italy, I can't thank my Family enough for the support they gave me throughout the eight years of university studies (and more than that ...). They are always there, at home, waiting for me. Thanks to dad Marco and uncle Franco for the home-airport taxi service, even at unlikely hours.

The final thought goes to my friends. To lifelong friends, partner in crime in a thousand adventures: Elena, Jacopo, Riccardo, Luca, and Cesare. To Matteo, Mario, and Alessandro, with whom I began the mathematical journey back in 2015: we have stayed in touch ever since. To the friends of the legendary "Cardano": Laura, Alessandro, and Marco. And lastly, to Sofia and Donato, for the time shared at the "Nave".

Ringraziamenti

Il lavoro contenuto in questa tesi di dottorato è stato sviluppato durante tre anni di viaggio tra l'Europa e il Medio Oriente: ringrazio le tre istituzioni universitarie che mi hanno permesso di vivere questa esperienza unica.

Ringrazio i miei mentori, Silvia Bertoluzza e Daniele Boffi, per la disponibilità, la gentilezza e l'entusiasmo con cui hanno sempre guidato il mio lavoro. Non avrei potuto seguire esempi migliori.

Un ringraziamento va anche a Lucia Gastaldi, Simone Scacchi e Daniele Prada per la bella collaborazione che ha contribuito sia ai risultati scientifici che alla mia formazione accademica.

A Daniele B. va anche la mia gratitudine per avermi dato l'opportunità di vivere e studiare in Arabia Saudita, tirandomi fuori dal "mondo piccolo" di Ferrera e Pavia. Sicuramente la vita a KAUST non sarebbe stata la stessa senza i miei compagni di viaggio: Simone, Clarissa e Stefanos ("the committee"), Linda e Najwa, Luca e Roberto, Paolo, Umberto. Mi mancheranno i momenti di confronto e svago in ufficio, le cene insieme, le giornate in piscina. Non dimenticherò mai i nostri viaggi ad Al Ula e in Egitto, le conferenze. Spero che la nostra amicizia rimanga nonostante le nostre strade si stiano in qualche modo dividendo.

Spostandomi in Italia, non posso che ringraziare infinitamente la mia Famiglia, sempre lì a casa ad aspettarmi, per il sostegno durante gli otto anni di studi universitari (e non solo ...). Un grazie a papà Marco e zio Franco per il servizio taxi casa-aeroporto, anche ad orari improbabili.

L'ultimo pensiero è per gli amici. Per gli amici di una vita, complici di mille avventure: Elena, Jacopo, Riccardo, Luca e Cesare. Per Matteo, Mario ed Alessandro, coi quali ho iniziato il percorso matematico nel lontano 2015: da allora non ci siamo persi di vista. Per i compagni del mitico "Cardano", Laura, Alessandro e Marco. E infine per Sofia e l'ingegner Donato, per il tempo condiviso in "Nave".

Contents

Functional analysis notation

In this section, we recall classical notation in functional analysis we are going to adopt through the work. Let us denote by $\Omega \subset \mathbb{R}^d$ a generic open bounded domain.

We first introduce the space of square integrable functions in Ω

$$\mathrm{L}^2(\Omega) = \left\{ v : \int_\Omega |v|^2 \, \mathrm{d}\mathbf{x} = \|v\|^2_{\mathrm{L}^2(\Omega)} < \infty \right\}.$$

In particular, the norm of this space is induced by the scalar product

$$(v, w)_\Omega = \int_\Omega vw \, \mathrm{d}\mathbf{x}.$$

For the sake of precision, notice that the term *functions* should be replaced by *classes of measurable functions*, where each class is composed by functions which differ only on a subset of Ω with zero Lebesgue measure. Moreover, we consider the following subspace made up of all the $\mathrm{L}^2(\Omega)$ functions with zero mean

$$\mathrm{L}^2_0(\Omega) = \left\{ v \in \mathrm{L}^2(\Omega) : \int_\Omega v \, \mathrm{d}\mathbf{x} = 0 \right\}.$$

We now introduce the concept of Sobolev space. Given an integer $r \geq 0$, we define

$$\mathrm{H}^r(\Omega) = \left\{ v \in \mathrm{L}^2(\Omega) : \, D^{\boldsymbol{\alpha}} v \in \mathrm{L}^2(\Omega), \quad \forall |\boldsymbol{\alpha}| \leq r \right\}.$$

In particular, $\boldsymbol{\alpha}$ is a multi-index, which means $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_d)$ and $|\boldsymbol{\alpha}| = \alpha_1 + \cdots + \alpha_d$. If $\mathbf{x} \in \mathbb{R}^d$, then the action of $\boldsymbol{\alpha}$ consists in

$$\mathbf{x}^{\boldsymbol{\alpha}} = x_1^{\alpha_1} \cdots x_d^{\alpha_d}.$$

The symbol $D^{\boldsymbol{\alpha}} v$ denotes the following derivative, which is intended in distributional sense

$$D^{\boldsymbol{\alpha}} v = \frac{\partial^{|\boldsymbol{\alpha}|} v}{\partial x_1^{\alpha_1} \cdots \partial x_d^{\alpha_d}}.$$

The Sobolev space $\mathrm{H}^r(\Omega)$ is endowed with the norm

$$\|v\|_{r,\Omega}^2 = \sum_{s \le r} |v|_{s,\Omega}^2,$$

where $|v|_{s,\Omega}$ denotes the s^{th} semi-norm

$$|v|_{s,\Omega}^2 = \sum_{|\boldsymbol{\alpha}|=s} |D^{\boldsymbol{\alpha}} v|_{\mathrm{L}^2(\Omega)}^2 .$$

Notice that $\mathrm{L}^2(\Omega)$ can be interpreted as the Sobolev space $\mathrm{H}^0(\Omega)$. Therefore, we set $\|v\|_{0,\Omega} = \|v\|_{\mathrm{L}^2(\Omega)}$. In particular, $\mathrm{H}^1(\Omega)$ is the space of square integrable functions up to the first order derivative; $\mathrm{H}_0^1(\Omega) \subset H^1(\Omega)$ is the subspace of functions with zero trace on $\partial\Omega$. In this subspace, the semi-norm $|v|_{1,\Omega}$ is equivalent to the full norm $\|v\|_{1,\Omega}$.

If the exponent r is any positive real number, we define the fractional Sobolev space

$$\mathrm{H}^r(\Omega) = \left\{ v \in \mathrm{L}^2(\Omega) : \frac{|v(\mathbf{x}_1) - v(\mathbf{x}_2)|}{|\mathbf{x}_1 - \mathbf{x}_2|^{\frac{d}{2}+r}} \in \mathrm{L}^2(\Omega \times \Omega) \right\}$$

endowed with the natural norm

$$\|v\|_{r,\Omega}^2 = \|v\|_{0,\Omega}^2 + \int_\Omega \int_\Omega \frac{|v(\mathbf{x}_1) - v(\mathbf{x}_2)|^2}{|\mathbf{x}_1 - \mathbf{x}_2|^{d+2r}} \mathrm{d}\mathbf{x}_1 \mathrm{d}\mathbf{x}_2,$$

where the term

$$|v|_{r,\Omega}^2 = \int_\Omega \int_\Omega \frac{|v(\mathbf{x}_1) - v(\mathbf{x}_2)|^2}{|\mathbf{x}_1 - \mathbf{x}_2|^{d+2r}} \mathrm{d}\mathbf{x}_1 \mathrm{d}\mathbf{x}_2$$

is called Gagliardo semi-norm of v.

The Lebesgue space containing all the measurable functions that are bounded almost everywhere in Ω is

$$\mathrm{L}^\infty(\Omega) = \left\{ v : \|v\|_{\infty,\Omega} = \operatorname{ess\,sup} |v| < \infty \right\}.$$

As done for $\mathrm{L}^2(\Omega)$, also in this case we can introduce a family of Sobolev spaces where derivatives are bounded as well

$$\mathrm{W}^{r,\infty} = \{ v \in \mathrm{L}^\infty(\Omega) : D^{\boldsymbol{\alpha}} v \in \mathrm{L}^\infty(\Omega), \quad \forall |\boldsymbol{\alpha}| \le r \}.$$

In general, given two functional spaces V and W, we denote by $\mathcal{L}(V, W)$ the space of linear functionals from V to W. Moreover, given the dual space V' of V, we denote the duality pairing by angle brackets $\langle \cdot, \cdot \rangle$.

Functional spaces of vector valued functions are indicated with boldface letters.

We denote by $C^0(\Omega)$ the space of continuous functions and by $C^1(\Omega)$ the space of continuous functions with continuous derivative.

Finally, given an integer $k \geq 0$, we introduce the space of polynomials of degree less or equal than k

$$\mathcal{P}_k(\Omega) = \{p : \ p \text{ is a polynomial of degree} \leq k\} .$$

References

I. J. L. Lions, and E. Magenes. *Non-homogeneous boundary value problems and applications: Vol. 1*, volume 181. Springer Science & Business Media, 2012.

II. H. Brézis. *Functional analysis, Sobolev spaces and partial differential equations*, volume 2. Springer, 2011.

III. D. Boffi, F. Brezzi, and M. Fortin. *Mixed finite element methods and applications*, volume 44. Springer, 2013.

Part I

Numerical approximation of fluid-structure interaction problems: a fictitious domain approach

Introduction

The study of fluid-structure interaction problems has been developed starting from the second half of the last century: this topic embraces several fields of study since mathematical modeling, physical phenomena, engineering interests are combined together.

In particular, the equations governing fluids and solids are complicated, since nonlinear terms are usually involved, with the difficulty of knowing analytical solutions. Due to these problems and thanks to the increase of computational power, nowadays we are able to simulate the dynamics of interaction systems.

For this reason, several mathematical and computational approaches have been presented during the years. They can be classified into two big categories [69]: the *monolithic approach* consists in modeling both fluid and structure behaviors in the same mathematical framework in order to obtain a single equation for the whole system, also containing the interface conditions; on the other hand, the *partitioned approach* manages the fluid and the structure as two separate entities, each in its own mathematical setting and with separate meshes; in this case, the interface conditions are imposed explicitly.

These methods can be also classified with respect to the approach used to represent the immersion of the solid into the fluid or viceversa. In this case, the methods are divided into *boundary fitted approaches (BF)* and *non boundary fitted approaches (NBF)*. In the first case, the mesh of the fluid domain evolves and deforms around a Lagrangian mesh used for the structure: the two entities share the interface. A popular scheme of this category is the so-called Arbitrary Lagrangian Eulerian formulation (ALE [67, 48, 70, 49]): the Eulerian point of view used for the fluid is combined with the Lagrangian one, used for the solid, on a boundary fitted mesh; the kinematic constraints are satisfied by construction, but during the evolution of the system, the mesh may become severely distorted.

To overcome the drawbacks just described, NBF approaches were developed:

the solid discretization is superimposed on the fluid one; in this case, the disadvantage may be a reduction in accuracy near the interface. Since there is not an optimal method for the wide range of phenomena one can model, this family includes several methods.

For example, the Nitsche–XFEM [35, 1] method was born for the study of thin-walled elastic bodies immersed in incompressible fluids; the fluid domain discretization is done with an unstructured mesh not fitted to the solid mid-surface deformed mesh. In this way, weak and strong discontinuities are allowed across the interface and the fluid-structure coupling is enforced with Nitsche's mortaring.

A level set formulation [37] was introduced for modeling incompressible and immiscible Navier–Stokes equations separated by a free surface. In particular, the boundary of the two-fluid interface is treated as the set of zeros of a certain smooth function defined on the entire physical domain, so that complex geometries can be easily managed and the extension to the 3D case can be done in a natural way.

Finally, the immersogeometric analysis [72] is a NBF method based on the isogeometric analysis: indeed, NURBS and splines are used to accurately represent the involved geometries.

Our formulation originated from the immersed boundary method (IBM) [86] and then evolved towards a fictitious domain formulation [61, 60]. In the original finite element formulation of the IBM the evolution of the structure is governed by an ordinary differential equation involving the velocity of the fluid and the position of the solid body with the coupling modeled by Dirac delta functions. Then, in the spirit of the fictitious domain, a Lagrange multiplier has been introduced so that the motion of the structure can be variationally imposed through a bilinear form.

This part is divided into three chapters: in the first one, we derive the fictitious domain formulation starting from the immersed boundary method, discussing well-posedness and other properties of continuous and discrete problems. In the second chapter, following [20], we present two possible techniques for the assembly of the finite element interface matrix, which couples the fluid and structure evolution. Indeed, the coupling term can be assembled either implementing an exact composite quadrature rule or with inexact computations. The optimality of the results is analyzed with several numerical tests and quadrature error estimates are presented for the inexact technique. Finally, in Chapter 3, we present a preliminary study on a parallel solver for the efficient simulation of FSI problems with fictitious domain formulation [21, 22].

Chapter 1

The finite element immersed boundary method with distributed Lagrange multiplier

The immersed boundary method was introduced by Peskin in 1972 [85] in order to simulate the dynamics of the flexible leaflet of a human heart valve that moves within the blood flow. This first study was performed in a two dimensional setting and then extended to the three dimensional case in 1989 [87, 82]. During the past years, this method has been implemented for several applications. We mention, for instance, models for flapping flexible filaments in flowing soap films [97], simulation of aquatic animal locomotion [55], computational models of the cochlea [14], large-eddy simulations [88], shock/obstacle interactions [38] and numerical simulations of three dimensional foam [78]. From the mathematical point of view, several aspects have been investigated: for instance, adaptive mesh refinement [89], reduced numerical viscosity [75], a stochastic approach for fluid-structure dynamics at microscopic length scales [6], a penalty method for elastic boundaries with mass [73], and the recent Fourier spectral approach [40]. A review article was published by Peskin in 2002 [86].

The mathematical formulation is based on the use of Eulerian variables for describing the fluid dynamics, while deformations and motion of the immersed solid body are tracked using Lagrangian variables. Moreover, the forces exerted by the structure on the fluid are represented via the use of Dirac delta functions. From the numerical point of view, the method was presented in the finite difference

framework.

In 2003, Boffi and Gastaldi [23] presented a first modification of the immersed boundary method by means of the finite element method, hence they developed the variational formulation of the considered problem and then presented a first proof of existence of the solution in a simple one dimensional problem. During the years, this finite element version has been studied from several points of view: investigations about the numerical stability and CFL condition were published in 2007 [27, 28] in collaboration with Heltai, an hyper-elastic formulation was developed in 2008 [29] with the contribution of Peskin, while the simulation of systems with different fluid and solid densities was presented in 2011 [18]. The variational implementation of immersed finite element methods, such as the IBM, has been discussed in 2012 by Heltai and Costanzo [66]. By the same authors, we also mention [90], where the fully variational method is validated by means of various benchmarks for fluid-structure interaction numerical schemes, and [91], where the method is applied to brain Biomechanics problems.

Finally, in 2015, Boffi, Cavallini and Gastaldi [19] presented a new modification of the finite element immersed boundary method. This is based on the use of a distributed Lagrange multiplier in order to weakly couple the evolution of fluid and structure and enforce the ordinary differential equation governing the motion of the solid. Moreover, it was proved that, when semi-implicit time advancing schemes are chosen, the method is unconditionally stable, i.e. stable without any restriction on the time marching step. This new formulation is a current research topic and several aspects have been studied: in 2017, the inf-sup conditions for both continuous and discrete problems have been proved for the stationary problem originating from the sequence of time semi-discretized problems [24]; in 2019, application and stability of several time advancing schemes like Backward Euler, BDFs, Crank–Nicolson were studied [31]; in 2020 Boffi and Gastaldi investigated about existence and uniqueness of the solution for a simplified linearized version of a FSI problem [25].

The outline of this chapter is the following: in Section 1.1, we present the theoretical background of the immersed boundary method in the case of hyper-elastic structures immersed in incompressible fluids governed by the Navier–Stokes equations. In Section 1.2 we present the new version of the method based on a distributed Lagrange multiplier in the spirit of the fictitious domain approach, while Section 1.3 and Section 1.4 are focused on time and finite element discretizations

respectively. Finally, in Section 1.5, we recall the main existing results on stability and well-posedness.

1.1 The immersed boundary method

1.1.1 Problem setting

We consider fluid-structure interaction problems characterized by an incompressible elastic body immersed in an incompressible fluid, in two or three dimensions ($d = 2, 3$). At the generic time instant t, these two entities occupy two disjoint regions: we call $\Omega_t^f \subset \mathbb{R}^d$ the region occupied by the fluid and Ω_t^s the region occupied by the solid. The solid body can be either of codimension one (*thin* structure) or of codimension zero (*thick* structure). In this work, we focus our attention on the codimension zero case, but the model can deal with more general situations as well. We introduce a third domain, $\Omega \subset \mathbb{R}^d$, as the union of Ω_t^f and Ω_t^s: this domain is independent from time. Moreover, we assume that the solid body cannot touch the boundary of the container Ω.

In order to represent the motion of the solid, we introduce a reference domain $\mathcal{B} \subset \mathbb{R}^d$ associated with the Lagrangian variable $\mathbf{s} \in \mathcal{B}$, while the fluid motion is represented by the Eulerian variable $\mathbf{x}$. In this setting, at each time instant, Ω_t^s can be seen as the image of $\mathcal{B}$ through the map $\mathbf{X} : \mathcal{B} \longrightarrow \Omega_t^s$ and each $\mathbf{x} \in \Omega_t^s$ can be represented in terms of $\mathbf{s}$

$$\mathbf{x} = \mathbf{X}(\mathbf{s}, t). \tag{1.1}$$

A schematic representation is depicted in Figure 1.1.

In particular, the kinematic equation describing the motion of the structure is given by

$$\mathbf{u}^s(\mathbf{x}, t) = \frac{\partial \mathbf{X}}{\partial t}(\mathbf{s}, t) \quad \text{for } \mathbf{x} = \mathbf{X}(\mathbf{s}, t) \tag{1.2}$$

where $\mathbf{u}^s$ denotes the velocity of the solid. Conversely, if we work with the derivatives with respect to $\mathbf{s}$, we get the deformation gradient $\mathbb{F} = \boldsymbol{\nabla}_{\mathrm{s}} \mathbf{X}$ with its determinant $|\mathbb{F}|$. Since we consider the case of incompressible solid materials, $|\mathbb{F}|$ is constant in time, and, in addition, $|\mathbb{F}| = 1$ when $\mathcal{B}$ corresponds with the initial position Ω_0^s of the structure. On the other hand, the fluid evolution is studied considering velocity $\mathbf{u}^f$ and pressure p^f. Denoting by ρ_f and $\nu_f > 0$ the fluid density and viscosity respectively and introducing the symmetric gradient $\underline{\boldsymbol{\varepsilon}}(\mathbf{v}) = (\nabla \mathbf{v} + \nabla^\top \mathbf{v})/2$, we

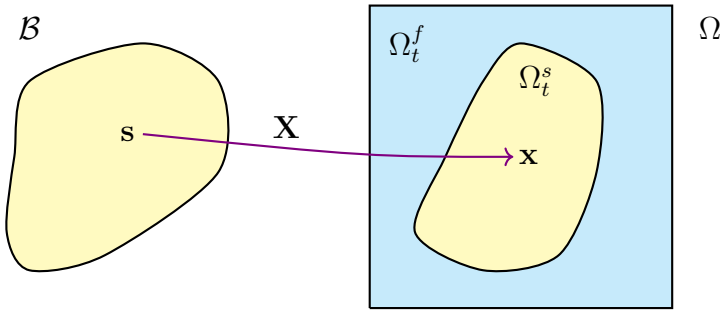


Figure 1.1: Geometric configuration of the problem. The Lagrangian points $\mathbf{s}$ in the solid reference domain $\mathcal{B}$ are mapped into the actual position of the body Ω_t^s through the map $\mathbf{X}$. The domain Ω acts like a container.

can define the Cauchy stress tensor

$$\boldsymbol{\sigma}^f = -p^f \mathbb{I} + \nu_f \, \underline{\boldsymbol{\varepsilon}}(\mathbf{u}^f) \tag{1.3}$$

so that we can state the incompressible Navier–Stokes equations describing the fluid dynamics in Ω_t^f as follows

$$\begin{aligned} &\rho_f \left(\frac{\partial \mathbf{u}^f}{\partial t} + \mathbf{u}^f \cdot \nabla \, \mathbf{u}^f \right) = \operatorname{div} \boldsymbol{\sigma}^f \\ &\operatorname{div} \mathbf{u}^f = 0. \end{aligned} \tag{1.4}$$

For the solid, we consider the case of a viscoelastic material, therefore the characterization can be done splitting the Cauchy stress tensor into two contributions

$$\boldsymbol{\sigma}^s = \boldsymbol{\sigma}_f^s + \boldsymbol{\sigma}_s^s \tag{1.5}$$

where $\boldsymbol{\sigma}_f^s$ is similar to the fluid tensor $\boldsymbol{\sigma}^f$, but defined via the solid viscosity $\nu_s > 0$ and the pressure p^s, which is a Lagrange multiplier used to enforce the incompressibility condition

$$\boldsymbol{\sigma}_f^s = -p^s \mathbb{I} + \nu_s \, \underline{\boldsymbol{\varepsilon}}(\mathbf{u}^s), \tag{1.6}$$

while $\boldsymbol{\sigma}_s^s$ is defined using the Piola–Kirchhoff elasticity stress tensor $\mathbb{P}$

$$\boldsymbol{\sigma}_s^s = |\mathbb{F}|^{-1} \, \mathbb{P}\mathbb{F}. \tag{1.7}$$

In particular, in the Lagrangian setting, we are expressing the elastic contribution to the Cauchy stress tensor using $\mathbb{P}$, indeed $\mathbb{P}(\mathbf{s}, t) = |\mathbb{F}(\mathbf{s}, t)| \, \boldsymbol{\sigma}_s^s(\mathbf{x}, t) \mathbb{F}^{-\top}(\mathbf{s}, t)$

for $\mathbf{x} = \mathbf{X}(\mathbf{s}, t)$. This represents the elastic force per unit reference volume or area in the physical space (pointwise expression). It is important to notice that if $\mathcal{O}$ is a smooth portion of the reference domain $\mathcal{B}$ evolving as $\mathcal{O}_t = \mathbf{X}(\mathcal{O}, t)$, then we have

$$\int_{\partial\mathcal{O}_t} \boldsymbol{\sigma}_s^s \cdot \mathbf{n}\, \mathrm{da} = \int_{\partial\mathcal{O}} \mathbb{P} \cdot \mathbf{N}\, \mathrm{dA} \quad \forall \mathcal{O} \subset \mathcal{B} \tag{1.8}$$

where $\mathbf{n}$ is the outer normal to $\mathcal{O}_t$ and $\mathbf{N}$ is the outer normal to $\mathcal{O}$ in Lagrangian coordinates.

We now have all the ingredients we need to state the equations for the solid: denoting by ρ_s the solid density, we have

$$\begin{aligned} \rho_s \frac{\partial^2 \mathbf{X}}{\partial t^2} &= \mathrm{div}_{\mathbf{s}} \left(|\mathbb{F}|\, \boldsymbol{\sigma}_f^s \mathbb{F}^{-\top} + \mathbb{P}(\mathbb{F}) \right) \\ \mathrm{div}\, \mathbf{u}^s &= 0. \end{aligned} \tag{1.9}$$

Finally, we enforce continuity for velocity and Cauchy stress tensor by imposing the following transmission conditions

$$\begin{aligned} &\mathbf{u}^f = \mathbf{u}^s && \text{on } \partial\Omega_t^s \\ &\boldsymbol{\sigma}_f \mathbf{n}_f = -\left(\boldsymbol{\sigma}_f^s + |\mathbb{F}|^{-1}\, \mathbb{P}\mathbb{F}^\top\right)\mathbf{n}_s && \text{on } \partial\Omega_t^s, \end{aligned} \tag{1.10}$$

where $\mathbf{n}_s$ and $\mathbf{n}_f$ denote the outer unit normal vectors to Ω_t^s and Ω_t^f, respectively. Moreover, also the following initial conditions are considered

$$\begin{aligned} &\mathbf{u}^f(0) = \mathbf{u}_0^f && \text{in } \Omega_0^f \\ &\mathbf{u}^s(0) = \mathbf{u}_0^s && \text{in } \Omega_0^s \\ &\mathbf{X}(0) = \mathbf{X}_0 && \text{in } \mathcal{B} \\ &\mathbf{u}^f = 0 && \text{on } \partial\Omega. \end{aligned} \tag{1.11}$$

1.1.2 Derivation of the model

To derive our model, we now extend to the entire domain Ω all the involved quantities, therefore we have

$$\mathbf{u} = \begin{cases} \mathbf{u}^f \text{ in } \Omega_t^f \\ \mathbf{u}^s \text{ in } \Omega_t^s \end{cases} \qquad p = \begin{cases} p^f \text{ in } \Omega_t^f \\ p^s \text{ in } \Omega_t^s \end{cases} \tag{1.12}$$

$$\rho = \begin{cases} \rho^f \text{ in } \Omega_t^f \\ \rho^s \text{ in } \Omega_t^s \end{cases} \qquad \nu = \begin{cases} \nu^f \text{ in } \Omega_t^f \\ \nu^s \text{ in } \Omega_t^s \end{cases} \qquad \boldsymbol{\sigma} = \begin{cases} \boldsymbol{\sigma}^f \text{ in } \Omega_t^f \\ \boldsymbol{\sigma}^s \text{ in } \Omega_t^s \end{cases} \tag{1.13}$$

so that we can introduce the mass balance equation

$$\frac{\partial \rho}{\partial t} + \rho \operatorname{div} \mathbf{u} = 0 \tag{1.14}$$

and the conservation of momenta

$$\rho \dot{\mathbf{u}} = \rho \left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} \right) = \operatorname{div} \boldsymbol{\sigma}. \tag{1.15}$$

We also notice that the kinematic equation (1.2) becomes

$$\mathbf{u}(\mathbf{X}(\mathbf{s}, t), t) = \frac{\partial \mathbf{X}}{\partial t}(\mathbf{s}, t) \quad \text{for } \mathbf{s} \in \mathcal{B}, \tag{1.16}$$

which is the constraint taking into account the correspondence between the material velocity of the solid and the velocity in Ω_t^s.

Starting from this last equation and using the principle of virtual works, we can write

$$\int_\Omega \rho \dot{\mathbf{u}} \cdot \mathbf{v} \, \mathrm{d}\mathbf{x} - \int_\Omega (\operatorname{div} \boldsymbol{\sigma}) \mathbf{v} \, \mathrm{d}\mathbf{x} = 0 \tag{1.17}$$

where $\mathbf{v}$ is an arbitrary test function, so that, integrating by parts, we get

$$\int_\Omega \rho \dot{\mathbf{u}} \cdot \mathbf{v} \, \mathrm{d}\mathbf{x} + \int_\Omega \boldsymbol{\sigma} : \nabla \mathbf{v} \, \mathrm{d}\mathbf{x} - \int_{\partial\Omega} (\boldsymbol{\sigma} \cdot \mathbf{n}) \mathbf{v} \, \mathrm{da} = 0 \tag{1.18}$$

denoting by $\mathbf{n}$ the outer normal to $\partial\Omega$.

We rewrite the conservation of momenta in terms of $\rho_f, \rho_s, \boldsymbol{\sigma}_f$ and $\boldsymbol{\sigma}_s$:

$$\begin{aligned} \int_\Omega \rho_f \dot{\mathbf{u}} \cdot \mathbf{v} \, \mathrm{d}\mathbf{x} + \int_\Omega \boldsymbol{\sigma}_f : \nabla \mathbf{v} \, \mathrm{d}\mathbf{x} - \int_{\partial\Omega} \boldsymbol{\sigma}_f \mathbf{n} \cdot \mathbf{v} \, \mathrm{da} \\ = -(\rho_s - \rho_f) \int_{\Omega_t^s} \dot{\mathbf{u}} \cdot \mathbf{v} \, \mathrm{d}\mathbf{x} - \int_{\Omega_t^s} \boldsymbol{\sigma}_s : \nabla \mathbf{v} \, \mathrm{d}\mathbf{x}. \end{aligned} \tag{1.19}$$

In the following, the difference $\rho_s - \rho_f$ will be denoted by $\delta\rho$. Thanks to the definition of $\mathbf{u}^s$ as $\partial \mathbf{X}/\partial t$, we can observe that $\dot{\mathbf{u}} = \partial^2 \mathbf{X}/\partial t^2$ in Ω_t^s; using that $|\mathbb{F}| = 1$ when $\mathcal{B} = \Omega_0^s$, we can write

$$\begin{aligned} \int_\Omega \rho_f \dot{\mathbf{u}} \cdot \mathbf{v} \, \mathrm{d}\mathbf{x} + \int_\Omega \boldsymbol{\sigma}_f : \nabla \mathbf{v} \, \mathrm{d}\mathbf{x} - \int_{\partial\Omega} \boldsymbol{\sigma}_f \mathbf{n} \cdot \mathbf{v} \, \mathrm{da} \\ = -\delta\rho \int_{\mathcal{B}} \frac{\partial^2 \mathbf{X}}{\partial t^2} \cdot \mathbf{v}(\mathbf{X}(\mathbf{s}, t)) \, \mathrm{d}\mathbf{s} - \int_{\mathcal{B}} \mathbb{P} : \boldsymbol{\nabla}_{\mathbf{s}} \mathbf{v}(\mathbf{X}(\mathbf{s}, t)) \, \mathrm{d}\mathbf{s}. \end{aligned} \tag{1.20}$$

By applying integration by parts for the second, the third and the last term as follows

$$\begin{aligned} -\int_\Omega (\operatorname{div}\boldsymbol{\sigma}_f)\mathbf{v}\,\mathrm{d}\mathbf{x} &= \int_\Omega \boldsymbol{\sigma}_f : \nabla\,\mathbf{v}\,\mathrm{d}\mathbf{x} - \int_{\partial\Omega} \boldsymbol{\sigma}_f\mathbf{n}\cdot\mathbf{v}\,\mathrm{da} \\ \int_\mathcal{B} \mathbb{P} : \boldsymbol{\nabla}_\mathrm{s}\,\mathbf{v}(\mathbf{X}(\mathbf{s},t))\,\mathrm{d}\mathbf{s} &= -\int_\mathcal{B} (\operatorname{div}_s \mathbb{P})\mathbf{v}\,\mathrm{d}\mathbf{s} + \int_{\partial\mathcal{B}} \mathbb{P}\mathbf{N}\cdot\mathbf{v}\,\mathrm{dA} \end{aligned} \tag{1.21}$$

we obtain

$$\begin{aligned} \int_\Omega (\rho_f\dot{\mathbf{u}} - \operatorname{div}\boldsymbol{\sigma}_f)\cdot\mathbf{v}\,\mathrm{d}\mathbf{x} = &-\delta\rho\int_\mathcal{B}\frac{\partial^2\mathbf{X}}{\partial t^2}\cdot\mathbf{v}(\mathbf{X}(\mathbf{s},t))\,\mathrm{d}\mathbf{s} \\ &+\int_\mathcal{B}\mathbb{P} : \boldsymbol{\nabla}_\mathrm{s}\,\mathbf{v}(\mathbf{X}(\mathbf{s},t))\,\mathrm{d}\mathbf{s} - \int_{\partial\mathcal{B}}\mathbb{P}\mathbf{N}\cdot\mathbf{v}\,\mathrm{dA}. \end{aligned} \tag{1.22}$$

Introducing the d-dimensional Dirac delta distribution and using the fact that a function can be evaluated multiplying it by a shifted delta and integrating, we can write

$$\mathbf{v}(\mathbf{X}(\mathbf{s},t)) = \int_\Omega \mathbf{v}(\mathbf{x})\boldsymbol{\delta}(\mathbf{x}-\mathbf{X}(\mathbf{s},t))\,\mathrm{d}\mathbf{x} \quad \forall\mathbf{s}\in\mathcal{B} \tag{1.23}$$

hence, all the terms at the right hand side of (1.22) can be rewritten accordingly. We show the procedure only for the first one, since the others are analogous:

$$\begin{aligned} \delta\rho\int_\mathcal{B}\frac{\partial^2\mathbf{X}}{\partial t^2}\cdot\mathbf{v}(\mathbf{X}(\mathbf{s},t))\,\mathrm{d}\mathbf{s} &= \delta\rho\int_\mathcal{B}\frac{\partial^2\mathbf{X}}{\partial t^2}\left[\int_\Omega \mathbf{v}(\mathbf{x})\boldsymbol{\delta}(\mathbf{x}-\mathbf{X}(\mathbf{s},t))\,\mathrm{d}\mathbf{x}\right]\mathrm{d}\mathbf{s} \\ &= \int_\Omega\left[\delta\rho\int_\mathcal{B}\frac{\partial^2\mathbf{X}}{\partial t^2}\cdot\boldsymbol{\delta}(\mathbf{x}-\mathbf{X}(\mathbf{s},t))\,\mathrm{d}\mathbf{s}\right]\cdot\mathbf{v}(\mathbf{X}(\mathbf{s},t))\,\mathrm{d}\mathbf{x}; \end{aligned} \tag{1.24}$$

therefore, equation (1.22) becomes

$$\begin{aligned} \int_\Omega (\rho_f\dot{\mathbf{u}} - \operatorname{div}\boldsymbol{\sigma}_f)\cdot\mathbf{v}\,\mathrm{d}\mathbf{x} = &-\int_\Omega\left[\delta\rho\int_\mathcal{B}\frac{\partial^2\mathbf{X}}{\partial t^2}\cdot\boldsymbol{\delta}(\mathbf{x}-\mathbf{X}(\mathbf{s},t))\,\mathrm{d}\mathbf{s}\right]\cdot\mathbf{v}(\mathbf{X}(\mathbf{s},t))\,\mathrm{d}\mathbf{x} \\ &+\int_\Omega\left[\int_\mathcal{B}(\operatorname{div}_\mathbf{s}\mathbb{P})\boldsymbol{\delta}(\mathbf{x}-\mathbf{X}(\mathbf{s},t))\,\mathrm{d}\mathbf{s}\right]\cdot\mathbf{v}(\mathbf{X}(\mathbf{s},t))\,\mathrm{d}\mathbf{x} \\ &-\int_\Omega\left[\int_{\partial\mathcal{B}}\mathbb{P}\mathbf{N}\boldsymbol{\delta}(\mathbf{x}-\mathbf{X}(\mathbf{s},t))\,\mathrm{dA}\right]\cdot\mathbf{v}(\mathbf{X}(\mathbf{s},t))\,\mathrm{d}\mathbf{x}. \end{aligned}$$

At this point, using the fundamental lemma of calculus of variations, we can delete the integrals related to Ω and $\mathbf{v}$ so that we get the equations governing the interaction between fluid and structure in the setting of the immersed boundary method

$$\begin{aligned}\rho_f\dot{\mathbf{u}} - \operatorname{div}\boldsymbol{\sigma}_f = &-\delta\rho\int_{\mathcal{B}}\frac{\partial^2\mathbf{X}}{\partial t^2}\cdot\boldsymbol{\delta}(\mathbf{x}-\mathbf{X}(\mathbf{s},t))\,\mathrm{d}\mathbf{s}\\ &+\int_{\mathcal{B}}(\operatorname{div}_{\mathbf{s}}\mathbb{P})\boldsymbol{\delta}(\mathbf{x}-\mathbf{X}(\mathbf{s},t))\,\mathrm{d}\mathbf{s}-\int_{\partial\mathcal{B}}\mathbb{P}\mathbf{N}\boldsymbol{\delta}(\mathbf{x}-\mathbf{X}(\mathbf{s},t))\,\mathrm{dA}.\end{aligned} \tag{1.25}$$

We define

$$\begin{aligned}\mathbf{d}(\mathbf{x},t) &= -\delta\rho\int_{\mathcal{B}}\frac{\partial^2\mathbf{X}}{\partial t^2}\cdot\boldsymbol{\delta}(\mathbf{x}-\mathbf{X}(\mathbf{s},t))\,\mathrm{d}\mathbf{s}\\ \mathbf{f}(\mathbf{x},t) &= \int_{\mathcal{B}}(\operatorname{div}_{\mathbf{s}}\mathbb{P})\boldsymbol{\delta}(\mathbf{x}-\mathbf{X}(\mathbf{s},t))\,\mathrm{d}\mathbf{s}\\ \mathbf{t}(\mathbf{x},t) &= -\int_{\partial\mathcal{B}}\mathbb{P}\mathbf{N}\boldsymbol{\delta}(\mathbf{x}-\mathbf{X}(\mathbf{s},t))\,\mathrm{dA}\end{aligned} \tag{1.26}$$

as, respectively, the excess Lagrangian mass density, the inner force density and the transmission force density.

Now, exploiting the definition of $\boldsymbol{\sigma}_f$ given in (1.3), we deduce the Navier–Stokes equation in Ω

$$\begin{aligned}\rho_f\dot{\mathbf{u}} - \operatorname{div}\boldsymbol{\sigma}_f &= \rho_f\left(\frac{\partial\mathbf{u}}{\partial t}+\mathbf{u}\cdot\nabla\,\mathbf{u}\right)-\operatorname{div}\left(-p\mathbb{I}+\nu\,\underline{\boldsymbol{\varepsilon}}(\mathbf{u})\right)\\ &= \rho_f\left(\frac{\partial\mathbf{u}}{\partial t}+\mathbf{u}\cdot\nabla\,\mathbf{u}\right)+\nabla\,p-\nu\operatorname{div}(\underline{\boldsymbol{\varepsilon}}(\mathbf{u}))\end{aligned} \tag{1.27}$$

so that we can present the strong formulation of our model problem for fluid-structure interactions.

Problem 1.1.1. *Find* $\boldsymbol{u}$*,* p *and* $\boldsymbol{X}$ *which satisfy:*

$$\rho_f\left(\frac{\partial\mathbf{u}}{\partial t}+\mathbf{u}\cdot\nabla\,\mathbf{u}\right)+\nabla\,p-\nu\operatorname{div}(\underline{\boldsymbol{\varepsilon}}(\mathbf{u}))=\mathbf{d}+\mathbf{f}+\mathbf{t} \quad \textit{in } \Omega\times(0,T) \tag{1.28a}$$

$$\operatorname{div}\mathbf{u}=0 \quad \textit{in } \Omega\times(0,T) \tag{1.28b}$$

$$\mathbf{d}(\mathbf{x},t)=-\delta\rho\int_{\mathcal{B}}\frac{\partial^2\mathbf{X}}{\partial t^2}\cdot\boldsymbol{\delta}(\mathbf{x}-\mathbf{X}(\mathbf{s},t))\,\mathrm{d}\mathbf{s} \quad \textit{in } \Omega\times(0,T) \tag{1.28c}$$

$$\mathbf{f}(\mathbf{x},t)=\int_{\mathcal{B}}(\operatorname{div}_{\mathbf{s}}\mathbb{P})\boldsymbol{\delta}(\mathbf{x}-\mathbf{X}(\mathbf{s},t))\,\mathrm{d}\mathbf{s} \quad \textit{in } \Omega\times(0,T) \tag{1.28d}$$

$$\mathbf{t}(\mathbf{x},t)=-\int_{\partial\mathcal{B}}\mathbb{P}\mathbf{N}\boldsymbol{\delta}(\mathbf{x}-\mathbf{X}(\mathbf{s},t))\,\mathrm{dA} \quad \textit{in } \Omega\times(0,T) \tag{1.28e}$$

$$\frac{\partial\mathbf{X}}{\partial t}(\mathbf{s},t)=\mathbf{u}(\mathbf{X}(\mathbf{s},t),t) \quad \textit{in } \mathcal{B}\times(0,T) \tag{1.28f}$$

$$\mathbf{u}(\mathbf{x},t) = 0 \qquad \text{on } \partial\Omega \times (0,T) \tag{1.28g}$$
$$\mathbf{u}(\mathbf{x},0) = \mathbf{u}_0(\mathbf{x}) \qquad \text{in } \Omega \tag{1.28h}$$
$$\mathbf{X}(\mathbf{x},0) = \mathbf{X}_0(\mathbf{s}) \qquad \text{in } \mathcal{B}. \tag{1.28i}$$

We now derive the variational formulation of our problem to discretize it using the finite element method. In order to treat the source terms $\mathbf{d}, \mathbf{f}, \mathbf{t}$ as linear continuous functionals in distributional sense, we introduce the following lemma [23, 29].

Lemma 1.1.1. *Assume that $\mathcal{B}$ is a Lipschitz domain, that, $\forall t \in [0,T]$, $\mathbf{X} \in \mathbf{W}^{1,\infty}(\mathcal{B})$, $\partial^2\mathbf{X}/\partial t^2 \in \mathbf{H}^{-1}(\mathcal{B})$ and $\mathbb{P} \in \mathbf{W}^{1,\infty}(\mathcal{B})$. Then $\forall t \in (0,T)$, the excess Lagrangian mass density $\mathbf{d}$ and the force density $\mathbf{F}(t) = \mathbf{f} + \mathbf{t}$ are distribution functions belonging to $\mathbf{H}^{-1}(\mathcal{B})$ defined as follow: for all $\mathbf{v} \in \mathbf{H}_0^1(\Omega)$*

$$\begin{aligned} {}_{\mathrm{H}^{-1}}\langle \mathbf{d}(t), \mathbf{v}\rangle_{\mathrm{H}_0^1} &= -\delta\rho \int_{\mathcal{B}} \frac{\partial^2 \mathbf{X}}{\partial t^2} \mathbf{v}(\mathbf{X}(\mathbf{s},t))\, \mathrm{d}\mathbf{s} && \forall t \in (0,T) \\ {}_{\mathrm{H}^{-1}}\langle \mathbf{F}(t), \mathbf{v}\rangle_{\mathrm{H}_0^1} &= -\int_{\mathcal{B}} \mathbb{P}(\mathbb{F}(\mathbf{s},t)) : \underline{\boldsymbol{\varepsilon}}(\mathbf{v}(\mathbf{X}(\mathbf{s},t))\, \mathrm{d}\mathbf{s} && \forall t \in (0,T). \end{aligned} \tag{1.29}$$

On the other hand, for (1.28a) and (1.28b) we work in the usual way. For (1.28a), if we consider a test function $\mathbf{v} \in \mathbf{H}_0^1(\Omega)$, we obtain

$$\begin{aligned} \int_\Omega \rho_f \left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla\, \mathbf{u} \right) \mathbf{v}\, \mathrm{d}\mathbf{x} + \int_\Omega \nabla\, p \cdot \mathbf{v}\, \mathrm{d}\mathbf{x} \\ - \int_\Omega \nu\, \mathrm{div}(\underline{\boldsymbol{\varepsilon}}(\mathbf{u}))\mathbf{v}\, \mathrm{d}\mathbf{x} = \langle \mathbf{d}, \mathbf{v}\rangle + \langle \mathbf{F}, \mathbf{v}\rangle, \end{aligned} \tag{1.30}$$

where the second and the third terms can be integrated by parts taking into account that the boundary terms vanish, hence

$$-\int_\Omega \nu\, \mathrm{div}(\underline{\boldsymbol{\varepsilon}}(\mathbf{u}))\mathbf{v}\, \mathrm{d}\mathbf{x} = \nu \int_\Omega \underline{\boldsymbol{\varepsilon}}(\mathbf{u}) : \underline{\boldsymbol{\varepsilon}}(\mathbf{v})\, \mathrm{d}\mathbf{x}$$
$$\int_\Omega \nabla\, p \cdot \mathbf{v}\, \mathrm{d}\mathbf{x} = -\int_\Omega p\ \mathrm{div}\, \mathbf{v}\, \mathrm{d}\mathbf{x}$$

and the first is expanded using linearity.

For (1.28b), we simply have that

$$\int_\Omega (\mathrm{div}\, \mathbf{u}) q\, \mathrm{d}\mathbf{x} = 0 \quad \forall q \in \mathrm{L}_0^2(\Omega). \tag{1.31}$$

Finally, we obtain the variational formulation of Problem 1.1.1.

Problem 1.1.2. *Given* $\mathbf{u}_0 \in \mathbf{H}^1_0(\Omega)$ *and* $\mathbf{X}_0 : \mathcal{B} \longrightarrow \Omega^s_t$, $\forall t \in (0,T)$, *find* $(\mathbf{u}(t), p(t)) \in \mathbf{H}^1_0(\Omega) \times \mathrm{L}^2_0(\Omega)$ *and* $\mathbf{X}(t) \in \mathbf{W}^{1,\infty}(\mathcal{B})$, *such that*

$$\rho_f \frac{\partial}{\partial t}(\mathbf{u}(t), \mathbf{v})_\Omega + b(\mathbf{u}(t), \mathbf{u}(t), \mathbf{v}) + a(\mathbf{u}(t), \mathbf{v}) - (\operatorname{div} \mathbf{v}, p(t))_\Omega = \langle \mathbf{d}(t), \mathbf{v} \rangle + \langle \mathbf{F}(t), \mathbf{v} \rangle \qquad \forall \mathbf{v} \in \mathbf{H}^1_0(\Omega) \tag{1.32a}$$

$$(\operatorname{div} \mathbf{u}(t), q)_\Omega = 0 \qquad \forall q \in \mathrm{L}^2_0(\Omega) \tag{1.32b}$$

$$\langle \mathbf{d}(t), \mathbf{v} \rangle = -\delta\rho \int_{\mathcal{B}} \frac{\partial^2 \mathbf{X}}{\partial t^2} \mathbf{v}(\mathbf{X}(\mathbf{s}, t))\, \mathrm{d}\mathbf{s} \qquad \forall \mathbf{v} \in \mathbf{H}^1_0(\Omega) \tag{1.32c}$$

$$\langle \mathbf{F}(t), \mathbf{v} \rangle = -\int_{\mathcal{B}} \mathbb{P}(\mathbb{F}(\mathbf{s}, t)) : \boldsymbol{\nabla}_{\mathrm{s}}\, \mathbf{v}(\mathbf{X}(\mathbf{s}, t))\, \mathrm{d}\mathbf{s} \qquad \forall \mathbf{v} \in \mathbf{H}^1_0(\Omega) \tag{1.32d}$$

$$\frac{\partial \mathbf{X}}{\partial t}(\mathbf{s}, t) = \mathbf{u}(\mathbf{X}(\mathbf{s}, t), t) \qquad \forall \mathbf{s} \in \mathcal{B} \tag{1.32e}$$

$$\mathbf{u}(\mathbf{x}, 0) = \mathbf{u}_0(\mathbf{x}) \qquad \forall \mathbf{x} \in \Omega \tag{1.32f}$$

$$\mathbf{X}(\mathbf{s}, 0) = \mathbf{X}_0(\mathbf{s}) \qquad \forall \mathbf{s} \in \mathcal{B} \tag{1.32g}$$

where

$$a(\mathbf{u}, \mathbf{v}) = \nu(\underline{\boldsymbol{\varepsilon}}(\mathbf{u}), \underline{\boldsymbol{\varepsilon}}(\mathbf{v}))_\Omega$$
$$b(\mathbf{u}, \mathbf{v}, \mathbf{w}) = \frac{\rho_f}{2}\big((\mathbf{u} \cdot \nabla \mathbf{v}, \mathbf{w})_\Omega - (\mathbf{u} \cdot \nabla \mathbf{w}, \mathbf{v})_\Omega\big).$$

Moving from Problem 1.1.1 to Problem 1.1.2, we have introduced the Navier–Stokes trilinear form b: in particular, we have implicitly used the result of the following proposition [93].

Proposition 1.1.1. *The trilinear form* b *satisfies the following equalities*

$$b(\mathbf{u}, \mathbf{u}, \mathbf{v}) = \int_\Omega (\mathbf{u} \cdot \nabla \mathbf{u}) \mathbf{v}\, \mathrm{d}\mathbf{x} = \frac{1}{2} \int_\Omega (\mathbf{u} \cdot \nabla \mathbf{u}) \mathbf{v}\, \mathrm{d}\mathbf{x} - \frac{1}{2} \int_\Omega (\mathbf{u} \cdot \nabla \mathbf{v}) \mathbf{u}\, \mathrm{d}\mathbf{x}. \tag{1.33}$$

1.1.3 Stability estimate

Before introducing the new formulation of Problem 1.1.2 based on the use of a distributed Lagrange multiplier in the spirit of the fictitious domain approach, let us discuss its stability [19].

In particular, we recall the main features of hyper-elastic materials [29, 18, 65, 64], which are considered in our model. We introduce a positive potential energy $W(\mathbb{F}) > 0$ which depends only on the deformation gradient; indeed, the properties of the body are a consequence of its current position and not of the path it followed during its evolution.

The properties of W are consequence of physical considerations:

- if no deformation is applied to the material, there is no energy to be stored, hence $\mathbb{F} = \mathbb{I}$ and $W(\mathbb{I}) = 0$;
- an object made up of hyper-elastic material cannot be infinitely expanded or shrunk to have null volume, therefore

$$|\mathbb{F}| \longrightarrow 0, \infty \implies W(\mathbb{F}) \longrightarrow 0;$$

- the energy function is independent from rigid body motions, so that

$$W(\widetilde{\mathbb{F}}) = W(\mathbb{F})$$

  if $\widetilde{\mathbb{F}}$ is obtained from $\mathbb{F}$ as a consequence of a rigid motion.

Combining the third property with the fact that $\mathbb{F}$ does not change under rigid translations, the energy W is such that for all rotations $\mathfrak{R} \in \mathbb{R}^{d\times d}$ satisfying $\mathfrak{R}^\top \mathfrak{R} = \mathbb{I}$ and $\det \mathfrak{R} > 0$

$$\mathbf{Y} = \mathfrak{R}\mathbf{X} + \mathbf{U} \implies W(\boldsymbol{\nabla}_{\mathrm{s}} \mathbf{Y}) = W(\boldsymbol{\nabla}_{\mathrm{s}} \mathbf{X} \cdot \mathfrak{R}^\top) = W(\boldsymbol{\nabla}_{\mathrm{s}} \mathbf{X}) \tag{1.34}$$

where $\mathbf{U}$ is a fixed translation vector.

Moreover, the link between this energy density W and the first Piola–Kirchhoff stress tensor is given by the following relation

$$(\mathbb{P}(\mathbb{F}(\mathbf{s}, t)))_{i,j} = \frac{\partial W}{\partial \mathbb{F}_{i,j}}(\mathbb{F}(\mathbf{s}, t)) = \left(\frac{\partial W}{\partial \mathbb{F}}(\mathbf{s}, t)\right)_{i,j} \tag{1.35}$$

where $i, j = 1, \ldots, d$. A variation in the deformation of a rigid body produces a corresponding variation on its energy, with the generation of an internal stress to re-establish the original configuration: this stress is obtained in Lagrangian framework.

Finally, we have that the total elastic potential energy associated with $\mathcal{B}$ is expressed as

$$\mathscr{E}(\mathbf{X}(t)) = \int_{\mathcal{B}} W(\mathbb{F}(\mathbf{s}, t))\, \mathrm{d}\mathbf{s}. \tag{1.36}$$

Remark 1.1.1. *In general, for elasticity problems, we should assume that the potential energy W is polyconvex [62]. Since we are considering incompressible materials, $|\mathbb{F}|$ is constant in time and it makes sense to assume W to be a C^1 convex function.*

In the following proposition [18], we state the stability estimate for Problem 1.1.2.

Proposition 1.1.2. *Let us assume that for almost every $t \in [0,T]$, $\mathbf{u}(t) \in \mathbf{H}_0^1(\Omega)$ and $\mathbf{X}(t) \in \mathbf{W}^{1,\infty}(\mathcal{B})$ are solutions for Problem 1.1.2, then the following equation holds true*

$$\frac{\rho_f}{2}\frac{\partial}{\partial t}\left\|\mathbf{u}(t)\right\|_{0,\Omega}^2 + \nu\left\|\underline{\boldsymbol{\varepsilon}}(\mathbf{u}(t))\right\|_{0,\Omega}^2 + \frac{\delta\rho}{2}\frac{\partial}{\partial t}\left\|\frac{\partial \mathbf{X}}{\partial t}\right\|_{0,\mathcal{B}}^2 + \frac{\partial}{\partial t}\mathscr{E}(\mathbf{X}(t)) = 0. \tag{1.37}$$

1.2 Fictitious domain approach with distributed Lagrange multiplier

We now introduce the recent formulation of the immersed boundary method based on a distributed Lagrange multiplier [19]. We observe that the method has been developed in the spirit of the fictitious domain approach, where the fluid domain Ω_t^f is extended also in Ω_t^s introducing Ω. In particular, the introduction of the distributed Lagrange multiplier has the aim of enforcing the kinematic constraint

$$\frac{\partial \mathbf{X}}{\partial t} = \mathbf{u}(\mathbf{X}(\mathbf{s},t),t) \quad \text{for } \mathbf{x} = \mathbf{X}(\mathbf{s},t). \tag{1.38}$$

For this purpose, let us consider two Hilbert spaces $\underline{\boldsymbol{\Lambda}}$, $\mathbf{M}$ and define a continuous bilinear form $\mathbf{c} : \underline{\boldsymbol{\Lambda}} \times \mathbf{M} \longrightarrow \mathbb{R}$ satisfying the property

$$\mathbf{c}(\boldsymbol{\mu}, \mathbf{Z}) = 0 \quad \forall \boldsymbol{\mu} \in \underline{\boldsymbol{\Lambda}} \implies \mathbf{Z} = 0 \text{ in } \mathcal{B}. \tag{1.39}$$

A reasonable choice for $\mathbf{M}$ is $\mathbf{H}^1(\mathcal{B})$ since we can assume that $\mathbf{X}(t) \in \mathbf{W}^{1,\infty}(\mathcal{B})$ for almost every $t \in (0,T)$ so that $\mathbf{u}(\mathbf{X}(\cdot,t),t) \in \mathbf{H}^1(\mathcal{B})$.

Moreover, we present two possible choices for $\underline{\boldsymbol{\Lambda}}$ and the form $\mathbf{c}$: we can consider $\mathbf{c}$ as the duality pairing between $\mathbf{H}^1(\mathcal{B})$ and its dual space $\underline{\boldsymbol{\Lambda}} = (\mathbf{H}^1(\mathcal{B}))'$

$$\mathbf{c}(\boldsymbol{\mu}, \mathbf{Z}) = \langle \boldsymbol{\mu}, \mathbf{Z} \rangle \quad \forall \boldsymbol{\mu} \in \underline{\boldsymbol{\Lambda}}, \forall \mathbf{Z} \in \mathbf{H}^1(\mathcal{B}); \tag{1.40}$$

alternatively, we can take $\underline{\boldsymbol{\Lambda}} = \mathbf{M} = \mathbf{H}^1(\mathcal{B})$ and $\mathbf{c}$ as the scalar product in $\mathbf{H}^1(\mathcal{B})$, therefore

$$\mathbf{c}(\boldsymbol{\mu}, \mathbf{Z}) = (\boldsymbol{\mu}, \mathbf{Z})_{\mathcal{B}} + (\boldsymbol{\nabla}_{\mathrm{s}}\,\boldsymbol{\mu}, \boldsymbol{\nabla}_{\mathrm{s}}\,\mathbf{Z})_{\mathcal{B}} \quad \forall \boldsymbol{\mu}, \mathbf{Z} \in \mathbf{H}^1(\mathcal{B}). \tag{1.41}$$

Using the defining property of $\mathbf{c}$ given by (1.39), Equation (1.38) is equivalent to

$$\mathbf{c}\left(\boldsymbol{\mu}, \mathbf{u}(\mathbf{X}(\mathbf{s},t),t) - \frac{\partial \mathbf{X}(\mathbf{s},t)}{\partial t}\right) = 0 \quad \forall \boldsymbol{\mu} \in \underline{\boldsymbol{\Lambda}}. \tag{1.42}$$

Clearly, we need also to add terms involving $\mathbf{c}$ and the multiplier in the equations for fluid and solid in order to couple them. Therefore, introducing the Lagrange multiplier $\boldsymbol{\lambda}(t) \in \underline{\boldsymbol{\Lambda}}$ and taking again into account (1.39), we can rewrite Equation (1.32a) in the following way

$$\begin{aligned}
&\rho_f \frac{\partial}{\partial t}(\mathbf{u}(t), \mathbf{v})_\Omega + b(\mathbf{u}(t), \mathbf{u}(t), \mathbf{v}) + a(\mathbf{u}(t), \mathbf{v}) + \mathbf{c}(\boldsymbol{\lambda}(t), \mathbf{v}(\mathbf{X}(t))) = 0 \qquad \forall \mathbf{v} \in \mathbf{H}^1_0(\Omega) \\
&\delta\rho\left(\frac{\partial^2 \mathbf{X}}{\partial t^2}(t), \mathbf{Y}\right)_{\mathcal{B}} + (\mathbb{P}(\mathbb{F}(t)), \boldsymbol{\nabla}_{\mathrm{s}}\, \mathbf{Y})_{\mathcal{B}} - \mathbf{c}(\boldsymbol{\lambda}(t), \mathbf{Y}) = 0 \qquad \forall \mathbf{Y} \in \mathbf{H}^1(\mathcal{B}).
\end{aligned} \tag{1.43}$$

At this point, we can rewrite Problem 1.1.2 in its new form.

Problem 1.2.1. *Given* $\mathbf{u}_0 \in \mathbf{H}^1_0(\Omega)$ *and* $\mathbf{X}_0 : \mathcal{B} \longrightarrow \Omega$, $\forall t \in (0, T)$, *find* $\mathbf{u}(t) \in \mathbf{H}^1_0(\Omega)$, $p(t) \in \mathrm{L}^2_0(\Omega)$, $\mathbf{X}(t) \in \mathbf{W}^{1,\infty}(\mathcal{B})$ *and* $\boldsymbol{\lambda}(t) \in \underline{\boldsymbol{\Lambda}}$, *such that*

$$\rho_f \frac{\partial}{\partial t}(\mathbf{u}(t), \mathbf{v})_\Omega + b(\mathbf{u}(t), \mathbf{u}(t), \mathbf{v}) + a(\mathbf{u}(t), \mathbf{v}) - (\operatorname{div} \mathbf{v}, p(t))_\Omega + \mathbf{c}(\boldsymbol{\lambda}(t), \mathbf{v}(\mathbf{X}(t))) = 0 \quad \forall \mathbf{v} \in \mathbf{H}^1_0(\Omega) \tag{1.44a}$$

$$(\operatorname{div} \mathbf{u}(t), q)_\Omega = 0 \qquad \forall q \in \mathrm{L}^2_0(\Omega) \tag{1.44b}$$

$$\delta\rho\left(\frac{\partial^2 \mathbf{X}}{\partial t^2}, \mathbf{Y}\right)_{\mathcal{B}} + (\mathbb{P}(\mathbb{F}(\mathbf{s}, t)), \boldsymbol{\nabla}_{\mathrm{s}}\, \mathbf{Y})_{\mathcal{B}} - \mathbf{c}(\boldsymbol{\lambda}(t), \mathbf{Y}) = 0 \qquad \forall \mathbf{Y} \in \mathbf{H}^1(\mathcal{B}) \tag{1.44c}$$

$$\mathbf{c}\left(\boldsymbol{\mu}, \mathbf{u}(\mathbf{X}(\cdot, t), t) - \frac{\partial \mathbf{X}}{\partial t}(t)\right) = 0 \qquad \forall \boldsymbol{\mu} \in \underline{\boldsymbol{\Lambda}} \tag{1.44d}$$

$$\mathbf{u}(\mathbf{x}, 0) = \mathbf{u}_0(\mathbf{x}) \qquad \forall \mathbf{x} \in \Omega \tag{1.44e}$$

$$\mathbf{X}(\mathbf{s}, 0) = \mathbf{X}_0(\mathbf{s}) \qquad \forall \mathbf{s} \in \mathcal{B} \tag{1.44f}$$

Also for this new formulation we can write an energy estimate similar to the one presented in Proposition 1.1.2.

Proposition 1.2.1 ([19])**.** *For almost every* $t \in (0, T)$, *let* $\mathbf{u}(t) \in \mathbf{H}^1_0(\Omega)$ *and* $\mathbf{X}(t) \in \mathbf{H}^1(\mathcal{B})$ *be solutions of Problem 1.2.1 with* $\partial \mathbf{X}(t)/\partial t \in \mathbf{L}^2(\mathcal{B})$, *then the following energy estimate holds true*

$$\frac{\rho_f}{2}\frac{\partial}{\partial t}\left\|\mathbf{u}(t)\right\|^2_{0,\Omega} + \nu\left\|\underline{\boldsymbol{\varepsilon}}(\mathbf{u}(t))\right\|^2_{0,\Omega} + \frac{\delta\rho}{2}\frac{\partial}{\partial t}\left\|\frac{\partial \mathbf{X}}{\partial t}\right\|^2_{0,\mathcal{B}} + \frac{\partial}{\partial t}\mathscr{E}(\mathbf{X}(t)) = 0. \tag{1.45}$$

Remark 1.2.1. *Existence and uniqueness of the solution for Problem 1.2.1 have been proved in [25] following the Galerkin approximation technique used in [93, Chapt.III.1]. A linearized version of the problem is considered: the convective term*

b is neglected and $\mathbb{P}(\mathbb{F}) = \kappa \nabla_s \mathbb{F}$. *Moreover,* $\mathbf{c}$ *is defined as in Equation* (1.41). *In particular, we have*

$$\begin{aligned}
&\mathbf{u} \in \mathbf{L}^\infty(0,T;\mathscr{V}_0) \cap \mathbf{L}^2(0,T;\mathscr{H}_0)\\
&p \in \mathrm{L}^2(0,T;\mathrm{L}^2_0(\Omega))\\
&\mathbf{X} \in \mathbf{L}^\infty(0,T;\mathbf{H}^1(\mathcal{B})) \text{ with } \frac{\partial \mathbf{X}}{\partial t} \in \mathbf{L}^\infty(0,T;\mathbf{L}^2(\mathcal{B})) \cap \mathbf{L}^2(0,T;\mathbf{H}^1(\mathcal{B}))\\
&\boldsymbol{\lambda} \in \mathbf{L}^2(0,T;\mathbf{H}^1(\mathcal{B}))
\end{aligned}$$

where

$$\begin{aligned}
\mathscr{V} &= \{\mathbf{v} \in (\mathcal{D}(\Omega))^d : \operatorname{div} \mathbf{v} = 0\}\\
\mathscr{H}_0 &= \text{ the closure of } \mathscr{V} \text{ in } \mathbf{H}^1_0(\Omega)\\
\mathscr{V}_0 &= \text{ the closure of } \mathscr{V} \text{ in } \mathbf{L}^2_0(\Omega).
\end{aligned}$$

and $\mathcal{D}(\Omega)$ *denotes the space of infinitely differentiable functions with compact support in* Ω.

1.3 Time semi-discretization

In this section, we discuss the time semi-discretization of Problem 1.2.1. We present a scheme based on backward finite differences [19]. We will show that also high order schemes can be considered.

Let us consider $N \in \mathbb{N}$ and subdivide the time interval $[0,T]$ in N equal parts with time step $\Delta t = T/N$ and nodes $t_n = n\Delta t$ for $n = 0,\dots,N$. The approximation of time derivatives for a generic function f reads as follows

$$\frac{\partial f}{\partial t}(t_{n+1}) \approx \frac{f^{n+1} - f^n}{\Delta t}, \qquad \frac{\partial^2 f}{\partial t^2}(t_{n+1}) \approx \frac{f^{n+1} - 2f^n + f^{n-1}}{\Delta t^2}. \tag{1.46}$$

As a consequence, we obtain the following fully implicit scheme.

Problem 1.3.1. *Given* $\mathbf{u}^0 \in \mathbf{H}^1_0(\Omega)$ *and* $\mathbf{X}^0 \in \mathbf{W}^{1,\infty}(\mathcal{B})$, *for* $n = 1,\dots,N$ *find* $(\mathbf{u}^n, p^n) \in \mathbf{H}^1_0(\Omega) \times \mathrm{L}^2_0(\Omega)$, $\mathbf{X}^n \in \mathbf{H}^1(\mathcal{B})$, *and* $\boldsymbol{\lambda}^n \in \underline{\boldsymbol{\Lambda}}$, *such that*

$$\begin{aligned}
&\rho_f\left(\frac{\mathbf{u}^{n+1} - \mathbf{u}^n}{\Delta t}, \mathbf{v}\right)_\Omega + b(\mathbf{u}^{n+1}, \mathbf{u}^{n+1}, \mathbf{v}) + a(\mathbf{u}^{n+1}, \mathbf{v})\\
&\qquad - (\operatorname{div} \mathbf{v}, p^{n+1})_\Omega + \mathbf{c}(\boldsymbol{\lambda}^{n+1}, \mathbf{v}(\mathbf{X}^{n+1})) = 0 \qquad \forall \mathbf{v} \in \mathbf{H}^1_0(\Omega) \qquad (1.47a)\\
&(\operatorname{div} \mathbf{u}^{n+1}, q)_\Omega = 0 \qquad \forall q \in \mathrm{L}^2_0(\Omega) \qquad (1.47b)
\end{aligned}$$

$$\delta\rho\left(\frac{\mathbf{X}^{n+1}-2\mathbf{X}^{n}+\mathbf{X}^{n-1}}{\Delta t^2},\mathbf{Y}\right)_{\mathcal{B}}+(\mathbb{P}(\mathbb{F}^{n+1}),\boldsymbol{\nabla}_{\mathrm{s}}\,\mathbf{Y})_{\mathcal{B}}$$
$$-\mathbf{c}(\boldsymbol{\lambda}^{n+1},\mathbf{Y})=0 \qquad \forall\mathbf{Y}\in\mathbf{H}^1(\mathcal{B}) \qquad (1.47c)$$
$$\mathbf{c}\left(\boldsymbol{\mu},\mathbf{u}^{n+1}(\mathbf{X}^{n+1})-\frac{\mathbf{X}^{n+1}-\mathbf{X}^{n}}{\Delta t}\right)=0 \qquad \forall\boldsymbol{\mu}\in\underline{\boldsymbol{\Lambda}} \qquad (1.47d)$$

However, there is a detail to be discussed. We have to compute terms related to $\mathbf{H}^1_0(\Omega)$ functions evaluated along the structure through the mapping $\mathbf{X}^{n+1}$: for example, the term $\mathbf{v}(\mathbf{X}^{n+1})$ requires the knowledge of $\mathbf{X}^{n+1}$ which has not been computed yet, hence we choose to evaluate these terms using the mapping $\mathbf{X}^n$ related to the previous step. The same change is done also on b, which is linearized by computing the transport velocity at the previous time step. Finally, we obtain a modified Backward Euler time advancing scheme.

Problem 1.3.2. *Given* $\mathbf{u}^0\in\mathbf{H}^1_0(\Omega)$ *and* $\mathbf{X}^0\in\mathbf{W}^{1,\infty}(\mathcal{B})$*, for* $n=1,\dots,N$ *find* $(\mathbf{u}^n,p^n)\in\mathbf{H}^1_0(\Omega)\times\mathrm{L}^2_0(\Omega)$*,* $\mathbf{X}^n\in\mathbf{H}^1(\mathcal{B})$*, and* $\boldsymbol{\lambda}^n\in\underline{\boldsymbol{\Lambda}}$*, such that*

$$\rho_f\left(\frac{\mathbf{u}^{n+1}-\mathbf{u}^{n}}{\Delta t},\mathbf{v}\right)_{\Omega}+b(\mathbf{u}^n,\mathbf{u}^{n+1},\mathbf{v})+a(\mathbf{u}^{n+1},\mathbf{v})$$
$$-(\operatorname{div}\mathbf{v},p^{n+1})_{\Omega}+\mathbf{c}(\boldsymbol{\lambda}^{n+1},\mathbf{v}(\mathbf{X}^n))=0 \qquad \forall\mathbf{v}\in\mathbf{H}^1_0(\Omega) \qquad (1.48a)$$
$$(\operatorname{div}\mathbf{u}^{n+1},q)_{\Omega}=0 \qquad \forall q\in\mathrm{L}^2_0(\Omega) \qquad (1.48b)$$
$$\delta\rho\left(\frac{\mathbf{X}^{n+1}-2\mathbf{X}^{n}+\mathbf{X}^{n-1}}{\Delta t^2},\mathbf{Y}\right)_{\mathcal{B}}+(\mathbb{P}(\mathbb{F}^{n+1}),\boldsymbol{\nabla}_{\mathrm{s}}\,\mathbf{Y})_{\mathcal{B}}$$
$$-\mathbf{c}(\boldsymbol{\lambda}^{n+1},\mathbf{Y})=0 \qquad \forall\mathbf{Y}\in\mathbf{H}^1(\mathcal{B}) \qquad (1.48c)$$
$$\mathbf{c}\left(\boldsymbol{\mu},\mathbf{u}^{n+1}(\mathbf{X}^{n})-\frac{\mathbf{X}^{n+1}-\mathbf{X}^{n}}{\Delta t}\right)=0 \qquad \forall\boldsymbol{\mu}\in\underline{\boldsymbol{\Lambda}} \qquad (1.48d)$$

In particular, we can define $\mathbf{X}^{-1}$ manipulating

$$\frac{\mathbf{X}^0-\mathbf{X}^{-1}}{\Delta t}=\mathbf{u}^s_0 \quad \text{ in } \mathcal{B}. \qquad (1.49)$$

This semi-implicit time advancing scheme is unconditionally stable since we can prove the following energy estimate without limitations on the choice of the time step [19].

Proposition 1.3.1. *Let us assume* W *to be a convex function as in Remark 1.1.1 and* $\delta\rho>0$*. Let* $\mathbf{u}^n\in\mathbf{H}^1_0(\Omega)$ *and* $\mathbf{X}^n\in\mathbf{H}^1(\mathcal{B})$ *for* $n=0,\dots,N$ *satisfying Problem 1.3.2 with* $\mathbf{X}^n\in\mathbf{W}^{1,\infty}(\mathcal{B})$*. Then the following estimate holds true for all*

$n = 0, \dots, N-1$

$$\frac{\rho_f}{2\Delta t}\left(\left\|\mathbf{u}^{n+1}\right\|_{0,\Omega}^2 - \left\|\mathbf{u}^{n}\right\|_{0,\Omega}^2 \right) + \nu \left\|\underline{\boldsymbol{\varepsilon}}(\mathbf{u}^{n+1})\right\|_{0,\Omega}^2 + \frac{\delta\rho}{2\Delta t}\left(\left\| \frac{\mathbf{X}^{n+1}-\mathbf{X}^{n}}{\Delta t} \right\|_{0,\mathcal{B}}^2 - \left\| \frac{\mathbf{X}^{n}-\mathbf{X}^{n-1}}{\Delta t} \right\|_{0,\mathcal{B}}^2 \right) + \frac{\mathscr{E}(\mathbf{X}^{n+1}) - \mathscr{E}(\mathbf{X}^{n})}{\Delta t} \leq 0 \tag{1.50}$$

As mentioned at the beginning of this section, high order schemes can be chosen for the time discretization. For instance, in [31], the second order Backward differentiation formula BDF2 was considered following the idea already used in [50, 39, 84] producing an unconditionally stable numerical scheme. Moreover, a Crank–Nicolson scheme, implemented with both the midpoint and the trapezoidal rule, was presented: in the first case, the unconditional stability is still valid, while in the second case it is not straightforward to be proved; nevertheless, several numerical tests did not show any type of instability.

Remark 1.3.1. *We point out that Problem 1.3.1 and Problem 1.3.2 can be interpreted as a sequence of stationary problems. In the next sections, we analyze the associated stationary problem, which has the structure of saddle point problem, in both the continuous and discrete setting.*

1.4 Finite element discretization

We have just presented the time semi-discretization of our fluid-structure interaction problem, so that we can now introduce the discretization in space making use of mixed finite elements. Here we describe the main features since the more technical aspects will be addressed in the next sections.

Let us consider a mesh $\mathcal{T}_h^{\Omega}$ for the domain Ω with mesh size h_Ω and a second mesh $\mathcal{T}_h^{\mathcal{B}}$, with mesh size $h_{\mathcal{B}}$, for the solid reference domain $\mathcal{B}$. Accordingly with the theory of the mixed formulation for the Stokes problem [15], we consider two finite dimensional subspaces $\underline{\mathbf{V}}_h \subset \mathbf{H}_0^1(\Omega)$ and $Q_h \subset \mathrm{L}_0^2(\Omega)$ satisfying the inf-sup condition to approximate the fluid variables $\mathbf{u}$ and p. Moreover, we consider other two discrete spaces for the approximation of $\mathbf{X}$ and $\boldsymbol{\lambda}$: $\underline{\mathbf{S}}_h \subset \mathbf{H}^1(\mathcal{B})$ and $\underline{\boldsymbol{\Lambda}}_h \subset \underline{\boldsymbol{\Lambda}}$. In this setting, we are able to present the finite element counterpart of Problem 1.3.2.

Problem 1.4.1. *Given* $\mathbf{u}_h^0 \in \underline{\mathbf{V}}_h$ *and* $\mathbf{X}_h^0 \in \mathbf{W}^{1,\infty}(\mathcal{B})$*, for* $n = 1, \ldots, N$ *find* $\mathbf{u}_h^n \in \underline{\mathbf{V}}_h$*,* $p_h^n \in Q_h$*,* $\mathbf{X}_h^n \in \underline{\mathbf{S}}_h$*, and* $\boldsymbol{\lambda}_h^n \in \underline{\boldsymbol{\Lambda}}_h$*, such that*

$$\rho_f\left(\frac{\mathbf{u}_h^{n+1}-\mathbf{u}_h^n}{\Delta t},\mathbf{v}\right)_\Omega + b(\mathbf{u}_h^n,\mathbf{u}_h^{n+1},\mathbf{v}) + a(\mathbf{u}_h^{n+1},\mathbf{v}) - (\operatorname{div}\mathbf{v},p_h^{n+1})_\Omega + \mathbf{c}(\boldsymbol{\lambda}_h^{n+1},\mathbf{v}(\mathbf{X}_h^n)) = 0 \qquad \forall \mathbf{v}\in\underline{\mathbf{V}}_h \tag{1.51a}$$

$$(\operatorname{div}\mathbf{u}_h^{n+1},q)_\Omega = 0 \qquad \forall q \in Q_h \tag{1.51b}$$

$$\delta\rho\left(\frac{\mathbf{X}_h^{n+1}-2\mathbf{X}_h^n+\mathbf{X}_h^{n-1}}{\Delta t^2},\mathbf{Y}\right)_\mathcal{B} + (\mathbb{P}(\mathbb{F}_h^{n+1}),\boldsymbol{\nabla}_{\mathrm{s}}\,\mathbf{Y})_\mathcal{B} - \mathbf{c}(\boldsymbol{\lambda}_h^{n+1},\mathbf{Y}) = 0 \qquad \forall \mathbf{Y}\in\underline{\mathbf{S}}_h \tag{1.51c}$$

$$\mathbf{c}\left(\boldsymbol{\mu},\mathbf{u}_h^{n+1}(\mathbf{X}_h^n) - \frac{\mathbf{X}_h^{n+1}-\mathbf{X}_h^n}{\Delta t}\right) = 0 \qquad \forall \boldsymbol{\mu}\in\underline{\boldsymbol{\Lambda}}_h \tag{1.51d}$$

where $\mathbb{F}_h^{n+1} = \boldsymbol{\nabla}_{\mathrm{s}}\,\mathbf{X}_h^{n+1}$.

Assuming that $\mathbb{P}(\mathbb{F}) = \kappa\mathbb{F} = \kappa\,\boldsymbol{\nabla}_{\mathrm{s}}\,\mathbf{X}$ and denoting by $\boldsymbol{\varphi}$, ψ, $\boldsymbol{\chi}$ and $\boldsymbol{\zeta}$ the basis functions of $\underline{\mathbf{V}}_h$, Q_h, $\underline{\mathbf{S}}_h$ and $\underline{\boldsymbol{\Lambda}}_h$ respectively, we write Problem 1.4.1 in matrix form as

$$\left[\begin{array}{cc|c|c} \mathsf{A} & \mathsf{B}^\top & 0 & \mathsf{C}_f(\mathbf{X}_h^n)^\top \\ \mathsf{B} & 0 & 0 & 0 \\ \hline 0 & 0 & \mathsf{A}_s & -\mathsf{C}_s^\top \\ \hline \mathsf{C}_f(\mathbf{X}_h^n) & 0 & -\frac{1}{\Delta t}\mathsf{C}_s & 0 \end{array}\right] \left[\begin{array}{c} \mathbf{u}_h^{n+1} \\ p_h^{n+1} \\ \hline \mathbf{X}_h^{n+1} \\ \hline \boldsymbol{\lambda}_h^{n+1} \end{array}\right] = \left[\begin{array}{c} \mathbf{f} \\ \mathbf{0} \\ \hline \mathbf{g} \\ \hline \mathbf{d} \end{array}\right] \tag{1.52}$$

where

$\mathsf{A} = \frac{\rho_f}{\Delta t}\mathsf{M}_f + \mathsf{K}_f$ with $(\mathsf{M}_f)_{ij} = (\boldsymbol{\varphi}_j,\boldsymbol{\varphi}_i)_\Omega$ and $(\mathsf{K}_f)_{ij} = a(\boldsymbol{\varphi}_j,\boldsymbol{\varphi}_i) + b(\mathbf{u}_h^n,\boldsymbol{\varphi}_j,\boldsymbol{\varphi}_i)$

$\mathsf{B}_{ki} = -(\operatorname{div}\boldsymbol{\varphi}_i,\psi_k)_\Omega$

$\mathsf{A}_s = \frac{\delta\rho}{\Delta t^2}\mathsf{M}_s + \mathsf{K}_s$ with $(\mathsf{M}_s)_{ij} = (\boldsymbol{\chi}_j,\boldsymbol{\chi}_i)_\mathcal{B}$ and $(\mathsf{K}_s)_{ij} = \kappa(\boldsymbol{\nabla}_{\mathrm{s}}\,\boldsymbol{\chi}_j,\boldsymbol{\nabla}_{\mathrm{s}}\,\boldsymbol{\chi}_i)_\mathcal{B}$

$(\mathsf{C}_f(\mathbf{X}_h^n))_{\ell j} = \mathbf{c}(\boldsymbol{\zeta}_\ell,\boldsymbol{\varphi}_j(\mathbf{X}_h^n))$

$(\mathsf{C}_s)_{\ell j} = \mathbf{c}(\boldsymbol{\zeta}_\ell,\boldsymbol{\chi}_j)$

$\mathbf{f}_i = \frac{\rho_f}{\Delta t}(\mathsf{M}_f\mathbf{u}_h^n)_i$

$\mathbf{g}_i = \frac{\delta\rho}{\Delta t^2}(\mathsf{M}_s(2\mathbf{X}_h^n - \mathbf{X}_h^{n-1}))_i$

$\mathbf{d}_\ell = -\frac{1}{\Delta t}(\mathsf{C}_s\mathbf{X}_h^n)_\ell$.

Remark 1.4.1. *We observe that when* $\mathbf{c}$ *is defined as the duality pairing between* $\mathbf{H}^1(\mathcal{B})$ *and its dual, as presented in* (1.40)*, provided that* $\underline{\mathbf{\Lambda}}_h \subset \mathbf{L}^2(\mathcal{B})$*, we can identify it as the inner product of* $\mathbf{L}^2(\mathcal{B})$*, hence*

$$\mathbf{c}(\boldsymbol{\mu}_h, \mathbf{Y}_h) = (\boldsymbol{\mu}_h, \mathbf{Y}_h)_{\mathcal{B}} \quad \forall \boldsymbol{\mu}_h \in \underline{\mathbf{\Lambda}}_h, \forall \mathbf{Y}_h \in \underline{\mathbf{S}}_h. \tag{1.53}$$

Conversely, if $\mathbf{c}$ *is the scalar product in* $\mathbf{H}^1(\mathcal{B})$*, no modification is needed.*

1.5 Analysis of the stationary problem

In order to study the well-posedness of our problem, we focus our attention on the associated stationary problem [24, 26]: indeed, as previously mentioned, we can read the time semi-discretized Problem 1.3.2 as a sequence of stationary problems.

For simplicity, let us consider again the case of the linear solid model given by $\mathbb{P}(\mathbb{F}) = \kappa\, \boldsymbol{\nabla}_{\mathrm{s}}\, \mathbf{X}$ and set $\overline{\mathbf{X}} = \mathbf{X}^n$ and $\overline{\mathbf{u}} = \mathbf{u}^n$. Then, we define two new bilinear forms, related to the fluid and the structure equations respectively

$$\begin{aligned} \mathbf{a}_f(\mathbf{u}, \mathbf{v}) &= \alpha(\mathbf{u}, \mathbf{v})_\Omega + a(\mathbf{u}, \mathbf{v}) + b(\overline{\mathbf{u}}, \mathbf{u}, \mathbf{v}) \quad \forall \mathbf{u}, \mathbf{v} \in \mathbf{H}^1_0(\Omega) \\ \mathbf{a}_s(\mathbf{X}, \mathbf{Y}) &= \beta(\mathbf{X}, \mathbf{Y})_{\mathcal{B}} + \gamma(\boldsymbol{\nabla}_{\mathrm{s}}\, \mathbf{X}, \boldsymbol{\nabla}_{\mathrm{s}}\, \mathbf{Y})_{\mathcal{B}} \quad \forall \mathbf{X}, \mathbf{Y} \in \mathbf{H}^1(\mathcal{B}) \end{aligned} \tag{1.54}$$

and we deduce right hand sides and constants from time step and physical coefficients of the model as

$$\begin{aligned} &\mathbf{u} = \mathbf{u}^{n+1},\ p = p^{n+1},\ \mathbf{X} = \frac{\mathbf{X}^{n+1}}{\Delta t},\ \boldsymbol{\lambda} = \boldsymbol{\lambda}^{n+1} \\ &\mathbf{f} = \frac{\rho_f}{\Delta t}\mathbf{u}^n \\ &\mathbf{g} = \frac{\delta\rho}{\Delta t^2}(2\mathbf{X}^n - \mathbf{X}^{n-1}) \\ &\mathbf{d} = -\frac{1}{\Delta t}\mathbf{X}^n \\ &\alpha = \frac{\rho_f}{\Delta t},\ \beta = \frac{\delta\rho}{\Delta t},\ \gamma = \kappa\Delta t. \end{aligned}$$

Therefore, the continuous formulation for the stationary problem reads as follows.

Problem 1.5.1. *Let* $\overline{\mathbf{X}} \in \mathbf{W}^{1,\infty}(\mathcal{B})$ *be invertible with Lipschitz inverse and* $\overline{\mathbf{u}} \in \mathbf{H}^1_0(\Omega)$ *such that* $\operatorname{div} \overline{\mathbf{u}} = 0$*. Given* $\mathbf{f} \in \mathbf{L}^2(\Omega)$*,* $\mathbf{g} \in \mathbf{L}^2(\mathcal{B})$ *and* $\mathbf{d} \in \mathbf{L}^2(\mathcal{B})$*, find* $(\mathbf{u}, p) \in \mathbf{H}^1_0(\Omega) \times \mathrm{L}^2_0(\Omega)$*,* $\mathbf{X} \in \mathbf{H}^1(\mathcal{B})$ *and* $\boldsymbol{\lambda} \in \underline{\mathbf{\Lambda}}$*, such that*

$$\mathbf{a}_f(\mathbf{u}, \mathbf{v}) - (\operatorname{div} \mathbf{v}, p)_\Omega + \mathbf{c}(\boldsymbol{\lambda}, \mathbf{v}(\overline{\mathbf{X}})) = (\mathbf{f}, \mathbf{v})_\Omega \qquad \forall \mathbf{v} \in \mathbf{H}^1_0(\Omega) \tag{1.55a}$$

$$(\operatorname{div}\mathbf{u}, q)_\Omega = 0 \qquad \forall q \in \mathrm{L}^2_0(\Omega) \tag{1.55b}$$

$$\mathbf{a}_s(\mathbf{X}, \mathbf{Y}) - \mathbf{c}(\boldsymbol{\lambda}, \mathbf{Y}) = (\mathbf{g}, \mathbf{Y})_{\mathcal{B}} \qquad \forall \mathbf{Y} \in \mathbf{H}^1(\mathcal{B}) \tag{1.55c}$$

$$\mathbf{c}(\boldsymbol{\mu}, \mathbf{u}(\overline{\mathbf{X}}) - \mathbf{X}) = \mathbf{c}(\boldsymbol{\mu}, \mathbf{d}) \qquad \forall \boldsymbol{\mu} \in \underline{\boldsymbol{\Lambda}} \tag{1.55d}$$

It is easy to see that this formulation presents a saddle point structure; indeed, moving to the operational formulation, we have

$$\left[\begin{array}{ccc|c} \mathsf{A}_f & \mathsf{B}^\top & 0 & \mathsf{C}_f^\top \\ \mathsf{B} & 0 & 0 & 0 \\ 0 & 0 & \mathsf{A}_s & -\mathsf{C}_s^\top \\ \hline \mathsf{C}_f & 0 & -\mathsf{C}_s & 0 \end{array}\right] \left[\begin{array}{c} \mathbf{u} \\ p \\ \mathbf{X} \\ \hline \boldsymbol{\lambda} \end{array}\right] = \left[\begin{array}{c} \mathbf{f} \\ \mathbf{0} \\ \mathbf{g} \\ \hline \mathbf{d} \end{array}\right]. \tag{1.56}$$

As a consequence, its well-posedness can be studied in terms of inf-sup conditions [15]: in order to do that, it is convenient to rearrange the variables, so that we obtain

$$\left[\begin{array}{ccc|c} \mathsf{A}_f & 0 & \mathsf{C}_f^\top & \mathsf{B}^\top \\ 0 & \mathsf{A}_s & -\mathsf{C}_s^\top & 0 \\ \mathsf{C}_f & -\mathsf{C}_s & 0 & 0 \\ \hline \mathsf{B}^\top & 0 & 0 & 0 \end{array}\right] \left[\begin{array}{c} \mathbf{u} \\ \mathbf{X} \\ \boldsymbol{\lambda} \\ \hline p \end{array}\right] = \left[\begin{array}{c} \mathbf{f} \\ \mathbf{g} \\ \mathbf{d} \\ \hline \mathbf{0} \end{array}\right]; \tag{1.57}$$

hence, following the work done in [24, 26], Problem 1.5.1 can be reformulated making use of only two bilinear forms.

Problem 1.5.2. *Let* $\overline{\mathbf{X}} \in \mathbf{W}^{1,\infty}(\mathcal{B})$ *be invertible with Lipschitz inverse and* $\overline{\mathbf{u}} \in \mathbf{H}^1_0(\Omega)$ *such that* $\operatorname{div}\overline{\mathbf{u}} = 0$*; given* $\mathbf{f} \in \mathbf{L}^2(\Omega)$*,* $\mathbf{g} \in \mathbf{L}^2(\mathcal{B})$ *and* $\mathbf{d} \in \mathbf{L}^2(\mathcal{B})$*, find* $(\mathbf{U}, p) \in \boldsymbol{\mathcal{V}} \times \mathrm{L}^2_0(\Omega)$ *such that*

$$\begin{aligned} &\mathscr{A}(\mathbf{U}, \mathbf{V}) + \mathscr{B}(\mathbf{V}, p) = (\mathbf{f}, \mathbf{v})_\Omega + (\mathbf{g}, \mathbf{Y})_{\mathcal{B}} - \mathbf{c}(\boldsymbol{\mu}, \mathbf{d}) && \forall \mathbf{V} \in \boldsymbol{\mathcal{V}} \\ &\mathscr{B}(\mathbf{U}, q) = 0 && \forall q \in \mathrm{L}^2_0(\Omega). \end{aligned} \tag{1.58}$$

In particular, $\boldsymbol{\mathcal{V}}$ denotes the product space $\mathbf{H}^1_0(\Omega) \times \mathbf{H}^1(\mathcal{B}) \times \mathbf{H}^1(\mathcal{B})$ containing all the triplets $\mathbf{V} = (\mathbf{v}, \mathbf{Y}, \boldsymbol{\mu})$ endowed with the norm

$$|||\mathbf{V}||| = \|\mathbf{v}\|_{1,\Omega} + \|\mathbf{Y}\|_{1,\mathcal{B}} + \|\boldsymbol{\mu}\|_{\underline{\boldsymbol{\Lambda}}} \tag{1.59}$$

so that the bilinear forms $\mathscr{A}$ and $\mathscr{B}$ are defined as follows

$$\begin{aligned}
&\mathscr{A} : \boldsymbol{\mathcal{V}} \times \boldsymbol{\mathcal{V}} \to \mathbb{R} \\
&\mathscr{A}(\mathbf{U}, \mathbf{V}) = \mathbf{a}_f(\mathbf{u}, \mathbf{v}) + \mathbf{a}_s(\mathbf{X}, \mathbf{Y}) + \mathbf{c}(\boldsymbol{\lambda}, \mathbf{v}(\overline{\mathbf{X}}) - \mathbf{Y}) - \mathbf{c}(\boldsymbol{\mu}, \mathbf{u}(\overline{\mathbf{X}}) - \mathbf{X}) \\
&\mathscr{B} : \boldsymbol{\mathcal{V}} \times \mathrm{L}^2_0(\Omega) \to \mathbb{R} \\
&\mathscr{B}(\mathbf{V}, q) = (\operatorname{div} \mathbf{v}, q)_\Omega
\end{aligned} \tag{1.60}$$

In particular, the inf-sup conditions for Problem 1.5.1 are equivalent to the conditions for Problem 1.5.2 which have been proved in [24].

Proposition 1.5.1. *There exist two positive constants θ and η such that*

$$\inf_{\mathbf{U} \in \boldsymbol{\mathcal{K}}[\mathscr{B}]} \sup_{\mathbf{V} \in \boldsymbol{\mathcal{K}}[\mathscr{B}]} \frac{\mathscr{A}(\mathbf{U}, \mathbf{V})}{|||\mathbf{U}||| \, |||\mathbf{V}|||} \geq \theta, \qquad \inf_{q \in \mathrm{L}^2_0(\Omega)} \sup_{\mathbf{V} \in \boldsymbol{\mathcal{V}}} \frac{\mathscr{B}(\mathbf{V}, q)}{|||\mathbf{V}||| \, \|q\|_{0,\Omega}} \geq \eta, \tag{1.61}$$

where $\boldsymbol{\mathcal{K}}[\mathscr{B}] = \{\mathbf{V} \in \boldsymbol{\mathcal{V}} : \mathscr{B}(\mathbf{V}, q) = 0 \quad \forall q \in \mathrm{L}^2_0(\Omega)\}$.

Consequently, the solution operator $\mathscr{L} : \boldsymbol{\mathcal{V}} \times \mathrm{L}^2_0(\Omega) \to \boldsymbol{\mathcal{V}}' \times (\mathrm{L}^2_0(\Omega))'$ associated to our problem

$$\mathscr{L}(\mathbf{U}, p) = (\mathbf{f}, \mathbf{g}, \mathbf{d}, \mathbf{0}) \tag{1.62}$$

is an isomorphism.

Once the inf-sup conditions are proved for the continuous problem, we can move to study the well-posedness for the finite element discrete problem: we can proceed as in Section 1.4 and introduce the discrete spaces $\underline{\mathbf{V}}_h$, Q_h defined on the fluid mesh $\mathcal{T}_h^\Omega$ and $\underline{\mathbf{S}}_h$, $\underline{\boldsymbol{\Lambda}}_h$ defined on the solid mesh $\mathcal{T}_h^\mathcal{B}$. In particular, even if the $\underline{\mathbf{S}}_h$ and $\underline{\boldsymbol{\Lambda}}_h$ can be either equal or different to each other, we assume that $\underline{\mathbf{S}}_h = \underline{\boldsymbol{\Lambda}}_h$. In particular, we set them to be made up of piecewise linear elements

$$\underline{\mathbf{S}}_h = \underline{\boldsymbol{\Lambda}}_h = \{\mathbf{Y} \in \mathbf{H}^1(\mathcal{B}) : \mathbf{Y}_{|\mathrm{T}} \in (\mathcal{P}_1(\mathrm{T}))^d \quad \forall \mathrm{T} \in \mathcal{T}_h^\mathcal{B}\}. \tag{1.63}$$

This last assumption is not too restrictive and was already considered in [24], whereas other possible choices for $\underline{\mathbf{S}}_h$ and $\underline{\boldsymbol{\Lambda}}_h$ are discussed in [26, 2].

Therefore, the discrete counterpart of Problem 1.5.1 reads.

Problem 1.5.3. *Let $\overline{\mathbf{X}} \in \mathbf{W}^{1,\infty}(\mathcal{B})$ be invertible with Lipschitz inverse and $\overline{\mathbf{u}} \in \mathbf{H}^1_0(\Omega)$ such that $\operatorname{div} \overline{\mathbf{u}} = 0$. Given $\mathbf{f} \in \mathbf{L}^2(\Omega)$, $\mathbf{g} \in \mathbf{L}^2(\mathcal{B})$ and $\mathbf{d} \in \mathbf{L}^2(\mathcal{B})$, find $(\mathbf{u}_h, p_h) \in \underline{\mathbf{V}}_h \times Q_h$, $\mathbf{X}_h \in \underline{\mathbf{S}}_h$ and $\boldsymbol{\lambda}_h \in \underline{\boldsymbol{\Lambda}}_h$, such that*

$$\mathbf{a}_f(\mathbf{u}_h, \mathbf{v}_h) - (\operatorname{div} \mathbf{v}_h, p_h)_\Omega + \mathbf{c}(\boldsymbol{\lambda}_h, \mathbf{v}_h(\overline{\mathbf{X}})) = (\mathbf{f}, \mathbf{v}_h)_\Omega \qquad \forall \mathbf{v}_h \in \underline{\mathbf{V}}_h \tag{1.64a}$$

$$(\operatorname{div}\mathbf{u}_h, q_h)_\Omega = 0 \qquad \forall q_h \in Q_h \qquad (1.64b)$$

$$\mathbf{a}_s(\mathbf{X}_h, \mathbf{Y}_h) - \mathbf{c}(\boldsymbol{\lambda}_h, \mathbf{Y}_h) = (\mathbf{g}, \mathbf{Y}_h)_{\mathcal{B}} \qquad \forall \mathbf{Y}_h \in \underline{\mathbf{S}}_h \qquad (1.64c)$$

$$\mathbf{c}(\boldsymbol{\mu}_h, \mathbf{u}_h(\overline{\mathbf{X}}) - \mathbf{X}_h) = \mathbf{c}(\boldsymbol{\mu}_h, \mathbf{d}) \qquad \forall \boldsymbol{\mu}_h \in \underline{\boldsymbol{\Lambda}}_h. \qquad (1.64d)$$

Analogously to the continuous case, we now introduce the discrete product space

$$\boldsymbol{\mathcal{V}}_h = \underline{\mathbf{V}}_h \times \underline{\mathbf{S}}_h \times \underline{\boldsymbol{\Lambda}}_h \subset \boldsymbol{\mathcal{V}}$$

associated to the norm $|||\cdot|||$ and with generic element denoted by $\mathbf{V}_h = (\mathbf{v}_h, \mathbf{Y}_h, \boldsymbol{\mu}_h)$. For the proof of the inf-sup conditions we refer again to [24].

Proposition 1.5.2. *There exist two positive constants $\tilde{\theta}$ and $\tilde{\eta}$ such that*

$$\inf_{\mathbf{U}_h \in \boldsymbol{\mathcal{K}}_h[\mathscr{B}]} \sup_{\mathbf{V}_h \in \boldsymbol{\mathcal{K}}_h[\mathscr{B}]} \frac{\mathscr{A}(\mathbf{U}_h, \mathbf{V}_h)}{|||\mathbf{U}_h||| \, |||\mathbf{V}_h|||} \geq \tilde{\theta}, \qquad \inf_{q_h \in Q_h} \sup_{\mathbf{V}_h \in \boldsymbol{\mathcal{V}}_h} \frac{\mathscr{B}(\mathbf{V}_h, q_h)}{|||\mathbf{V}_h||| \, \|q_h\|_{0,\Omega}} \geq \tilde{\eta}, \qquad (1.65)$$

where $\boldsymbol{\mathcal{K}}_h[\mathscr{B}] = \{\mathbf{V}_h \in \boldsymbol{\mathcal{V}}_h : \mathscr{B}(\mathbf{V}_h, q_h) = 0 \quad \forall q_h \in Q_h\}$.

Therefore, the discrete counterpart of the operator $\mathscr{L}$ defined in (1.62)

$$\mathscr{L}_h : \boldsymbol{\mathcal{V}}_h \times Q_h \longrightarrow \boldsymbol{\mathcal{V}}_h' \times Q_h' \quad \text{such that} \quad \mathscr{L}_h(\mathbf{U}_h, p_h) = (\mathbf{f}, \mathbf{g}, \mathbf{d}, \mathbf{0}) \qquad (1.66)$$

is still an isomorphism.

Finally, we state the optimal convergence theorem, which directly follows from the well-posedness.

Theorem 1.5.1. *Let $\underline{\mathbf{V}}_h$ and Q_h satisfy the usual compatibility conditions for the Stokes problem. If $(\mathbf{u}, p, \mathbf{X}, \boldsymbol{\lambda})$ and $(\mathbf{u}_h, p_h, \mathbf{X}_h, \boldsymbol{\lambda}_h)$ denote respectively the solution for the continuous and the discrete problem, the following error estimate holds true*

$$\begin{aligned} &\|\mathbf{u} - \mathbf{u}_h\|_{1,\Omega} + \|p - p_h\|_{0,\Omega} + \|\mathbf{X} - \mathbf{X}_h\|_{1,\mathcal{B}} + \|\boldsymbol{\lambda} - \boldsymbol{\lambda}_h\|_{\underline{\boldsymbol{\Lambda}}} \\ &\quad \leq C\big(\inf_{\mathbf{v} \in \underline{\mathbf{V}}_h} \|\mathbf{u} - \mathbf{v}\|_{1,\Omega} + \inf_{q \in Q_h} \|p - q\|_{0,\Omega} + \inf_{\mathbf{Y} \in \underline{\mathbf{S}}_h} \|\mathbf{X} - \mathbf{Y}\|_{1,\mathcal{B}} + \inf_{\boldsymbol{\mu} \in \underline{\boldsymbol{\Lambda}}_h} \|\boldsymbol{\lambda} - \boldsymbol{\mu}\|_{\underline{\boldsymbol{\Lambda}}} \big). \end{aligned} \qquad (1.67)$$

Chapter 2

The interface matrix

Since our model is based on a coupling method, a crucial role is performed by the interface matrix, which combines the solid kinematics with the fluid dynamics via integration over the solid mesh of both solid and fluid basis functions. The reciprocal position of the two meshes can be taken into account in two different ways: the first one consists in computing the intersection element by element so that we can implement a composite integration rule, whereas the second one avoids this geometric computation and consists in directly integrating on the elements of the solid mesh $\mathcal{T}_h^{\mathcal{B}}$.

A similar investigation has been performed in mortar element framework in the case of codimension one interface problems for second order elliptic equations [80], while the recent work [57] presents an interpolation technique between unfitted grids based on Lagrange extraction in immersed finite elements and isogeometric analysis. Moreover, a comparison of non-matching techniques for the finite element approximation of interface problems has been recently presented [16].

From the computational perspective, the study and development of efficient algorithms for the computation of the intersection between non-matching grids have been discussed in several papers as support tools for the simulation of coupled problems. For instance, in [81], a 3D Nitsche method is efficiently implemented in the cut–FEM setting by Massing, Larson and Logg. In [74], the authors present a `C++` package called MOONoLith (Multipurpose Object Oriented Numerics Library): this library implements an algorithm for the variational transfer of information between meshes based on a monolithic parallel approach. This involves the computation of the intersection of meshes, both in two and three dimensional cases. In [33], efficient algorithms for the intersection of simplicial elements of different

codimensions in a 3D setting are presented for the case of XFEM and mortar methods, while in the work by Farrell and Maddison [54], the implementation of Galerkin projections for conservative interpolation between discrete fields is based on the Sutherland–Hodgman algorithm for clipping polygons. We finally mention that boolean operations between meshes of different codimensions, as well as finite elements methods on unfitted grids, have been just introduced in the latest version of the `deal.II` library [5].

This chapter is subdivided into five sections. The first two sections are based on our work [20]: we describe the algorithms one can adopt in order to assemble the coupling matrix as described above, while in Section 2.2, we present a numerical investigation to understand how the two proposed techniques affect the convergence of our method. In Section 2.3, we discuss a general result on the effect of numerical integration on fluid-structure interaction problems, while in Section 2.4, we present quadrature error estimates for the approximate integration of the coupling term. Finally, in the last section, we discuss the well-posedness of the problem discretized with inexact coupling, proving that the inf-sup conditions are satisfied.

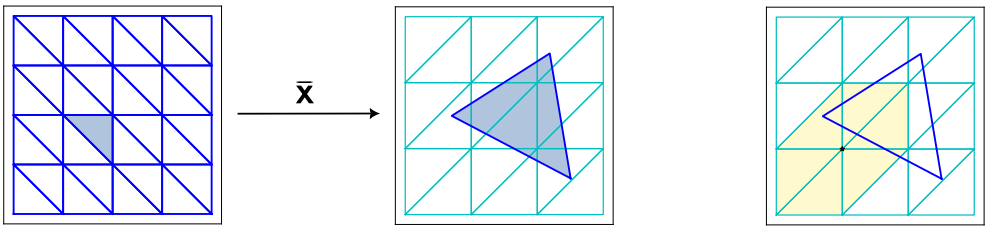
2.1 Assembly techniques

As we discussed in the previous chapters, the use of the fictitious domain approach consists in extending the fluid domain to the region occupied by the immersed solid body; consequently, we have that the solid mesh is superimposed on the fluid one when mapped into the fluid domain.

From the point of view of the equations modeling our interaction system, we have that the coupling between fluid and structure is represented by the bilinear form $\mathbf{c}(\boldsymbol{\mu}_h, \mathbf{v}_h(\overline{\mathbf{X}}))$ associated to the matrix C_f, as described in Sections 1.2 and 1.4.

Since $\mathbf{c}$ can be defined either as the scalar product of $\mathbf{L}^2(\mathcal{B})$ or $\mathbf{H}^1(\mathcal{B})$, it is evident the fact that we have to compute integrals over the solid domain $\mathcal{B}$ involving $\boldsymbol{\mu}_h \in \underline{\boldsymbol{\Lambda}}_h$, defined on $\mathcal{B}$, and $\mathbf{v}_h \in \underline{\mathbf{V}}_h$, that is related to the whole domain Ω: this means that we need to integrate the velocity shape functions, defined with respect to $\mathcal{T}_h^{\Omega}$, over the solid mesh $\mathcal{T}_h^{\mathcal{B}}$, also combined with the map $\overline{\mathbf{X}}$ so that the actual position occupied by the structure, and discretized by $\overline{\mathbf{X}}(\mathcal{T}_h^{\mathcal{B}})$, can be taken into account.

We focus on the case of triangular meshes in two dimensions, but more general situations can be considered, see Section 2.1.3.



(a) Mapping of a solid element into the fluid mesh. (b) Mismatching supports

Figure 2.1: Graphical example of reciprocal positions between fluid and solid elements. On the left, a particular element of the blue solid mesh $\mathcal{T}_h^{\mathcal{B}}$, colored in grey, is immersed in the cyan fluid mesh. On the right, the support of the fluid basis function related to the black node is colored in yellow and only partially matches the mapped solid element.

In Figure 2.1a, in a portion of fluid and solid mesh, we show the action of $\overline{\mathbf{X}}$ applied to a single solid element T_s of $\mathcal{T}_h^{\mathcal{B}}$. Its mapped counterpart $\overline{\mathbf{X}}(\mathrm{T}_s)$ is immersed in the fluid mesh and it is colored in grey. To perform this mapping, we apply $\overline{\mathbf{X}}$ to the nodes, so that we can keep straight the edges connecting them. Moreover, in Figure 2.1b, we present an example of mismatch between the support of a fluid function with respect to an immersed solid element: the support of the considered function, indicated in yellow, only partially matches the blue triangle.

It is important to notice that all these considerations do not affect the assembly of the matrix C_s, defined by $\mathbf{c}(\boldsymbol{\mu}_h, \mathbf{Y}_h)$ since the involved variables are both defined on $\mathcal{T}_h^{\mathcal{B}}$, indeed $\boldsymbol{\mu}_h \in \underline{\boldsymbol{\Lambda}}_h$ and $\mathbf{Y}_h \in \underline{\mathbf{S}}_h$.

After this preliminary discussion, we can now present the discrete version of the available definitions for the bilinear form $\mathbf{c}$. If $\mathbf{c}$ is the inner product in $\mathbf{L}^2(\mathcal{B})$ as discussed in (1.53), we can write

$$\mathbf{c}_h(\boldsymbol{\mu}_h, \mathbf{v}_h(\overline{\mathbf{X}})) = \int_{\mathcal{B}} \boldsymbol{\mu}_h \cdot \mathbf{v}_h(\overline{\mathbf{X}})\, \mathrm{d}\mathbf{s} = \sum_{\mathrm{T}_s \in \mathcal{T}_h^{\mathcal{B}}} \int_{\mathrm{T}_s} \boldsymbol{\mu}_h \cdot \mathbf{v}_h(\overline{\mathbf{X}})\, \mathrm{d}\mathbf{s}, \tag{2.1}$$

while, if $\mathbf{c}$ is the $\mathbf{H}^1(\mathcal{B})$ scalar product as presented in (1.41), we have that

$\mathbf{c}_h(\boldsymbol{\mu}_h, \mathbf{v}_h(\overline{\mathbf{X}}))$ can be written as follows

$$\begin{aligned}\mathbf{c}_h(\boldsymbol{\mu}_h, \mathbf{v}_h(\overline{\mathbf{X}})) &= \int_{\mathcal{B}} \boldsymbol{\mu}_h \cdot \mathbf{v}_h(\overline{\mathbf{X}}) + \boldsymbol{\nabla}_{\mathrm{s}}\, \boldsymbol{\mu}_h : \boldsymbol{\nabla}_{\mathrm{s}}\, \mathbf{v}_h(\overline{\mathbf{X}})\, \mathrm{d}\mathbf{s} \\ &= \sum_{\mathrm{T}_s \in \mathcal{T}_h^{\mathcal{B}}} \int_{\mathrm{T}_s} \boldsymbol{\mu}_h \cdot \mathbf{v}_h(\overline{\mathbf{X}}) + \boldsymbol{\nabla}_{\mathrm{s}}\, \boldsymbol{\mu}_h : \boldsymbol{\nabla}_{\mathrm{s}}\, \mathbf{v}_h(\overline{\mathbf{X}})\, \mathrm{d}\mathbf{s}.\end{aligned} \tag{2.2}$$

We remark that in the last integral, $\boldsymbol{\nabla}_{\mathrm{s}}\, \mathbf{v}_h(\overline{\mathbf{X}})$ represents a composite gradient, hence

$$\boldsymbol{\nabla}_{\mathrm{s}}\, \mathbf{v}_h(\overline{\mathbf{X}}) = \boldsymbol{\nabla}\, \mathbf{v}_h : \boldsymbol{\nabla}_{\mathrm{s}}\, \overline{\mathbf{X}}.$$

In (2.1) and (2.2), we reduced the global definitions to each triangle $\mathrm{T}_s \in \mathcal{T}_h^{\mathcal{B}}$. In the following, given the integral on a generic $\mathrm{T}_s \in \mathcal{T}_h^{\mathcal{B}}$, we present two possible assembly techniques: in order to fully take into account that a velocity function is integrated over a solid element, one can first compute the intersection between the fluid mesh $\mathcal{T}_h^{\Omega}$ and the solid one mapped into Ω_s, or, alternatively, one can directly integrate over $\mathrm{T}_s \in \mathcal{T}_h^{\mathcal{B}}$, without additional procedures. The integrals in (2.1) and (2.2) will be expanded from these two perspectives.

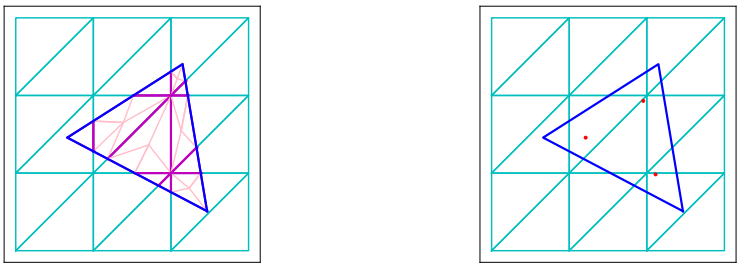
The area of a generic triangle T will be denoted by $|\mathrm{T}|$. We denote by $\{(\mathbf{p}_k^0, \omega_k^0)\}_{k=1}^{K_0}$ quadrature nodes and weights for the numerical computation of the $\mathbf{L}^2(\mathcal{B})$ scalar product, while $\{(\mathbf{p}_k^1, \omega_k^1)\}_{k=1}^{K_1}$ represent the quadrature rule we use for computing the $\mathbf{L}^2(\mathcal{B})$ scalar product of gradients.

2.1.1 Assembly with mesh intersection

The first technique we present allows us to compute *exactly* the integrals in (2.1) and (2.2) since we can make use of a composite rule over T_s, based on the intersection of the fluid and solid meshes.

Computing the intersection, we get a finer triangulation for the structure, characterized by the fact that each element is contained in a single fluid element $\mathrm{T}_f \in \mathcal{T}_h^{\Omega}$ so that the related velocity basis functions are completely supported. In order to get this type of partition for $\mathrm{T}_s \in \mathcal{T}_h^{\mathcal{B}}$, we intersect its mapped counterpart $\overline{\mathbf{X}}(\mathrm{T}_s)$ with each $\mathrm{T}_f \in \mathcal{T}_h^{\Omega}$, therefore T_s can be seen as the union of disjoint polygons $P_1, \ldots, P_J$

$$\mathrm{T}_s = \bigcup_{j=1}^{J} P_j; \tag{2.3}$$



(a) With mesh intersection (b) Without mesh intersection

Figure 2.2: A graphical example for the two approaches proposed for the assembly of the interface matrix. The velocity mesh is colored in cyan, while in blue we have the immersed counterpart of the solid element under consideration. On the left, the mapped solid triangle is partitioned into sub-polygons (purple lines) accordingly with its position with respect to the fluid mesh. If a polygon is not already a triangle, it is triangulated (pink lines). On the right, mapped nodes for a Gauss quadrature rule with order two are represented in the immersed solid element: notice that they belong to different fluid triangles.

in particular, if a polygon P_j is already a triangle, we integrate over it, otherwise, we simply partition it into triangles $\mathrm{T}_1, \ldots, \mathrm{T}_{I_j}$ connecting each vertex with the barycenter, hence

$$\mathrm{T}_s = \bigcup_{j=1}^{J} P_j = \bigcup_{j=1}^{J} \bigcup_{i=1}^{I_j} \mathrm{T}_i. \tag{2.4}$$

In Figure 2.2a, we schematically represent this procedure on a single mapped solid triangle (in blue) which is partitioned first into polygons (in purple) and then into smaller triangles (in pink).

As a consequence, the scalar product in $\mathbf{L}^2(\mathcal{B})$ can be numerically computed as follows

$$\begin{aligned} \int_{\mathrm{T}_s} \boldsymbol{\mu}_h \cdot \mathbf{v}_h(\overline{\mathbf{X}})\, \mathrm{d}\mathbf{s} &= \sum_{j=1}^{J} \int_{P_j} \boldsymbol{\mu}_h \cdot \mathbf{v}_h(\overline{\mathbf{X}})\, \mathrm{d}\mathbf{s} = \sum_{j=1}^{J} \sum_{i=1}^{I_j} \int_{\mathrm{T}_i} \boldsymbol{\mu}_h \cdot \mathbf{v}_h(\overline{\mathbf{X}})\, \mathrm{d}\mathbf{s} \\ &= \sum_{j=1}^{J} \sum_{i=1}^{I_j} |\mathrm{T}_i| \sum_{k=1}^{K_0} \omega_k^0\, \boldsymbol{\mu}_h(\mathbf{p}_k^0) \cdot \mathbf{v}_h(\overline{\mathbf{X}}(\mathbf{p}_k^0)), \end{aligned} \tag{2.5}$$

while, for the scalar product of gradients we have

$$\begin{aligned}\int_{\mathrm{T}_s} \boldsymbol{\nabla}_{\mathrm{s}}\, \boldsymbol{\mu}_h : \boldsymbol{\nabla}_{\mathrm{s}}\, \mathbf{v}_h(\overline{\mathbf{X}})\, \mathrm{d}\mathbf{s} &= \sum_{j=1}^{J} \int_{P_j} \boldsymbol{\nabla}_{\mathrm{s}}\, \boldsymbol{\mu}_h : \boldsymbol{\nabla}_{\mathrm{s}}\, \mathbf{v}_h(\overline{\mathbf{X}})\, \mathrm{d}\mathbf{s} \\ &= \sum_{j=1}^{J} \sum_{i=1}^{I_j} \int_{\mathrm{T}_i} \boldsymbol{\nabla}_{\mathrm{s}}\, \boldsymbol{\mu}_h : \boldsymbol{\nabla}_{\mathrm{s}}\, \mathbf{v}_h(\overline{\mathbf{X}})\, \mathrm{d}\mathbf{s} \\ &= \sum_{j=1}^{J} \sum_{i=1}^{I_j} |\mathrm{T}_i| \sum_{k=1}^{K_1} \omega_k^1\, \boldsymbol{\nabla}_{\mathrm{s}}\, \boldsymbol{\mu}_h(\mathbf{p}_k^1) : \boldsymbol{\nabla}_{\mathrm{s}}\, \mathbf{v}_h(\overline{\mathbf{X}}(\mathbf{p}_k^1)).\end{aligned} \tag{2.6}$$

The procedure is summarized in Algorithm 2.1.

Algorithm 2.1 Assembly C_f with mesh intersection

Data:

$\{\mathrm{T}_f\}_{f=1,\dots,N_F}$: elements of the fluid mesh $\mathcal{T}_h^{\Omega}$

$\{\mathrm{T}_s\}_{s=1,\dots,N_S}$: elements of the reference solid mesh $\mathcal{T}_h^{\mathcal{B}}$

Compute mesh intersection $\{\mathrm{T}_f\}_f \cap \{\overline{\mathbf{X}}(\mathrm{T}_s)\}_s$:

each $\overline{\mathbf{X}}(\mathrm{T}_s)$ is partitioned into $J \geq 1$ polygons $\{P_j\}_{j=1,\dots,J}$

associated with J fluid elements $\{\mathrm{T}_{f,j}\}_{j=1,\dots,J}$, i.e. $P_j \leftrightarrow \mathrm{T}_{f,j}$

for $\{\mathrm{T}_s\}_{s=1,\dots,N_S}$ **do**
  for $\{P_j\}_{j=1,\dots,J}$ **do**
    if P_j *is not a triangle* **then**
      Triangulate P_j into $\{\mathrm{T}_i\}_{i=1,\dots,I_j}$
    end
    else
      $I_j = 1$ and $\mathrm{T}_1 = P_j$
    end
    for $\{\mathrm{T}_i\}_{i=1,\dots,I_j}$ **do**
      Integrate using the basis functions related to T_s and $\mathrm{T}_{f,j}$
      Load contribution to the global matrix
    end
  end
end

2.1.2 Assembly without mesh intersection

This second possibility is cheaper than the previous one because skips all the geometric computations related to the intersection of the two meshes. Therefore, since we directly integrate on the elements of $\mathcal{T}_h^{\mathcal{B}}$, we can compute the quadrature nodes in the element T_s we are considering, accordingly to a certain quadrature rule.

Obviously, the evaluation of the solid shape functions does not require particular attention. Conversely, the evaluation of the velocity functions is delicate since the quadrature nodes could be placed in different fluid elements, hence we need to check in which fluid triangle each node is located. This operation, in the triangular case, can be easily carried out making use of barycentric coordinates, since they are a powerful tool for this type of computations. Indeed, the barycentric coordinates are a local coordinate system related to a simplex (in 2D, a triangle) so that a point can be represented using its distances from the three edges: if a point is in the interior or on the boundary of a triangle, its local coordinates are non-negative.

Once this procedure is done, at each mapped quadrature node we can evaluate the fluid functions related to the element containing it and, finally, apply the quadrature formula.

In Figure 2.2b we report an example of local coupling without mesh intersection based on a Gaussian rule with three nodes. We can see that the three points are in different fluid elements. Moreover, notice that the solid element interacts also with fluid triangles that we are not considering. It is evident that this procedure is *inexact* and may introduce an additional source of error.

Therefore, with this approach, the numerical version of the $\mathbf{L}^2(\mathcal{B})$ scalar product is given by

$$\int_{\mathrm{T}_s} \boldsymbol{\mu}_h \cdot \mathbf{v}_h(\overline{\mathbf{X}})\,\mathrm{d}\mathbf{s} \approx |\mathrm{T}_s| \sum_{k=1}^{K_0} \omega_k^0\, \boldsymbol{\mu}_h(\mathbf{p}_k^0) \cdot \mathbf{v}_h(\overline{\mathbf{X}}(\mathbf{p}_k^0)); \tag{2.7}$$

while for the term involving gradients, we get

$$\int_{\mathrm{T}_s} \boldsymbol{\nabla}_{\mathrm{s}}\, \boldsymbol{\mu}_h : \boldsymbol{\nabla}_{\mathrm{s}}\, \mathbf{v}_h(\overline{\mathbf{X}})\,\mathrm{d}\mathbf{s} \approx |\mathrm{T}_s| \sum_{k=1}^{K_1} \omega_k^1\, \boldsymbol{\nabla}_{\mathrm{s}}\, \boldsymbol{\mu}_h(\mathbf{p}_k^1) : \boldsymbol{\nabla}_{\mathrm{s}}\, \mathbf{v}_h(\overline{\mathbf{X}}(\mathbf{p}_k^1)). \tag{2.8}$$

This procedure is summarized in Algorithm 2.2.

Algorithm 2.2 Assembly the C_f matrix block without mesh intersection

Data:
$\{\mathrm{T}_f\}_{f=1,\dots,N_F}$: elements of the fluid mesh $\mathcal{T}_h^{\Omega}$
$\{\mathrm{T}_s\}_{s=1,\dots,N_S}$: elements of the reference solid mesh $\mathcal{T}_h^{\mathcal{B}}$

for $\{\mathrm{T}_s\}_{s=1,\dots,N_S}$ **do**
  Compute quadrature nodes $\mathbf{p}_1, \dots, \mathbf{p}_K$ in T_s for the rules under consideration
  Evaluate solid basis functions
  Find the fluid element containing each mapped quadrature point:
  i.e. $\overline{\mathbf{X}}(\mathbf{p}_k) \leftrightarrow \mathrm{T}_{f,k}$ with $k = 1, \dots, K$
  for $\{\mathrm{T}_{f,k}\}_{k=1,\dots,K}$ **do**
    Evaluate the fluid basis functions in $\overline{\mathbf{X}}(\mathbf{p}_k)$ and integrate over T_s
    Load contribution to the global matrix
  end
end

2.1.3 Generalization

We have just described the possible techniques one can adopt to assemble the interface matrix in the particular case of two dimensional triangular meshes.

Since for the finite element method different types of meshes are allowed, such as quadrilateral meshes, it is important to point out that our method still works and the two algorithms can be easily adapted to more general cases.

Also the extension to the three dimensional case is possible, obviously with an increase of computational costs due to the complexity of the geometry of tetrahedra and polyhedra. The computation of the intersection of three dimensional elements is non-trivial, but several specific packages can be used for additional support. For the assembly without mesh intersection, the use of barycentric coordinates is still a valid technique in the case of tetrahedral meshes. An example of tetrahedral element immersed in a fluid mesh is depicted in Figure 2.3.

2.2 A numerical investigation

At this point, we present a first series of numerical tests in order to discuss how the use of the two approaches just presented affects the convergence of the numerical

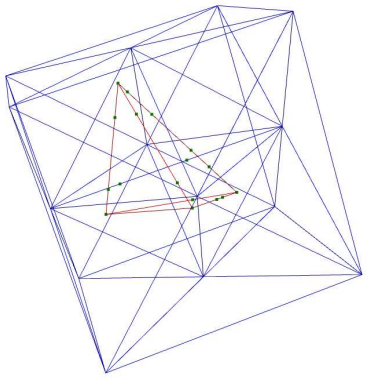


Figure 2.3: A mapped solid tetrahedron (red) is immersed in a portion of fluid tetrahedral mesh (blue). The green points denote the intersection between the edges of the red element with the faces of the blue tetrahedra, and vice versa. It is clear that the polyhedra resulting from the intersection are characterized by a complex geometry.

method. In particular, we are going to solve a two dimensional stationary problem with triangular finite elements.

In the following sections, we introduce the considered model problem and examine all the details related to the implementation of our Python 3.8 script written specifically for the purpose.

2.2.1 Model problem

Let us consider a simplified version of the stationary problem presented in the previous chapter, Problem 1.5.1.

Problem 2.2.1. *Let* $\overline{\mathbf{X}} \in \mathbf{W}^{1,\infty}(\mathcal{B})$ *be invertible with Lipschitz inverse. Given* $\mathbf{f} \in \mathbf{L}^2(\Omega)$, $\mathbf{g} \in \mathbf{L}^2(\mathcal{B})$ *and* $\mathbf{d} \in \mathbf{L}^2(\mathcal{B})$, *find* $(\mathbf{u}, p) \in \mathbf{H}^1_0(\Omega) \times \mathrm{L}^2_0(\Omega)$, $\mathbf{X} \in \mathbf{H}^1(\mathcal{B})$ *and* $\boldsymbol{\lambda} \in \mathbf{H}^1(\mathcal{B})$, *such that*

$$\mathbf{a}_f(\mathbf{u}, \mathbf{v}) - (\operatorname{div} \mathbf{v}, p)_\Omega + \mathbf{c}(\boldsymbol{\lambda}, \mathbf{v}(\overline{\mathbf{X}})) = (\mathbf{f}, \mathbf{v})_\Omega \qquad \forall \mathbf{v} \in \mathbf{H}^1_0(\Omega) \tag{2.9a}$$

$$(\operatorname{div} \mathbf{u}, q)_\Omega = 0 \qquad \forall q \in \mathrm{L}^2_0(\Omega) \tag{2.9b}$$

$$\mathbf{a}_s(\mathbf{X}, \mathbf{Y}) - \mathbf{c}(\boldsymbol{\lambda}, \mathbf{Y}) = (\mathbf{g}, \mathbf{Y})_\mathcal{B} \qquad \forall \mathbf{Y} \in \mathbf{H}^1(\mathcal{B}) \tag{2.9c}$$

$$\mathbf{c}(\boldsymbol{\mu}, \mathbf{u}(\overline{\mathbf{X}}) - \mathbf{X}) = \mathbf{c}(\boldsymbol{\mu}, \mathbf{d}) \qquad \forall \boldsymbol{\mu} \in \mathbf{H}^1(\mathcal{B}). \tag{2.9d}$$

In this case, we assume that the bilinear forms involved in the variational formulation are defined as follows

$$\mathbf{a}_f(\mathbf{u}, \mathbf{v}) = (\boldsymbol{\nabla}\, \mathbf{u}, \boldsymbol{\nabla}\, \mathbf{v})_\Omega$$
$$\mathbf{a}_s(\mathbf{X}, \mathbf{Y}) = (\boldsymbol{\nabla}_{\mathrm{s}}\, \mathbf{X}, \boldsymbol{\nabla}_{\mathrm{s}}\, \mathbf{Y})_{\mathcal{B}},$$

while for the coupling term we consider the scalar product in $\mathbf{H}^1(\mathcal{B})$

$$\mathbf{c}(\boldsymbol{\mu}, \mathbf{Y}) = (\boldsymbol{\mu}, \mathbf{Y})_{\mathcal{B}} + (\boldsymbol{\nabla}_{\mathrm{s}}\, \boldsymbol{\mu}, \boldsymbol{\nabla}_{\mathrm{s}}\, \mathbf{Y})_{\mathcal{B}}.$$

The solutions of our tests are chosen in advance so that they are known in their analytical form and do not belong to the finite element spaces chosen for the approximation; hence the right hand sides are accordingly computed:

$$\mathbf{f} = -\boldsymbol{\Delta}\mathbf{u} + \nabla\, p + \text{“}\mathbf{c}(\boldsymbol{\lambda}, \mathbf{v})\text{”}$$
$$\mathbf{g} = -\boldsymbol{\Delta}\mathbf{X} - \boldsymbol{\lambda} + \mathbf{g}_{\partial\mathcal{B}}$$
$$\mathbf{d} = \mathbf{u}(\overline{\mathbf{X}}) - \mathbf{X},$$

where the term $\mathbf{c}(\boldsymbol{\lambda}, \mathbf{v})$ in $\mathbf{f}$ is imposed only variationally and only in the region occupied by the solid body Ω^s; for this reason, the assembly of this particular term is performed by making use of the mesh intersection in order to get an exact computation.

Moreover, $\mathbf{g}_{\partial\mathcal{B}}$ denotes the boundary integral we obtain integrating by parts the equation of the solid, ignoring the multiplier term, indeed we have

$$\mathbf{a}_s(\mathbf{X}, \mathbf{Y}) = (\boldsymbol{\nabla}_{\mathrm{s}}\, \mathbf{X}, \boldsymbol{\nabla}_{\mathrm{s}}\, \mathbf{Y})_{\mathcal{B}} = (-\boldsymbol{\Delta}\mathbf{X}, \mathbf{Y})_{\mathcal{B}} + (\boldsymbol{\nabla}_{\mathrm{s}}\, \mathbf{X} \cdot \mathbf{n}_s, \mathbf{Y})_{\partial\mathcal{B}}.$$

We are going to present eight different numerical tests performed varying the mutual position between fluid and solid, using trivial and non-trivial choices of $\overline{\mathbf{X}}$ and both continuous and discontinuous pressure functions. In particular, for the first six tests, we assume that $\overline{\mathbf{X}}$ is the identity, implying that reference and actual solid domain coincide, $\mathcal{B} = \Omega^s$. This assumption is removed in the last two tests, where different domains are chosen. Furthermore, in the fifth and sixth tests, we analyze the behavior of our solver in the case of discontinuous pressure both in the case in which the discontinuity coincides with the fluid-structure interface or not.

In terms of finite element spaces, we work with the Bercovier–Pironneau element and its enhanced version for velocity and pressure, while we choose piecewise linear spaces for the variables defined on $\mathcal{B}$. The main features and the derivation of this Stokes pair from the classical Hood–Taylor element are reviewed in the next section.

2.2.2 Finite element spaces

One of the most used Stokes pairs is the lowest order Hood–Taylor element $\mathcal{P}_2/\mathcal{P}_1$ [15] characterized by the use of piecewise quadratic polynomials for the velocity and piecewise linear polynomials for the pressure:

$$\begin{aligned} \underline{\mathbf{V}}_{HT} &= \{\mathbf{v} \in \mathbf{H}^1(\mathcal{B}) : \mathbf{v}_{|\mathrm{T}} \in (\mathcal{P}_2(\mathrm{T}))^2 \quad \forall \mathrm{T} \in \mathcal{T}_h^\Omega\} \\ Q_{HT} &= \{q \in \mathrm{H}^1(\Omega) \cap \mathrm{L}_0^2(\Omega) : q_{|\mathrm{T}} \in \mathcal{P}_1(\mathrm{T}) \quad \forall \mathrm{T} \in \mathcal{T}_h^\Omega\}. \end{aligned} \tag{2.10}$$

With this choice, the local degrees of freedom for the velocity are six, vertices and midpoints, while for the pressure only the vertices are considered.

Since this setting implies the implementation of quadratic polynomials, a popular alternative often used in this field of study and engineering simulations is the Bercovier–Pironneau element $\mathcal{P}_1 - iso - \mathcal{P}_2/\mathcal{P}_1$ [13, 61]: this is intended as the cheaper version of (2.10) since they share the same set of degrees of freedom, but making use of piecewise linear functions also for the approximation of the velocity. The drawback is that we theoretically lose one order in the rate of convergence: this is actually true for the velocity, whereas the pressure error superconverges [17]. This phenomenon has been observed and studied for an another low order Stokes element, the MINI, both from the theoretical [53] and experimental [42, 43] point of view.

In order to combine the six local degrees of freedom of $\underline{\mathbf{V}}_h$ in (2.10) with the use of piecewise linear polynomials, we need to work with two different meshes for velocity and pressure: we can see each pressure element as a macroelement containing four velocity triangles, obtained connecting the midpoints. As a consequence, we have that if $\mathcal{T}_h^\Omega$ is the space discretization we use for the pressure, the velocity mesh is $\mathcal{T}_{h/2}^\Omega$. Finally, $(\underline{\mathbf{V}}_h, Q_h)$ is defined as follows

$$\begin{aligned} \underline{\mathbf{V}}_h &= \{\mathbf{v} \in \mathbf{H}_0^1(\Omega) : \mathbf{v}_{|\mathrm{T}} \in (\mathcal{P}_1(\mathrm{T}))^2 \quad \forall \mathrm{T} \in \mathcal{T}_{h/2}^\Omega\} \\ Q_h &= \{q \in \mathrm{H}^1(\Omega) \cap \mathrm{L}_0^2(\Omega) : q_{|\mathrm{T}} \in \mathcal{P}_1(\mathrm{T}) \quad \forall \mathrm{T} \in \mathcal{T}_h^\Omega\}. \end{aligned} \tag{2.11}$$

An enhanced version of (2.11) was introduced in [17]: since discontinuous pressure schemes satisfy a local mass conservation property (which means that we have null average divergence element by element), piecewise constant functions were added to Q_h:

$$\begin{aligned} Q_h = \{q \in \mathrm{L}_0^2(\Omega) : & q = q_1 + q_0, \\ & q_1 \in H^1(\Omega),\ q_{1|\mathrm{T}} \in \mathcal{P}_1(\mathrm{T}), \\ & q_{0|\mathrm{T}} \in \mathcal{P}_0(\mathrm{T}) \qquad \forall \mathrm{T} \in \mathcal{T}_h^\Omega\}. \end{aligned} \tag{2.12}$$

This modified element is also called $\mathcal{P}_1 - iso - \mathcal{P}_2/\mathcal{P}_1 + \mathcal{P}_0$ and the inf-sup conditions were proved making use of the macroelement technique under the assumption that each pressure element has at least one internal node. With the enhancement, we loose the superconvergence in pressure. Moreover, it is important to notice that this enhancement is not possible in the three dimensional case since there are no degrees of freedom in the interior of the faces.

In FSI applications it is very common that the pressure function presents a discontinuity along the interface. When this coincides with the fluid mesh, the use of discontinuous pressure elements produces an improvement with respect to the case of continuous finite elements: indeed, in the first case, the convergence rate remains optimal, since no oscillations originate (Figure 2.4a); conversely, in the second case, the error presents the so-called Gibbs phenomenon (Figure 2.4b). No improvement is obtained when the discontinuity does not coincide with the fluid mesh.

In our case, the use of two different meshes for velocity and pressure affects the assembly procedure of the interface matrix C_f since it is defined from $\mathbf{c}(\boldsymbol{\lambda}_h, \mathbf{v}_h(\overline{\mathbf{X}}))$: this implies that we have to couple a variable defined on $\mathcal{T}_h^{\mathcal{B}}$ with a variable defined on $\mathcal{T}_{h/2}^{\Omega}$.

For the spaces related to the structure variables, we choose piecewise linear polynomials as we did in Section 1.5

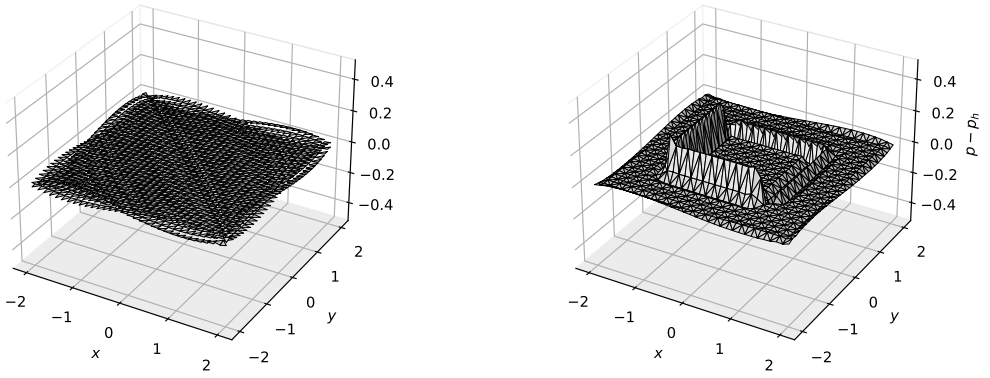
$$\underline{\mathbf{S}}_h = \underline{\boldsymbol{\Lambda}}_h = \{\mathbf{Y} \in \mathbf{H}^1(\mathcal{B}) : \mathbf{Y}_{|\mathrm{T}} \in (\mathcal{P}_1(\mathrm{T}))^2 \quad \forall \mathrm{T} \in \mathcal{T}_h^{\mathcal{B}}\}. \tag{2.13}$$

2.2.3 Mesh generation

The generation of the meshes is done on the unit square $[0,1]^2$ and then the nodes are mapped to Ω and $\mathcal{B}$.

The discretization of the fluid domain Ω is chosen to be uniform, while for the solid domain $\mathcal{B}$ we work with both uniform and unstructured grids; in particular, in the last case, the generation of the mesh is done making use of the `Gmsh` finite element mesh generator [58].

All the involved meshes satisfy the shape regularity property. When we work with the enhanced pair $\mathcal{P}_1 - iso - \mathcal{P}_2/\mathcal{P}_1 + \mathcal{P}_0$, we need to pay attention to the corner triangles so that they do not violate the stability assumption that at least one node has to be in the interior of the domain: if needed, we perform diagonal



(a) $\mathcal{P}_1 + \mathcal{P}_0$ discontinuous pressure

(b) $\mathcal{P}_1$ discontinuous pressure

Figure 2.4: On the left, error $p - p_h$ of a discontinuous pressure approximated with discontinuous finite elements. On the right, the error committed when continuous elements are used: the region of the jump is characterized by the oscillations of the Gibbs phenomenon

exchange to avoid wrong approximations and instabilities of the pressure on the boundary.

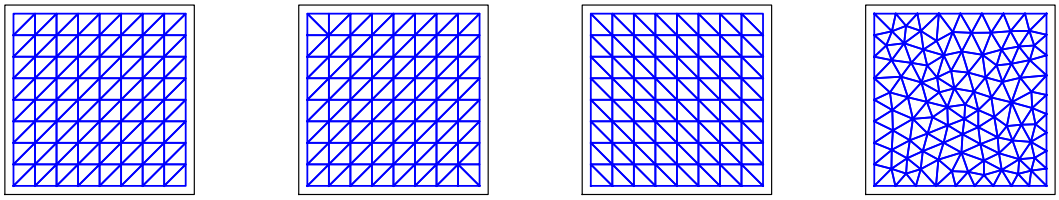
In Table 2.1, we report the number of degrees of freedom related to each unknown in dependence of the chosen finite element spaces and spatial discretizations.

In Figure 2.5, we show some examples of meshes used to discretize Ω and $\mathcal{B}$.

2.2.4 Mesh intersection

The computation of the intersection between two meshes is an expensive procedure, especially when fine meshes are used.

In particular, if N_F and N_S denote the number of fluid and solid elements, respectively, as done in Algorithms 2.1 and 2.2, then the number of pairs of elements we have to test for computing the intersection is $N_F \times N_S$; for this reason, a first way to reduce computational and time costs avoiding testing a pair of disjoint triangles is the implementation of the bounding box technique provided by the `Rtree` Python package. In this way, one predicts in advance if a pair has non-empty intersection checking the mutual position of the two boxes: if the boxes are disjoint, then the intersection between the two elements is empty and we rapidly move to analyze an another pair of elements.



(a) Right (b) Right with correction (c) Left (d) Unstructured

Figure 2.5: Some examples of meshes. (a) and (b) are uniform meshes used for Ω and, in particular, in (b) we can observe the diagonal exchange for correcting the corners elements for the $\mathcal{P}_1 + \mathcal{P}_0$ pressure approximation. The meshes in (c) and (d) are used to discretize $\mathcal{B}$.

DOFs $\mathbf{u}_h$	DOFs p_h	DOFs p_h^0	DOFs $\mathbf{X}_h$, $\boldsymbol{\lambda}_h$	DOFs $\mathbf{X}_h^{uns}$, $\boldsymbol{\lambda}_h^{uns}$
2,178	289	801	162	232
8,450	1,089	3,137	578	742
33,282	4,225	12,417	2,178	2,788
132,028	16,641	49,409	8,450	11,018
526,338	66,049	197,121	33,282	43,734
2,101,250	263,169	787,457	132,028	174,316

Table 2.1: Number of degrees of freedom for each variable in dependence of mesh and finite element space. The last column is related to the use of unstructured meshes for the discretization of the solid domain $\mathcal{B}$, while the other cases refer to uniform meshes. The notation p_h^0 is reserved for the pressure approximation by $\mathcal{P}_1 + \mathcal{P}_0$ elements.

For the actual computation of the intersection of two elements, the `Shapely` package provides the module `geometry` containing tools to represent geometric objects such as segments, lines and polygons and also to manage boolean operations between them. Indeed, in order to compute the intersection of two triangles, we represent them using the data structure `polygon` and then we compute the intersection with the `intersection()` method. The computation of the triangulation of the resulting polygon by connecting vertices and barycenter can be easily performed.

On the other hand, when we choose to assemble the interface matrix C_f without computing the intersection of the two meshes, we do not need particular tools since we can easily use the barycentric coordinate system as described in Section 2.1.2.

In Figure 2.6 we report the geometric configurations of our tests: to fix ideas, we represent both the solid and fluid discretizations with coarse meshes.

2.2.5 Quadrature rules for the interface matrix

Since we are using finite element spaces made of piecewise linear polynomials, it is easy to find quadrature rules allowing us to integrate exactly the quantities involved in the definitions of A_f, A_s, B and C_s. Here, we focus our attention on the quadrature rules we use for assembling the interface matrix C_f with the two techniques under consideration.

We are considering the definition of $\mathbf{c}(\boldsymbol{\mu}_h, \mathbf{v}_h(\overline{\mathbf{X}}))$ as the scalar product in $\mathbf{H}^1(\mathcal{B})$, hence we have to integrate both the product of the two functions $(\boldsymbol{\mu}_h, \mathbf{v}_h(\overline{\mathbf{X}}))_{\mathcal{B}}$ and the product of the gradients $(\boldsymbol{\nabla}_{\mathrm{s}}\,\boldsymbol{\mu}_h, \boldsymbol{\nabla}_{\mathrm{s}}\,\mathbf{v}_h(\overline{\mathbf{X}}))_{\mathcal{B}}$: as a consequence of the chosen finite element spaces for the velocity and the Lagrange multiplier, this means that for the first term we need to use a quadrature formula exact for quadratic polynomials and, for the second case, a rule with first order precision.

We explained before that the computation of the mesh intersection allows us to integrate on solid triangles which are uniquely included in a fluid element, hence applying the following Gaussian quadrature rules we get exact integrals for C_f.

Definition 2.2.1 (Gauss rule, order 1)**.** *A polynomial f of degree one can be integrated exactly by the following one-point formula*

$$\int_{\mathrm{T}} f(\mathbf{x})\, d\mathbf{x} \approx |\mathrm{T}|\, f(\mathbf{x}_{\mathrm{T}}) \tag{2.14}$$

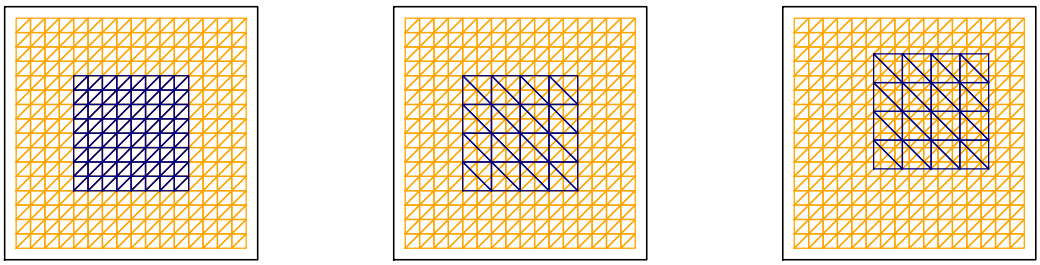
where $\mathbf{x}_{\mathrm{T}}$ is the barycenter of T*.*

Definition 2.2.2 (Gauss rule, order 2)**.** *A polynomial f of degree two can be integrated exactly by the following three-points formula*

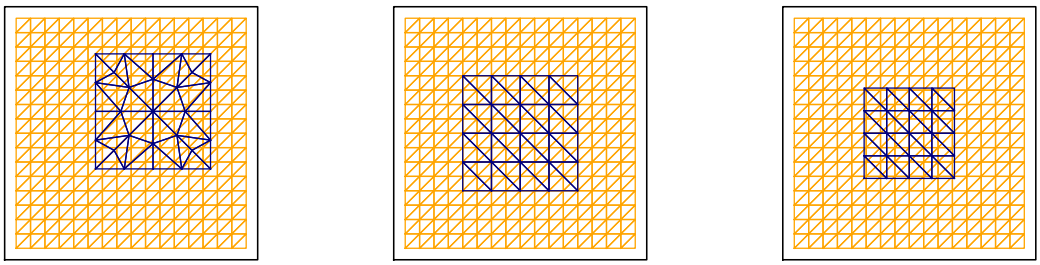
$$\int_{\mathrm{T}} f(\mathbf{x})\, d\mathbf{x} \approx \frac{|\mathrm{T}|}{3} \sum_{k=1}^{3} f\big(\mathbf{x}^{(k)}\big) \tag{2.15}$$

where the quadrature nodes, represented in barycentric coordinates, are

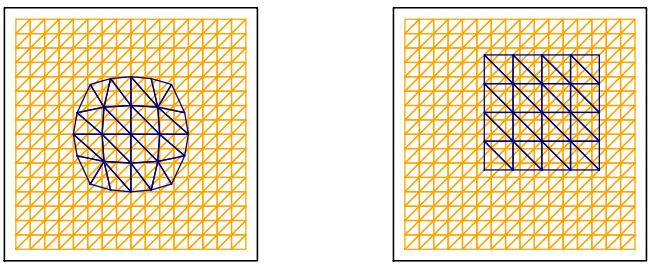
$$\mathbf{x}^{(1)} = (2/3, 1/6, 1/6)$$



(a) Test 1 (b) Test 2 (c) Test 3



(d) Test 4 (e) Test 5 (f) Test 6



(g) Test 7 (h) Test 8

Figure 2.6: The mutual position between fluid and solid body of our tests represented with coarse meshes; each configuration is kept unchanged when the the discretizations are refined. We recall that in the first six cases the reference domain $\mathcal{B}$ coincides with the actual position of the structure. Moreover, due to our choice of Stokes element, it is important to specify that the yellow mesh is the one related to the velocity.

and its permutations.

In the case of assembly without mesh intersection, we implement the formula in Equation (2.15) for both the $\mathbf{L}^2(\mathcal{B})$ and the $\mathbf{H}^1(\mathcal{B})$ contributions. Moreover, in order to collect more information about the behavior of our solver when based on this approach, we also assembled C_f using a Gauss quadrature rule with third order precision.

Definition 2.2.3 (Gauss rule, order 3)**.** *A polynomial f of degree three can be integrated exactly by the following four-points formula*

$$\int_{\mathrm{T}} f(\mathbf{x})\, d\mathbf{x} \approx |\mathrm{T}| \left(\frac{25}{48} \sum_{k=1}^{3} f\left(\mathbf{x}^{(k)}\right) - \frac{9}{16} f(\mathbf{x}_{\mathrm{T}}) \right) \tag{2.16}$$

where the quadrature nodes, represented in barycentric coordinates, are

$$\mathbf{x}^{(1)} = (3/5, 1/5, 1/5)$$

with its permutations and the barycenter $\mathbf{x}_{\mathrm{T}}$ of T.

2.2.6 Numerical results

After having listed and described the main features of the numerical method we have implemented for our study, we now move to the description of tests and related results.

We are going to show that the optimal convergence rate is reached only when the coupling matrix C_f is assembled computing the intersection between fluid and solid meshes; moreover, the method without mesh intersection does not improve its performance when we increase the precision of the quadrature rule.

Test 1

The first test is a sort of sanity check since we want to show that, in the particular situation where the mapped solid mesh perfectly matches the velocity one, the two assembly techniques for the interface matrix are equivalent.

We solve Problem 2.2.1 in the case of a square solid body $\mathcal{B} = \Omega^s = [-1, 1]^2$ immersed in the fluid domain $\Omega = [-2, 2]^2$ discretized by matching uniform right-oriented meshes. Moreover, as we stated before, the assumption $\mathcal{B} = \Omega^s$ implies that the map $\overline{\mathbf{X}}$ is simply the identity over $\mathcal{B}$, therefore the datum $\mathbf{d}$ is equal to $\mathbf{0}$.

The other right hand sides, $\mathbf{f}$ and $\mathbf{g}$, are computed in such a way that the solutions of our problem are the following:

$$\begin{aligned} &\mathbf{u}(x,y) = \text{curl}\left((4-x^2)^2(4-y^2)^2\right) \\ &\mathbf{u}_{|\partial\Omega} = \mathbf{0} \\ &p(x,y) = 150\sin(x) \\ &\mathbf{X}(x,y) = \mathbf{u}(x,y) \\ &\boldsymbol{\lambda}(x,y) = \left(e^x, e^y\right). \end{aligned} \tag{2.17}$$

In terms of convergence, according to the choice of the Bercovier–Pironneau element, which is a low-order Stokes pair, we expect that the rate of convergence of pressure and velocity with respect to the L^2 and H^1 norm respectively is one, in contrast with a second order convergence for the velocity in L^2.

For the solid variables, according to the standard theory of finite elements [32, 41], we expect that the L^2 error decays with order two, while the H^1 error with order one.

In Tables 2.2 and 2.3, we report relative errors and convergence rates related to the approximation with $\mathcal{P}_1 - iso - \mathcal{P}_2/\mathcal{P}_1/\mathcal{P}_1/\mathcal{P}_1$, while the results for the case $\mathcal{P}_1 - iso - \mathcal{P}_2/\mathcal{P}_1 + \mathcal{P}_0/\mathcal{P}_1/\mathcal{P}_1$ are collected in Tables 2.4 and 2.5.

Analyzing the obtained data, we see that effectively in this case the schemes with and without mesh intersection are equivalent. Moreover, the convergence rates satisfy our expectations, with a superconvergence for pressure and Lagrange multiplier when the classical Bercovier–Pironneu element is used; this phenomenon does not concern the case of the enhanced version.

Test 2

The aim of this test is to simulate a first simple situation in which only the boundary of the structure coincides with the spatial discretization of Ω.

We consider the same setting we used in the previous test, but making a different choice of meshes: indeed, we select a right-oriented uniform mesh for the fluid, while for the solid domain we use a left-oriented uniform mesh.

The results of the simulations are reported in Figure 2.7. We can see that, when the coupling term is assembled in approximate way with the second order quadrature rule, the method presents a slight loss of order of convergence unlike the case in which we use the quadrature formula of order three, which produces results similar to the case with intersection.

Test 3

Now we add another factor of complexity in terms of geometry: in addition to choosing two meshes with non-trivial intersection, we also make sure that the boundary of the solid body does not coincide with the fluid mesh: therefore, we set $\mathcal{B} = \Omega^s = [-0.62, 0.38]^2$ and again $\Omega = [-2, 2]^2$ with the same solutions as in (2.17).

The spatial discretization is done in the same way as in Test 2.

In Figure 2.8, we can see that in this new situation, the different behavior of the two schemes is more evident: it is clear that the optimal convergence rates are ensured only when the interface matrix is exactly assembled with mesh intersection. No improvements are observed when we increase the quadrature precision in the case without mesh intersection.

Test 4

At this point, in the same setting of Test 3, we replace the uniform mesh for $\mathcal{B}$ with an unstructured grid to analyze the convergence when a more general spatial discretization of the solid is considered.

The results of this test are summarized in Figure 2.9: in the case of the continuous pressure element, the convergence deficiency is evident for the approximate case; conversely, when the pressure is approximated with discontinuous elements, the behavior partially improves.

It is important to notice that, in this more general situation, we lose the superconvergence of $\|\boldsymbol{\lambda} - \boldsymbol{\lambda}_h\|_{1,\mathcal{B}}$ we got in the previous tests where only uniform meshes have been used.

Test 5

In this test we want to analyze the behavior of the two approaches in the case of a discontinuous pressure with the discontinuity placed on the boundary $\partial\Omega^s$ since this is a common situation in FSI simulations. In addition, we consider also the case where $\partial\Omega^s$ coincides with the fluid mesh. Therefore, in order to take into account this discontinuity in the numerical scheme, we add a new weak term in the computation of the right hand side $\mathbf{f}$, indeed the pressure term can be decomposed

as follows

$$\int_\Omega p \operatorname{div} \mathbf{v}\, \mathrm{d}\mathbf{x} = \int_{\Omega^s} \mathbf{v}\cdot\nabla p\, \mathrm{d}\mathbf{x} - \int_{\partial\Omega^s} (\mathbf{v}\cdot\mathbf{n}_s)\, p\, \mathrm{da} + \int_{\Omega\setminus\Omega^s} \mathbf{v}\cdot\nabla p\, \mathrm{d}\mathbf{x} - \int_{\partial\Omega^s} (\mathbf{v}\cdot\mathbf{n}_f)\, p\, \mathrm{da},$$

where, again, $\mathbf{n}_s$ and $\mathbf{n}_f$ are outer and inner normal unit vectors to $\partial\Omega^s$. The computation of the boundary terms concerns the integration of fluid variables on the solid mesh, hence the immersion and the mutual position between the two meshes have to be taken into account.

For our test, we choose $\Omega = [-2,2]^2$ and $\mathcal{B} = \Omega^s = [-1,1]^2$, so that $\mathbf{d} = \mathbf{0}$ since $\overline{\mathbf{X}}$ is the identity over $\mathcal{B}$. Once again, we discretize Ω with a sequence of right-oriented uniform meshes and $\mathcal{B}$ with left-oriented uniform meshes. We set the solutions of our problem to be

$$\begin{aligned}
&\mathbf{u}(x,y) = \operatorname{curl}\left((4-x^2)^2(4-y^2)^2\right)\\
&\mathbf{u}_{|\partial\Omega} = \mathbf{0}\\
&p(x,y) = \begin{cases} 150\sin(x) - \frac{50}{3} & \text{in } \Omega^f \\ 150\sin(x) + 50 & \text{in } \Omega^s = \mathcal{B}. \end{cases}\\
&\mathbf{X}(x,y) = \mathbf{u}(x,y)\\
&\boldsymbol{\lambda}(x,y) = \left(e^x, e^y\right).
\end{aligned} \tag{2.18}$$

and we compute $\mathbf{f}$ and $\mathbf{g}$ accordingly.

Tests of this type are useful to understand what are the advantages of using discontinuous elements for the approximation of the pressure function in FSI problem where situations of this type are very common. Indeed, the enhanced pressure space $\mathcal{P}_1 + \mathcal{P}_0$ allows to circumvent the Gibbs phenomenon, obtained instead in the case of classical $\mathcal{P}_1$ elements, reaching the optimal convergence.

In terms of assembling of the interface matrix, we get similar results to those of Test 2, where only the computation of the intersection provides good convergence. Convergence plots are reported in Figure 2.10.

Test 6

We work again with a pressure function which is discontinuous on the solid boundary $\partial\Omega^s$ with the addition of a complication, namely the fact that the discontinuity does not coincide with the velocity mesh. Therefore, still setting $\Omega = [-2,2]^2$, we choose $\mathcal{B} = \Omega^s = [\pi/4, \pi/4]^2$, both discretized with the same sequences of grids we previously used in Test 5.

The computation of the right hand sides is done in order to get as solutions

$$\begin{aligned}
&\mathbf{u}(x,y) = \text{curl}\left((4-x^2)^2(4-y^2)^2\right)\\
&\mathbf{u}_{|\partial\Omega} = \mathbf{0}\\
&p(x,y) = \begin{cases} 150\sin(x) + 50\frac{\pi^2}{4}\left(\frac{\pi^2}{4}-16\right)^{-1} & \text{in } \Omega^f \\ 150\sin(x)+50 & \text{in } \Omega^s = \mathcal{B}. \end{cases}\\
&\mathbf{X}(x,y) = \mathbf{u}(x,y)\\
&\boldsymbol{\lambda}(x,y) = \left(e^x, e^y\right).
\end{aligned} \tag{2.19}$$

Since we have no matching between the pressure discontinuity and the fluid mesh, in this case also the enhanced pressure space $\mathcal{P}_1 + \mathcal{P}_0$ does not work well; indeed the optimal convergence rate cannot be reached due to the Gibbs phenomenon.

As a consequence, we notice that with the two choices of finite element spaces, the fluid variables converge with the same rate both when we calculate the intersection of the meshes, and when we do not compute it. On the other hand, the solid variables are affected by the interface matrix assembly techniques as shown in Figure 2.11.

Test 7

The only situation that we have not already considered in our tests is the one in which the map $\overline{\mathbf{X}}$ is different from the identity.

Therefore, let us consider as reference solid domain the square $\mathcal{B} = [-1,1]^2$, partitioned with uniform left meshes, while we set the actual solid body domain to be the unit disk $\Omega^s = \{\mathbf{x} \in \mathbb{R}^2 : |x| \leq 1\}$, so that $\overline{\mathbf{X}}$ is defined as

$$\overline{\mathbf{X}}(\mathbf{s}) = \left(s_1\sqrt{1-\frac{s_2^2}{2}},\, s_2\sqrt{1-\frac{s_1^2}{2}}\right),$$

where $\mathbf{s} = (s_1, s_2)$. Since $\overline{\mathbf{X}}$ is non-trivial, the right hand side $\mathbf{d}$ is different from the null-vector and in particular we have $\mathbf{d} = \mathbf{u}(\overline{\mathbf{X}}) - \mathbf{X}$ which involves the computation of a term of type $\mathbf{c}(\boldsymbol{\mu}, \mathbf{u}(\overline{\mathbf{X}}))$ via mesh intersection as done for $\mathbf{f}$. Moreover, Ω and the solutions are the same as in Test 1 and (2.17).

In this more general case, we can see that the behavior we observed in all the previous tests is confirmed: the exact procedure is still the best one as reported in Figure 2.12.

Test 8

This last test is a combination of two previous tests: indeed, we consider the same problem setting as for Test 3, but with $\mathcal{B} = [0,1]^2$ which is mapped into the actual solid body $\Omega^s = [-0.62, 0.38]^2$ via the action of

$$\overline{\mathbf{X}}(\mathbf{s}) = (-0.62 + 2s_1,\ -0.62 + 2s_2).$$

The convergence plots are reported in Figure 2.13 and they agree with the results obtained in Test 7.

2.3 The effect of numerical integration: abstract result

Sometimes, in practical situations, the integrals defining bilinear forms and right hand sides are computed with quadrature formulas which are not exact; in our specific formulation, this is the case of the interface matrix, as previously explained.

In this section, we derive a general abstract result concerning the error of numerical integration for all the involved quantities. The starting point for this analysis are the existing results about the stationary problem we recalled in Section 1.5.

Let us first introduce an approximated version of Problem 1.5.3, where $\mathbf{a}_{f,h}$, $\mathbf{a}_{s,h}$ and $\mathbf{c}_h$ denote the numerical counterparts of the bilinear forms $\mathbf{a}_f$, $\mathbf{a}_s$, $\mathbf{c}$, respectively and $(\cdot,\cdot)_h$ denotes the numerical L^2 scalar product. Consequently, we can state the following problem.

Problem 2.3.1. *Let* $\overline{\mathbf{X}} \in \mathbf{W}^{1,\infty}(\mathcal{B})$ *be invertible with Lipschitz inverse and* $\overline{\mathbf{u}} \in \mathbf{H}^1_0(\Omega)$ *such that* $\operatorname{div}\overline{\mathbf{u}} = 0$. *Given* $\mathbf{f} \in \mathbf{L}^2(\Omega)$, $\mathbf{g} \in \mathbf{L}^2(\mathcal{B})$ *and* $\mathbf{d} \in \mathbf{L}^2(\mathcal{B})$, *find* $(\mathbf{u}_h^\star, p_h^\star) \in \underline{\mathbf{V}}_h \times Q_h$, $\mathbf{X}_h^\star \in \underline{\mathbf{S}}_h$ *and* $\boldsymbol{\lambda}_h^\star \in \underline{\boldsymbol{\Lambda}}_h$, *such that*

$$\mathbf{a}_{f,h}(\mathbf{u}_h^\star, \mathbf{v}_h) - (\operatorname{div}\mathbf{v}_h, p_h^\star)_{h,\Omega} + \mathbf{c}_h(\boldsymbol{\lambda}_h^\star, \mathbf{v}_h(\overline{\mathbf{X}})) = (\mathbf{f}, \mathbf{v}_h)_{h,\Omega} \quad \forall \mathbf{v}_h \in \underline{\mathbf{V}}_h \tag{2.20a}$$

$$(\operatorname{div}\mathbf{u}_h^\star, q_h)_{h,\Omega} = 0 \quad \forall q_h \in Q_h \tag{2.20b}$$

$$\mathbf{a}_{s,h}(\mathbf{X}_h^\star, \mathbf{Y}_h) - \mathbf{c}_h(\boldsymbol{\lambda}_h^\star, \mathbf{Y}_h) = (\mathbf{g}, \mathbf{Y}_h)_{h,\mathcal{B}} \quad \forall \mathbf{Y}_h \in \underline{\mathbf{S}}_h \tag{2.20c}$$

$$\mathbf{c}_h(\boldsymbol{\mu}_h, \mathbf{u}_h^\star(\overline{\mathbf{X}}) - \mathbf{X}_h^\star) = \mathbf{c}_h(\boldsymbol{\mu}_h, \mathbf{d}) \quad \forall \boldsymbol{\mu}_h \in \underline{\boldsymbol{\Lambda}}_h. \tag{2.20d}$$

Also in this case, it is useful to rearrange the problem as we did in Section 1.5, so that we can introduce two new discrete bilinear forms, $\mathscr{A}_h : \boldsymbol{\mathcal{V}}_h \times \boldsymbol{\mathcal{V}}_h \to \mathbb{R}$ and

Errors and convergence rates for Test 1 • $\mathcal{P}_1 - \mathbf{iso} - \mathcal{P}_2/\mathcal{P}_1/\mathcal{P}_1/\mathcal{P}_1$						
h_Ω	$\|p - p_h\|_{0,\Omega}$		$\|\mathbf{u} - \mathbf{u}_h\|_{0,\Omega}$		$\|\mathbf{u} - \mathbf{u}_h\|_{1,\Omega}$	
	Error	Rate	Error	Rate	Error	Rate
Coupling with mesh intersection						
1/4	2.102e-02	-	9.622e-03	-	7.684e-02	-
1/8	6.251e-03	1.75	2.408e-03	2.00	3.834e-02	1.00
1/16	1.977e-03	1.66	6.017e-04	2.00	1.915e-02	1.00
1/32	6.546e-04	1.59	1.504e-04	2.00	9.572e-03	1.00
1/64	2.233e-04	1.55	3.758e-05	2.00	4.785e-03	1.00
1/128	7.745e-05	1.53	9.394e-06	2.00	2.392e-03	1.00
Coupling without mesh intersection, quad. rule of order 2						
1/4	2.102e-02	-	9.622e-03	-	7.684e-02	-
1/8	6.251e-03	1.75	2.408e-03	2.00	3.834e-02	1.00
1/16	1.977e-03	1.66	6.017e-04	2.00	1.915e-02	1.00
1/32	6.546e-04	1.59	1.504e-04	2.00	9.572e-03	1.00
1/64	2.233e-04	1.55	3.758e-05	2.00	4.785e-03	1.00
1/128	7.745e-05	1.53	9.394e-06	2.00	2.392e-03	1.00
Coupling without mesh intersection, quad. rule of order 3						
1/4	2.102e-02	-	9.622e-03	-	7.684e-02	-
1/8	6.251e-03	1.75	2.408e-03	2.00	3.834e-02	1.00
1/16	1.977e-03	1.66	6.017e-04	2.00	1.915e-02	1.00
1/32	6.546e-04	1.59	1.504e-04	2.00	9.572e-03	1.00
1/64	2.233e-04	1.55	3.758e-05	2.00	4.785e-03	1.00
1/128	7.745e-05	1.53	9.394e-06	2.00	2.392e-03	1.00

Table 2.2: Errors and convergence rates for the fluid variables of Test 1 discretized with $\mathcal{P}_1 - iso - \mathcal{P}_2/\mathcal{P}_1/\mathcal{P}_1/\mathcal{P}_1$

Errors and convergence rates for Test 1 • $\mathcal{P}_1 - \text{iso} - \mathcal{P}_2/\mathcal{P}_1/\mathcal{P}_1/\mathcal{P}_1$								
$h_{\mathcal{B}}$	$\|\mathbf{X} - \mathbf{X}_h\|_{0,\mathcal{B}}$		$\|\mathbf{X} - \mathbf{X}_h\|_{1,\mathcal{B}}$		$\|\boldsymbol{\lambda} - \boldsymbol{\lambda}_h\|_{0,\mathcal{B}}$		$\|\boldsymbol{\lambda} - \boldsymbol{\lambda}_h\|_{1,\mathcal{B}}$	
	Error	Rate	Error	Rate	Error	Rate	Error	Rate
Coupling with mesh intersection								
1/8	8.011e-03	-	4.972e-02	-	1.908e-01	-	4.166e-01	-
1/16	2.011e-03	1.99	2.479e-02	1.00	4.786e-02	2.00	1.111e-01	1.91
1/32	5.032e-04	2.00	1.239e-02	1.00	1.197e-02	2.00	2.963e-02	1.91
1/64	1.258e-04	2.00	6.193e-03	1.00	2.991e-03	2.00	8.185e-03	1.86
1/128	3.146e-05	2.00	3.096e-03	1.00	7.476e-04	2.00	2.524e-03	1.70
1/256	7.864e-06	2.00	1.548e-03	1.00	1.869e-04	2.00	9.445e-04	1.42
Coupling without mesh intersection, quad. rule of order 2								
1/8	8.011e-03	-	4.972e-02	-	1.908e-01	-	4.166e-01	-
1/16	2.011e-03	1.99	2.479e-02	1.00	4.786e-02	2.00	1.111e-01	1.91
1/32	5.032e-04	2.00	1.239e-02	1.00	1.197e-02	2.00	2.963e-02	1.91
1/64	1.258e-04	2.00	6.193e-03	1.00	2.991e-03	2.00	8.185e-03	1.86
1/128	3.146e-05	2.00	3.096e-03	1.00	7.476e-04	2.00	2.524e-03	1.70
1/256	7.864e-06	2.00	1.548e-03	1.00	1.869e-04	2.00	9.445e-04	1.42
Coupling without mesh intersection, quad. rule of order 3								
1/8	8.011e-03	-	4.972e-02	-	1.908e-01	-	4.166e-01	-
1/16	2.011e-03	1.99	2.479e-02	1.00	4.786e-02	2.00	1.111e-01	1.91
1/32	5.032e-04	2.00	1.239e-02	1.00	1.197e-02	2.00	2.963e-02	1.91
1/64	1.258e-04	2.00	6.193e-03	1.00	2.991e-03	2.00	8.185e-03	1.86
1/128	3.146e-05	2.00	3.096e-03	1.00	7.476e-04	2.00	2.524e-03	1.70
1/256	7.864e-06	2.00	1.548e-03	1.00	1.869e-04	2.00	9.445e-04	1.42

Table 2.3: Errors and convergence rates for the solid variables of Test 1 discretized with $\mathcal{P}_1 - iso - \mathcal{P}_2/\mathcal{P}_1/\mathcal{P}_1/\mathcal{P}_1$

Errors and convergence rates for Test 1 • $\mathcal{P}_1 - \mathbf{iso} - \mathcal{P}_2/\mathcal{P}_1 + \mathcal{P}_0/\mathcal{P}_1/\mathcal{P}_1$						
h_Ω	$\|p - p_h\|_{0,\Omega}$		$\|\mathbf{u} - \mathbf{u}_h\|_{0,\Omega}$		$\|\mathbf{u} - \mathbf{u}_h\|_{1,\Omega}$	
	Error	Rate	Error	Rate	Error	Rate
Coupling with mesh intersection						
1/4	7.981e-02	-	1.043e-02	-	8.042e-02	-
1/8	3.939e-02	1.02	2.617e-03	1.99	4.017e-02	1.00
1/16	1.957e-02	1.01	6.549e-04	2.00	2.008e-02	1.00
1/32	9.749e-03	1.00	1.637e-04	2.00	1.004e-02	1.00
1/64	4.866e-03	1.00	4.093e-05	2.00	5.018e-03	1.00
1/128	2.431e-03	1.00	1.023e-05	2.00	2.509e-03	1.00
Coupling without mesh intersection, quad. rule of order 2						
1/4	7.981e-02	-	1.043e-02	-	8.042e-02	-
1/8	3.939e-02	1.02	2.617e-03	1.99	4.017e-02	1.00
1/16	1.957e-02	1.01	6.549e-04	2.00	2.008e-02	1.00
1/32	9.749e-03	1.00	1.637e-04	2.00	1.004e-02	1.00
1/64	4.866e-03	1.00	4.093e-05	2.00	5.018e-03	1.00
1/128	2.431e-03	1.00	1.023e-05	2.00	2.509e-03	1.00
Coupling without mesh intersection, quad. rule of order 3						
1/4	7.981e-02	-	1.043e-02	-	8.042e-02	-
1/8	3.939e-02	1.02	2.617e-03	1.99	4.017e-02	1.00
1/16	1.957e-02	1.01	6.549e-04	2.00	2.008e-02	1.00
1/32	9.749e-03	1.00	1.637e-04	2.00	1.004e-02	1.00
1/64	4.866e-03	1.00	4.093e-05	2.00	5.018e-03	1.00
1/128	2.431e-03	1.00	1.023e-05	2.00	2.509e-03	1.00

Table 2.4: Errors and convergence rates for the fluid variables of Test 1 discretized with $\mathcal{P}_1 - iso - \mathcal{P}_2/\mathcal{P}_1 + \mathcal{P}_0/\mathcal{P}_1/\mathcal{P}_1$

Errors and convergence rates for Test 1 • $\mathcal{P}_1 - \mathbf{iso} - \mathcal{P}_2/\mathcal{P}_1 + \mathcal{P}_0/\mathcal{P}_1/\mathcal{P}_1$								
$h_{\mathcal{B}}$	$\|\mathbf{X} - \mathbf{X}_h\|_{0,\mathcal{B}}$		$\|\mathbf{X} - \mathbf{X}_h\|_{1,\mathcal{B}}$		$\|\boldsymbol{\lambda} - \boldsymbol{\lambda}_h\|_{0,\mathcal{B}}$		$\|\boldsymbol{\lambda} - \boldsymbol{\lambda}_h\|_{1,\mathcal{B}}$	
	Error	Rate	Error	Rate	Error	Rate	Error	Rate
Coupling with mesh intersection								
1/8	8.854e-03	-	5.239e-02	-	2.300e-01	-	1.861e+00	-
1/16	2.228e-03	1.99	2.616e-02	1.00	5.802e-02	1.99	9.338e-01	0.99
1/32	5.578e-04	2.00	1.308e-02	1.00	1.453e-02	2.00	4.672e-01	1.00
1/64	1.395e-04	2.00	6.539e-03	1.00	3.632e-03	2.00	2.336e-01	1.00
1/128	3.487e-05	2.00	3.269e-03	1.00	9.078e-04	2.00	1.168e-01	1.00
1/256	8.718e-06	2.00	1.635e-03	1.00	2.269e-04	2.00	5.840e-02	1.00
Coupling without mesh intersection, quad. rule of order 2								
1/8	8.854e-03	-	5.239e-02	-	2.300e-01	-	1.861e+00	-
1/16	2.228e-03	1.99	2.616e-02	1.00	5.802e-02	1.99	9.338e-01	0.99
1/32	5.578e-04	2.00	1.308e-02	1.00	1.453e-02	2.00	4.672e-01	1.00
1/64	1.395e-04	2.00	6.539e-03	1.00	3.632e-03	2.00	2.336e-01	1.00
1/128	3.487e-05	2.00	3.269e-03	1.00	9.078e-04	2.00	1.168e-01	1.00
1/256	8.718e-06	2.00	1.635e-03	1.00	2.269e-04	2.00	5.840e-02	1.00
Coupling without mesh intersection, quad. rule of order 3								
1/8	8.854e-03	-	5.239e-02	-	2.300e-01	-	1.861e+00	-
1/16	2.228e-03	1.99	2.616e-02	1.00	5.802e-02	1.99	9.338e-01	0.99
1/32	5.578e-04	2.00	1.308e-02	1.00	1.453e-02	2.00	4.672e-01	1.00
1/64	1.395e-04	2.00	6.539e-03	1.00	3.632e-03	2.00	2.336e-01	1.00
1/128	3.487e-05	2.00	3.269e-03	1.00	9.078e-04	2.00	1.168e-01	1.00
1/256	8.718e-06	2.00	1.635e-03	1.00	2.269e-04	2.00	5.840e-02	1.00

Table 2.5: Errors and convergence rates for the solid variables of Test 1 discretized with $\mathcal{P}_1 - iso - \mathcal{P}_2/\mathcal{P}_1 + \mathcal{P}_0/\mathcal{P}_1/\mathcal{P}_1$

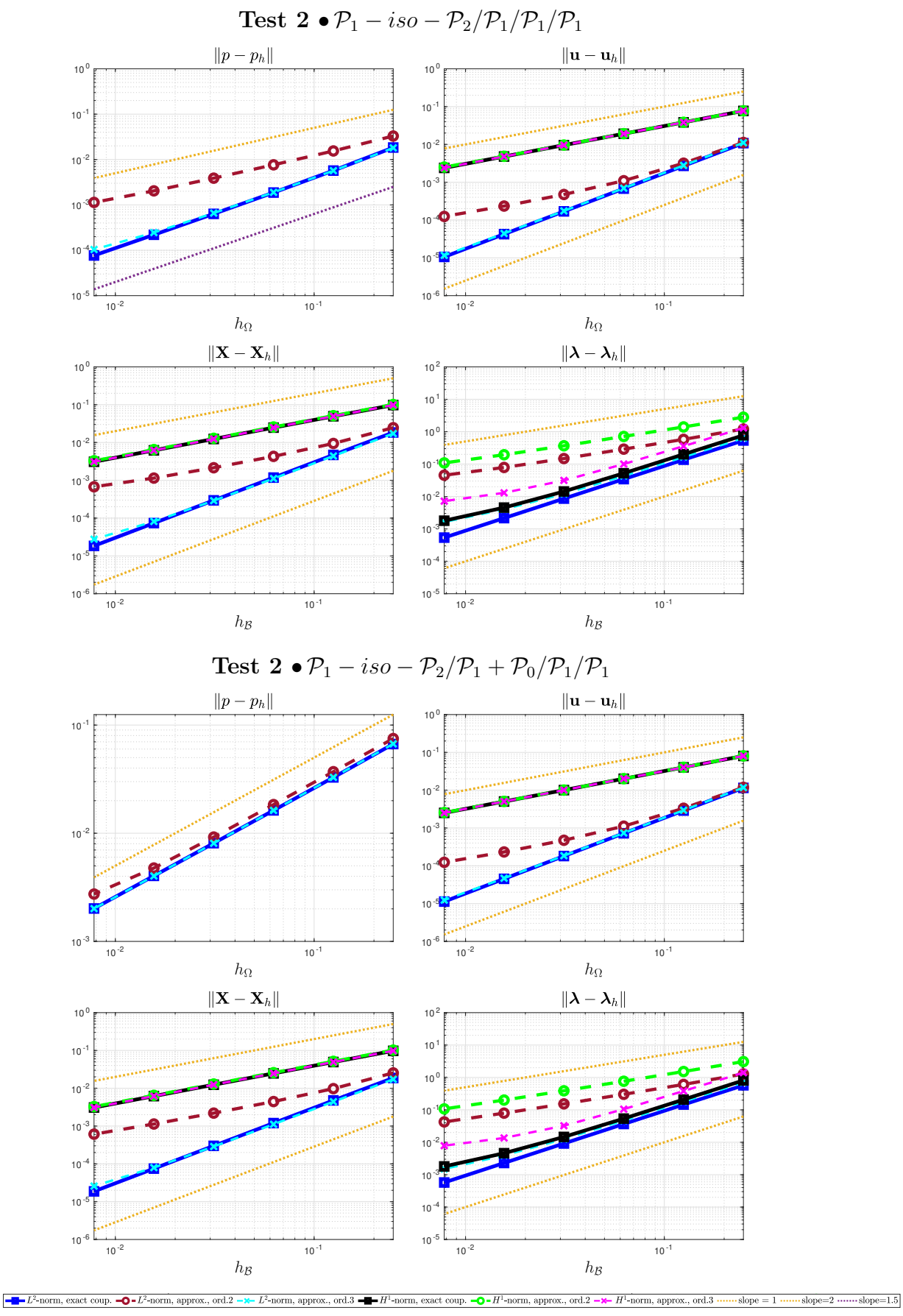


Figure 2.7: Convergence plots for Test 2

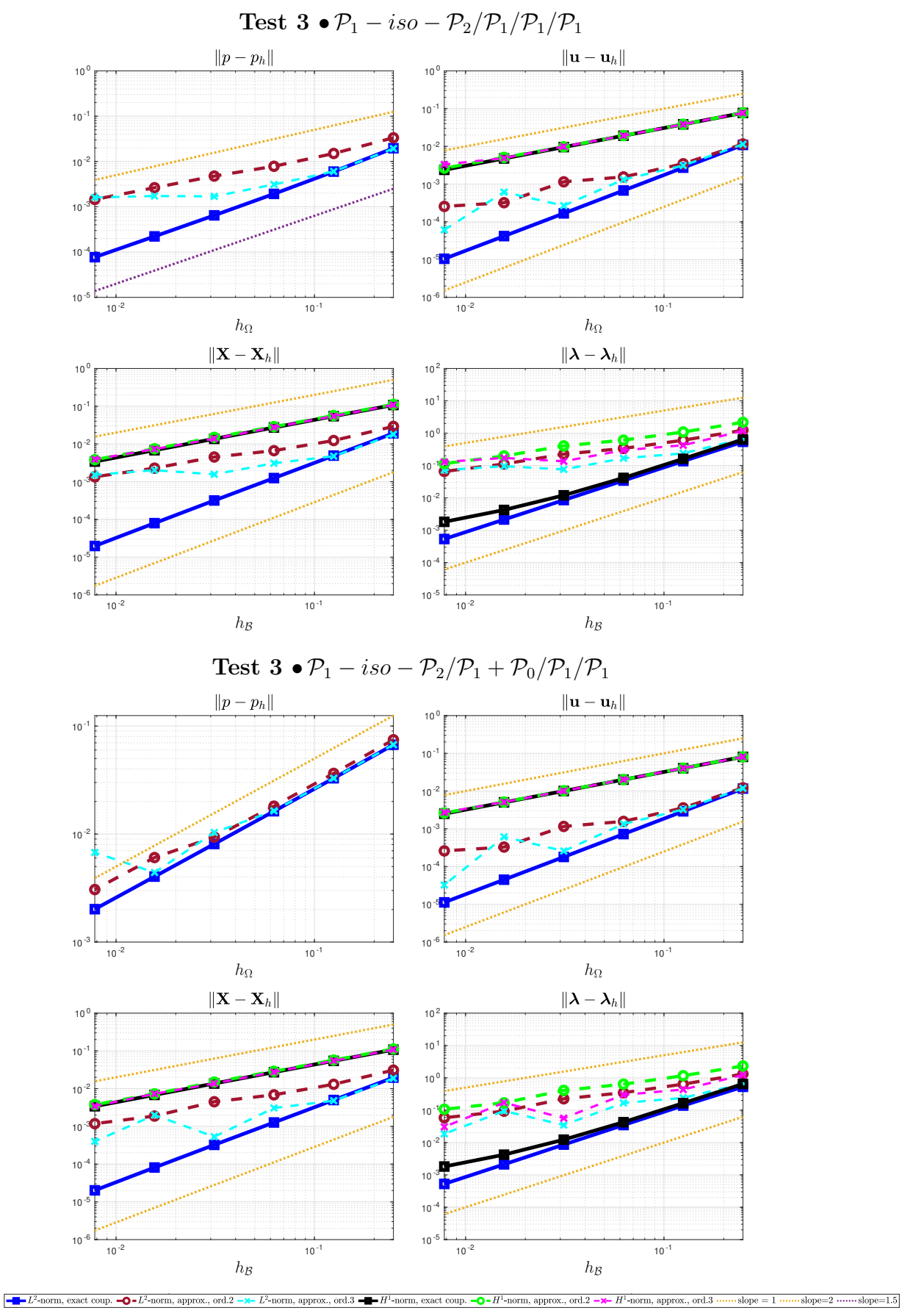


Figure 2.8: Convergence plots for Test 3

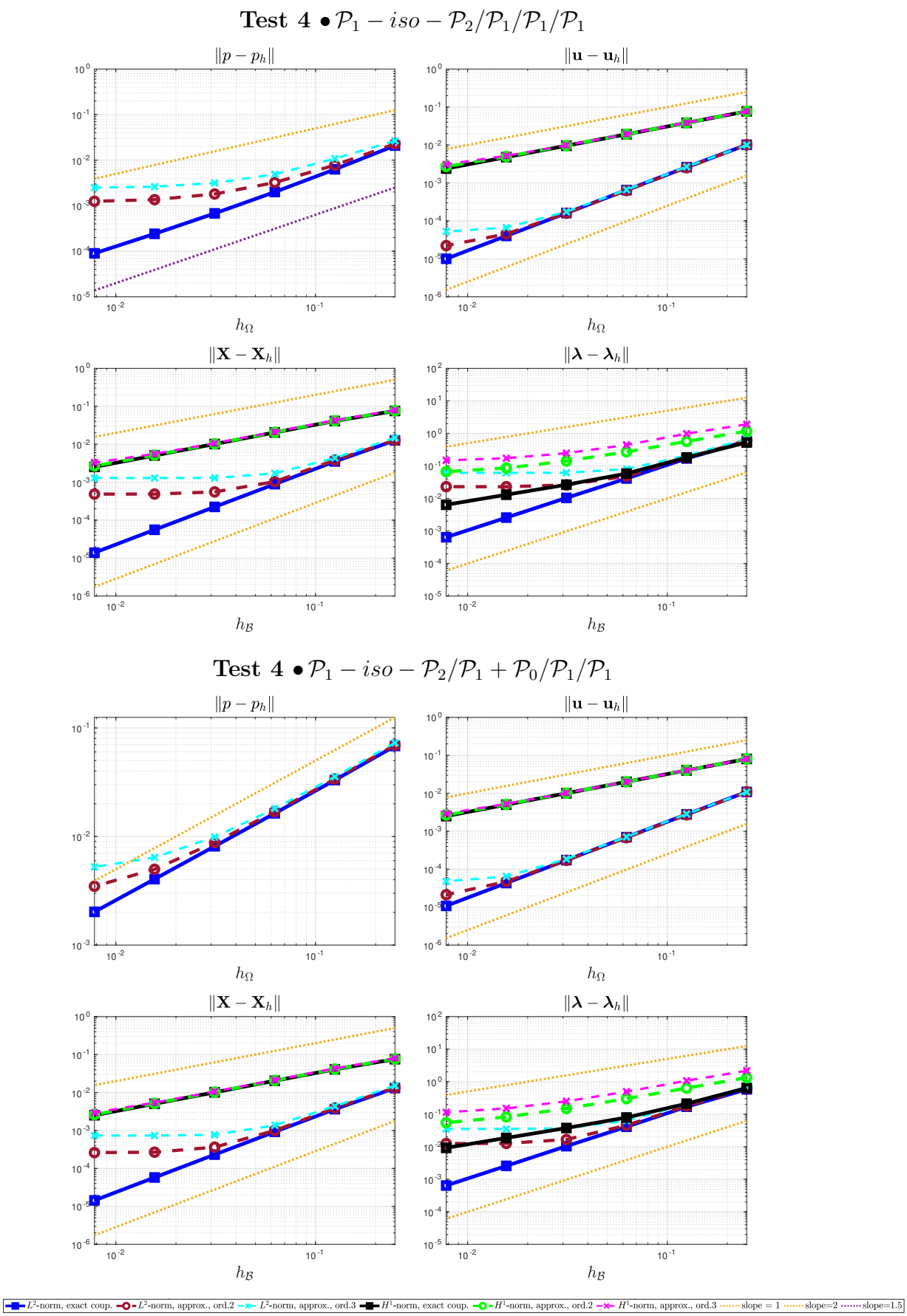


Figure 2.9: Convergence plots for Test 4

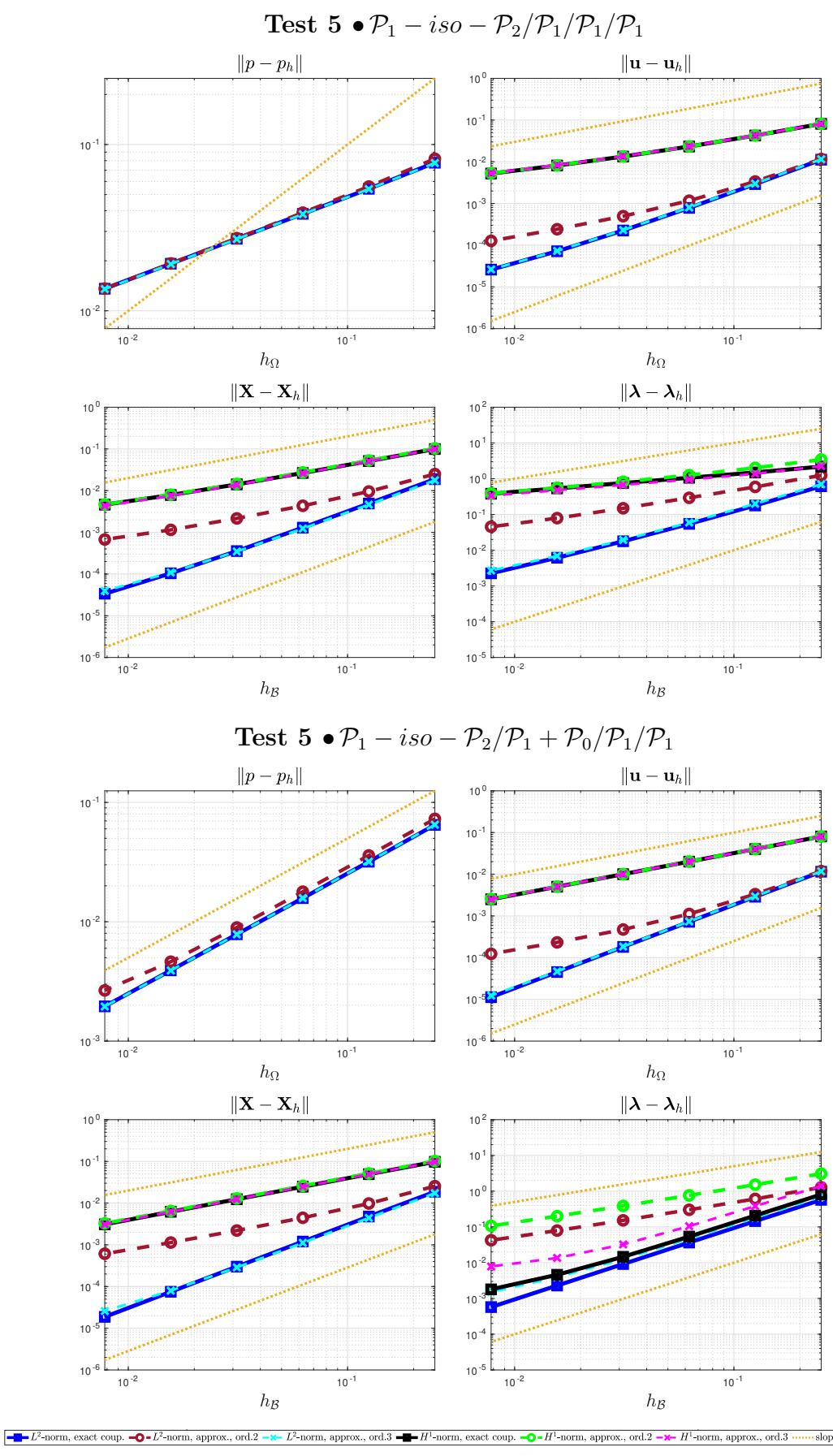


Figure 2.10: Convergence plots of Test 5

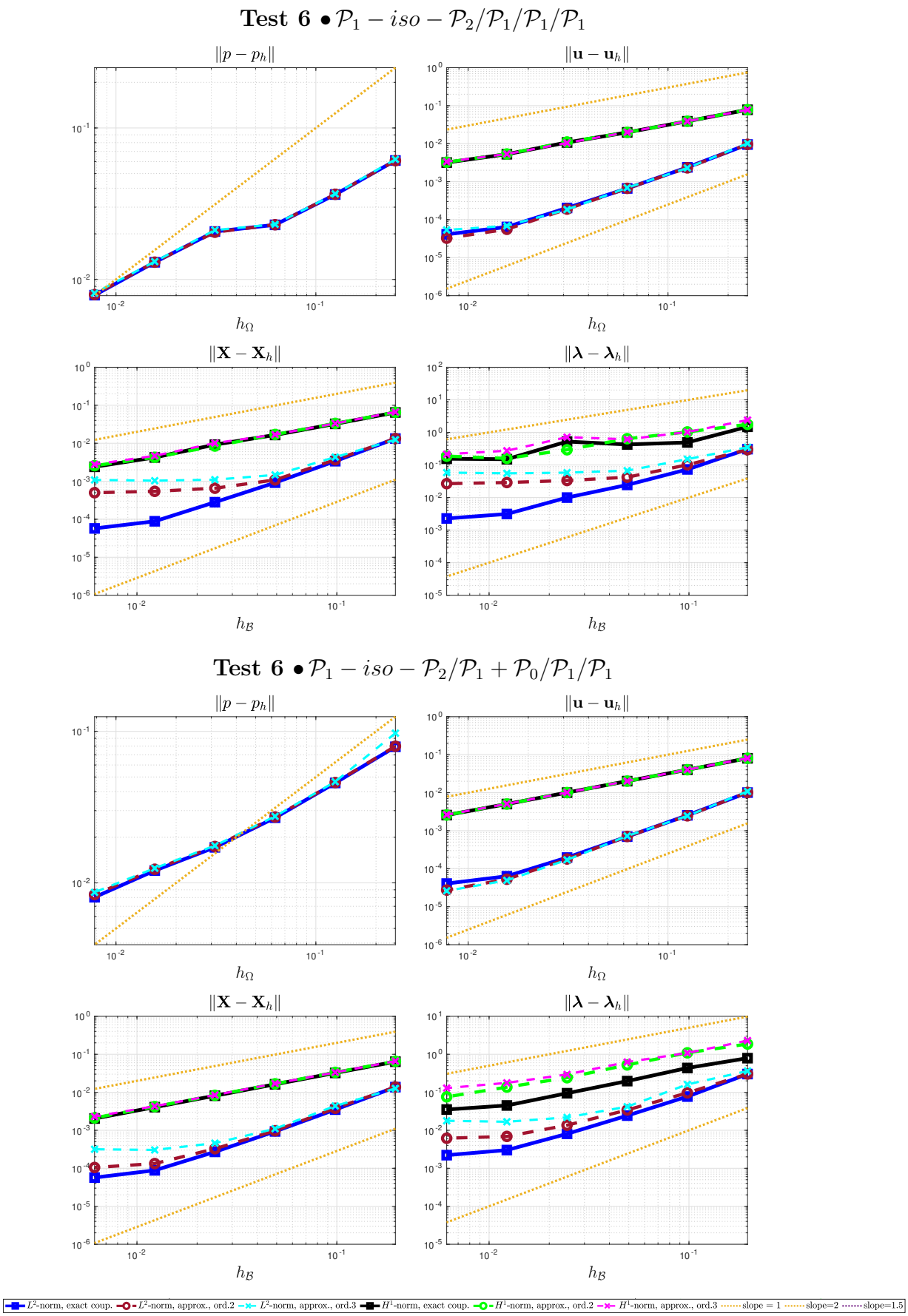


Figure 2.11: Convergence plots of Test 6

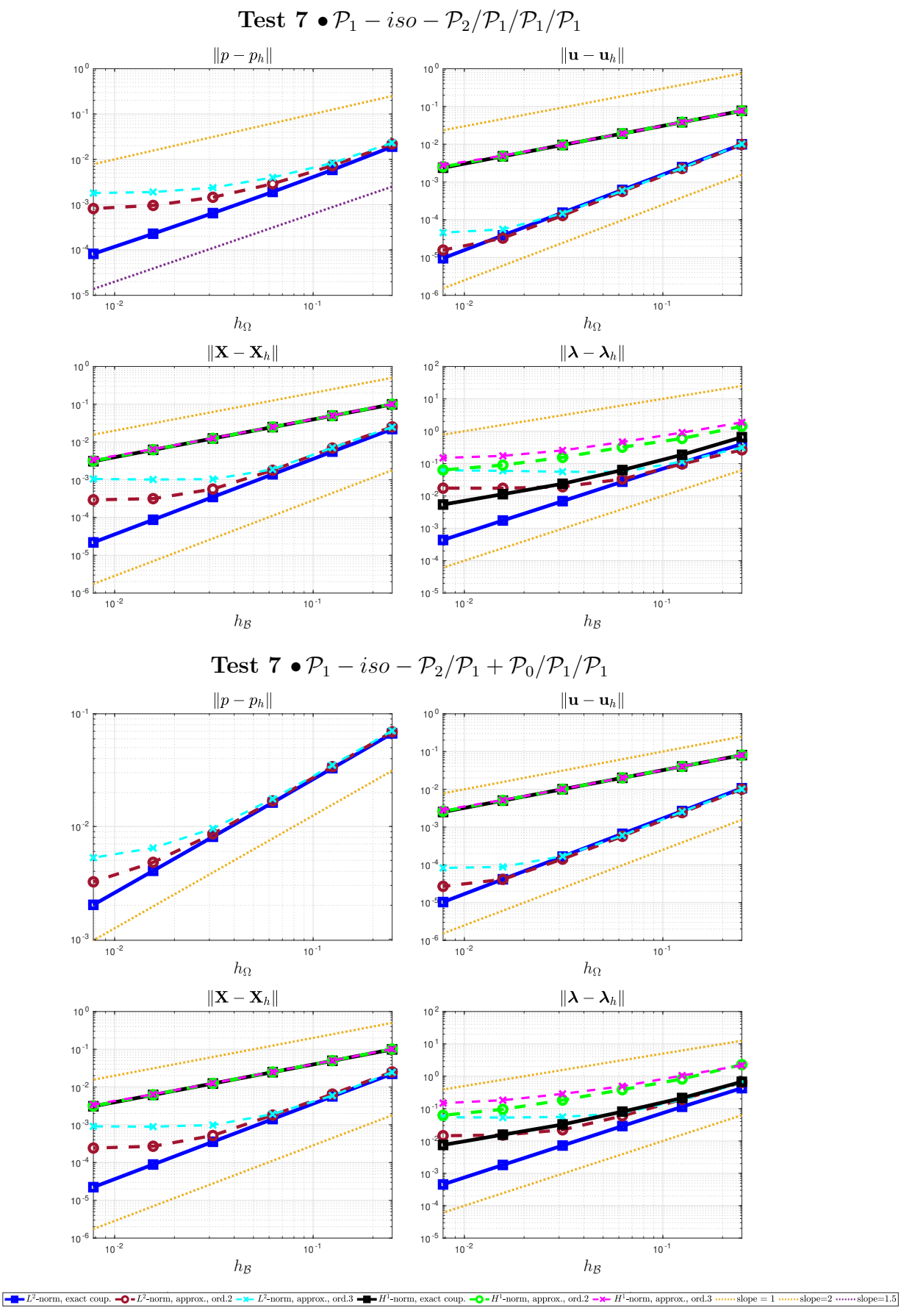


Figure 2.12: Convergence plots of Test 7

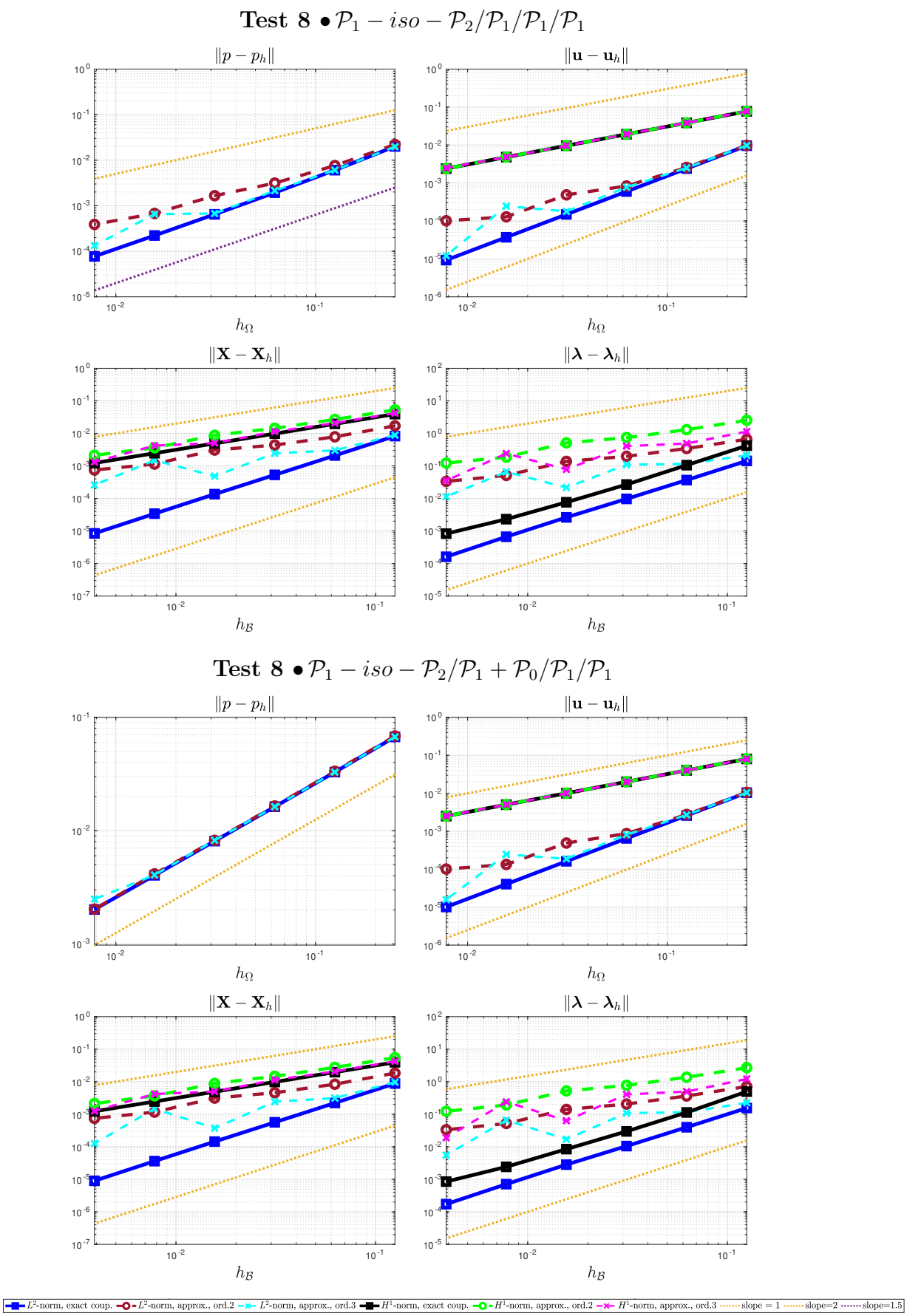


Figure 2.13: Convergence plots of Test 8

$\mathscr{B}_h : \boldsymbol{\mathcal{V}}_h \times Q_h \to \mathbb{R}$, such that

$$\begin{aligned}\mathscr{A}_h(\mathbf{U}_h, \mathbf{V}_h) &= \mathbf{a}_{f,h}(\mathbf{u}_h, \mathbf{v}_h) + \mathbf{a}_{s,h}(\mathbf{X}_h, \mathbf{Y}_h) \\ &\quad + \mathbf{c}_h(\boldsymbol{\lambda}_h, \mathbf{v}_h(\overline{\mathbf{X}}) - \mathbf{Y}_h) - \mathbf{c}_h(\boldsymbol{\mu}_h, \mathbf{u}_h(\overline{\mathbf{X}}) - \mathbf{X}_h) \\ \mathscr{B}_h(\mathbf{V}_h, q_h) &= (\operatorname{div} \mathbf{v}_h, q_h)_{h,\Omega}\end{aligned} \tag{2.21}$$

for all $\mathbf{V}_h \in \boldsymbol{\mathcal{V}}_h$, $q_h \in Q_h$.

In the same way, we introduce the solution operator $\mathscr{L}_h^\star : \boldsymbol{\mathcal{V}}_h \times Q_h \to \boldsymbol{\mathcal{V}}_h' \times Q_h'$ satisfying, for all $\mathbf{V}_h \in \boldsymbol{\mathcal{V}}_h$ and $q_h \in Q_h$,

$$\langle \mathscr{L}_h^\star(\mathbf{U}_h, p_h), (\mathbf{V}_h, q_h) \rangle = \mathscr{A}_h(\mathbf{U}_h, \mathbf{V}_h) + \mathscr{B}_h(\mathbf{V}_h, p_h) + \mathscr{B}_h(\mathbf{U}_h, q_h). \tag{2.22}$$

We are interested in measuring the error between the exact solution $(\mathbf{u}, p, \mathbf{X}, \boldsymbol{\lambda})$ and the approximated one, $(\mathbf{u}_h^\star, p_h^\star, \mathbf{X}_h^\star, \boldsymbol{\lambda}_h^\star)$, when solving Problem 2.3.1.

Before starting the analysis, we present some useful ingredients. First of all, we assume that Problem 2.3.1 is well-posed, i.e. that it satisfies the inf-sup conditions.

Assumption 2.3.1. *$\mathscr{A}_h$ and $\mathscr{B}_h$ satisfy the inf-sup conditions, that is there exist two positive constants $\theta^\star$ and $\eta^\star$ such that*

$$\inf_{\mathbf{U}_h \in \boldsymbol{\mathcal{K}}^\star[\mathscr{B}_h]} \sup_{\mathbf{V}_h \in \boldsymbol{\mathcal{K}}^\star[\mathscr{B}_h]} \frac{\mathscr{A}_h(\mathbf{U}_h, \mathbf{V}_h)}{|||\mathbf{U}_h||| \, |||\mathbf{V}_h|||} \geq \theta^\star, \qquad \inf_{q_h \in Q_h} \sup_{\mathbf{V}_h \in \boldsymbol{\mathcal{V}}_h} \frac{\mathscr{B}_h(\mathbf{V}_h, q_h)}{|||\mathbf{V}_h||| \, \|q_h\|_{0,\Omega}} \geq \eta^\star, \tag{2.23}$$

where $\boldsymbol{\mathcal{K}}^\star[\mathscr{B}_h] = \{\mathbf{V}_h \in \boldsymbol{\mathcal{V}}_h : \mathscr{B}_h(\mathbf{V}_h, q_h) = 0 \quad \forall q_h \in Q_h\}$.

We start our investigation applying [34, Prop.1.1]: there exists a pair $(\mathbf{V}_h, q_h)$ such that the following bound holds true

$$|||\mathbf{U}_h - \mathbf{U}_h^\star||| + \|p_h - p_h^\star\|_{0,\Omega} \leq M \frac{\langle \mathscr{L}_h^\star(\mathbf{U}_h - \mathbf{U}_h^\star, p_h - p_h^\star), (\mathbf{V}_h, q_h) \rangle}{|||\mathbf{V}_h||| + \|q_h\|_{0,\Omega}} \tag{2.24}$$

for a constant M depending on $\eta^\star$ and $\theta^\star$. This inequality is consequence of Assumption 2.3.1: the equivalence between Brezzi [34] and Babuška [7] theories has been discussed in [96].

Now, with some easy manipulations involving the definition of $\mathscr{A}_h$ and the linearity, we have that

$$\begin{aligned}\langle \mathscr{L}_h^\star(\mathbf{U}_h - \mathbf{U}_h^\star, p_h - p_h^\star), (\mathbf{V}_h, q_h) \rangle = \\ \langle (\mathscr{L}_h^\star - \mathscr{L}_h)(\mathbf{U}_h, p_h), (\mathbf{V}_h, q_h) \rangle + (\mathbf{f}, \mathbf{v}_h)_\Omega - (\mathbf{f}, \mathbf{v}_h)_{h,\Omega} \\ + (\mathbf{g}, \mathbf{Y}_h)_\mathcal{B} - (\mathbf{g}, \mathbf{Y}_h)_{h,\mathcal{B}} + \mathbf{c}(\mathbf{d}, \boldsymbol{\mu}_h) - \mathbf{c}_h(\mathbf{d}, \boldsymbol{\mu}_h)\end{aligned} \tag{2.25}$$

where we also used that $(\mathbf{U}_h, p_h)$ is solution of Problem 1.5.3.

Now, expanding the definition of the operators $\mathscr{L}_h$ and $\mathscr{L}_h^\star$, thanks to the triangular inequality, we get the following expression

$$\begin{aligned}|\langle(\mathscr{L}_h^\star - \mathscr{L}_h)(\mathbf{U}_h, p_h), (\mathbf{V}, q_h)\rangle| \leq\ & |\mathbf{a}_{f,h}(\mathbf{u}_h, \mathbf{v}_h) - \mathbf{a}_f(\mathbf{u}_h, \mathbf{v}_h)| \\ & + |\mathbf{a}_{s,h}(\mathbf{X}_h, \mathbf{Y}_h) - \mathbf{a}_s(\mathbf{X}_h, \mathbf{Y}_h)| \\ & + \left|\mathbf{c}_h(\boldsymbol{\lambda}_h, \mathbf{v}_h(\overline{\mathbf{X}}) - \mathbf{Y}_h) - \mathbf{c}(\boldsymbol{\lambda}_h, \mathbf{v}_h(\overline{\mathbf{X}}) - \mathbf{Y}_h)\right| \\ & + \left|\mathbf{c}_h(\boldsymbol{\mu}_h, \mathbf{u}_h(\overline{\mathbf{X}}) - \mathbf{X}_h) - \mathbf{c}(\boldsymbol{\mu}_h, \mathbf{u}_h(\overline{\mathbf{X}}) - \mathbf{X}_h)\right| \\ & + |\mathscr{B}_h(\mathbf{v}_h, p_h) - \mathscr{B}(\mathbf{v}_h, p_h)| \\ & + |\mathscr{B}_h(\mathbf{u}_h, q_h) - \mathscr{B}(\mathbf{u}_h, q_h)|\,.\end{aligned} \tag{2.26}$$

Therefore, combining (2.24) and (2.25), we obtain

$$\begin{aligned}|||\mathbf{U}_h - \mathbf{U}_h^\star||| + \|p_h - p_h^\star\|_{0,\Omega} &\leq M\langle\mathscr{L}_h^\star(\mathbf{U}_h - \mathbf{U}_h^\star, p_h - p_h^\star), (\mathbf{V}, q_h)\rangle(|||\mathbf{V}||| + \|q_h\|_{0,\Omega})^{-1} \\ &= M\big[\langle(\mathscr{L}_h^\star - \mathscr{L}_h)(\mathbf{U}_h, p_h), (\mathbf{V}, q_h)\rangle \\ &\quad + (\mathbf{f}, \mathbf{v}_h)_\Omega - (\mathbf{f}, \mathbf{v}_h)_{h,\Omega} + (\mathbf{g}, \mathbf{Y}_h)_\mathcal{B} - (\mathbf{g}, \mathbf{Y}_h)_{h,\mathcal{B}} \\ &\quad + \mathbf{c}(\mathbf{d}, \boldsymbol{\mu}_h) - \mathbf{c}_h(\mathbf{d}, \boldsymbol{\mu}_h)\big](|||\mathbf{V}||| + \|q_h\|_{0,\Omega})^{-1}\end{aligned} \tag{2.27}$$

so that taking into account equation (2.26), we get the final bound

$$\begin{aligned}&|||\mathbf{U}_h - \mathbf{U}_h^\star||| + \|p_h - p_h^\star\|_{0,\Omega} \\ &\quad\leq M\big\{|\mathbf{a}_{f,h}(\mathbf{u}_h, \mathbf{v}_h) - \mathbf{a}_f(\mathbf{u}_h, \mathbf{v}_h)| + |\mathbf{a}_{s,h}(\mathbf{X}_h, \mathbf{Y}_h) - \mathbf{a}_s(\mathbf{X}_h, \mathbf{Y}_h)| \\ &\qquad + \left|\mathbf{c}_h(\boldsymbol{\lambda}_h, \mathbf{v}_h(\overline{\mathbf{X}}) - \mathbf{Y}_h) - \mathbf{c}(\boldsymbol{\lambda}_h, \mathbf{v}_h(\overline{\mathbf{X}}) - \mathbf{Y}_h)\right| \\ &\qquad + \left|\mathbf{c}_h(\boldsymbol{\mu}_h, \mathbf{u}_h(\overline{\mathbf{X}}) - \mathbf{X}_h) - \mathbf{c}(\boldsymbol{\mu}_h, \mathbf{u}_h(\overline{\mathbf{X}}) - \mathbf{X}_h)\right| \\ &\qquad + |\mathscr{B}_h(\mathbf{v}_h, p_h) - \mathscr{B}(\mathbf{v}_h, p_h)| + |\mathscr{B}_h(\mathbf{u}_h, q_h) - \mathscr{B}(\mathbf{u}_h, q_h)| \\ &\qquad + |(\mathbf{f}, \mathbf{v}_h)_\Omega - (\mathbf{f}, \mathbf{v}_h)_{h,\Omega}| + |(\mathbf{g}, \mathbf{Y}_h)_\mathcal{B} - (\mathbf{g}, \mathbf{Y}_h)_{h,\mathcal{B}}| \\ &\qquad + |\mathbf{c}(\mathbf{d}, \boldsymbol{\mu}_h) - \mathbf{c}_h(\mathbf{d}, \boldsymbol{\mu}_h)|\big\}(|||\mathbf{V}_h||| + \|q_h\|_{0,\Omega})^{-1} \\ &\quad\leq M\big\{\mathcal{A}_f(\mathbf{u}_h) + \mathcal{A}_s(\mathbf{X}_h) + \mathcal{C}_f(\boldsymbol{\lambda}_h) + \mathcal{C}_s(\boldsymbol{\lambda}_h) + \mathcal{C}_f(\mathbf{u}_h) \\ &\qquad\quad + \mathcal{C}_s(\mathbf{X}_h) + \mathcal{H}(\mathbf{u}_h) + \mathcal{H}^\top(p_h) + \mathcal{F} + \mathcal{G} + \mathcal{D}\big\}\end{aligned} \tag{2.28}$$

where the bilinear forms are given by

$$\begin{aligned}
\mathcal{A}_f(\mathbf{u}_h) &= \sup_{\mathbf{v}_h\in\underline{\mathbf{V}}_h\setminus\{0\}} \frac{|\mathbf{a}_f(\mathbf{u}_h,\mathbf{v}_h)-\mathbf{a}_{f,h}(\mathbf{u}_h,\mathbf{v}_h)|}{\|\mathbf{v}_h\|_{1,\Omega}}\\
\mathcal{A}_s(\mathbf{X}_h) &= \sup_{\mathbf{Y}_h\in\underline{\mathbf{S}}_h\setminus\{0\}} \frac{|\mathbf{a}_s(\mathbf{X}_h,\mathbf{Y}_h)-\mathbf{a}_{s,h}(\mathbf{X}_h,\mathbf{Y}_h)|}{\|\mathbf{Y}_h\|_{1,\mathcal{B}}}\\
\mathcal{C}_s(\mathbf{X}_h) &= \sup_{\boldsymbol{\mu}_h\in\underline{\boldsymbol{\Lambda}}_h\setminus\{0\}} \frac{|\mathbf{c}(\boldsymbol{\mu}_h,\mathbf{X}_h)-\mathbf{c}_h(\boldsymbol{\mu}_h,\mathbf{X}_h)|}{\|\boldsymbol{\mu}_h\|_{\underline{\boldsymbol{\Lambda}}}}\\
\mathcal{C}_f(\mathbf{u}_h) &= \sup_{\boldsymbol{\mu}_h\in\underline{\boldsymbol{\Lambda}}_h\setminus\{0\}} \frac{\big|\mathbf{c}(\boldsymbol{\mu}_h,\mathbf{u}_h(\overline{\mathbf{X}}))-\mathbf{c}_h(\boldsymbol{\mu}_h,\mathbf{u}_h(\overline{\mathbf{X}}))\big|}{\|\boldsymbol{\mu}_h\|_{\underline{\boldsymbol{\Lambda}}}}\\
\mathcal{C}_s^\top(\boldsymbol{\lambda}_h) &= \sup_{\mathbf{Y}_h\in\underline{\mathbf{S}}_h\setminus\{0\}} \frac{|\mathbf{c}(\boldsymbol{\lambda}_h,\mathbf{Y}_h)-\mathbf{c}_h(\boldsymbol{\lambda}_h,\mathbf{Y}_h)|}{\|\mathbf{Y}_h\|_{1,\mathcal{B}}}\\
\mathcal{C}_f^\top(\boldsymbol{\lambda}_h) &= \sup_{\mathbf{v}_h\in\underline{\mathbf{V}}_h\setminus\{0\}} \frac{\big|\mathbf{c}(\boldsymbol{\lambda}_h,\mathbf{v}_h(\overline{\mathbf{X}}))-\mathbf{c}_h(\boldsymbol{\lambda}_h,\mathbf{v}_h(\overline{\mathbf{X}}))\big|}{\|\mathbf{v}_h\|_{1,\Omega}}\\
\mathcal{H}(\mathbf{u}_h) &= \sup_{q_h\in Q_h\setminus\{0\}} \frac{|\mathscr{B}(\mathbf{u}_h,q_h)-\mathscr{B}_h(\mathbf{u}_h,q_h)|}{\|q_h\|_{0,\Omega}}\\
\mathcal{H}^\top(p_h) &= \sup_{\mathbf{v}_h\in\underline{\mathbf{V}}_h\setminus\{0\}} \frac{|\mathscr{B}(\mathbf{v}_h,p_h)-\mathscr{B}_h(\mathbf{v}_h,p_h)|}{\|\mathbf{v}_h\|_{1,\Omega}}
\end{aligned} \tag{2.29}$$

and the terms on the right hand side are

$$\begin{aligned}
\mathcal{F} &= \sup_{\mathbf{v}_h\in\underline{\mathbf{V}}_h\setminus\{0\}} \frac{|(\mathbf{f},\mathbf{v}_h)_\Omega-(\mathbf{f},\mathbf{v}_h)_{h,\Omega}|}{\|\mathbf{v}_h\|_{1,\Omega}}\\
\mathcal{G} &= \sup_{\mathbf{Y}_h\in\underline{\mathbf{S}}_h\setminus\{0\}} \frac{|(\mathbf{g},\mathbf{Y}_h)_\mathcal{B}-(\mathbf{g},\mathbf{Y}_h)_{h,\mathcal{B}}|}{\|\mathbf{Y}_h\|_{1,\mathcal{B}}}\\
\mathcal{D} &= \sup_{\boldsymbol{\mu}_h\in\underline{\boldsymbol{\Lambda}}_h\setminus\{0\}} \frac{|\mathbf{c}(\mathbf{d},\boldsymbol{\mu}_h)-\mathbf{c}_h(\mathbf{d},\boldsymbol{\mu}_h)|}{\|\boldsymbol{\mu}_h\|_{\underline{\boldsymbol{\Lambda}}}}.
\end{aligned} \tag{2.30}$$

Finally, the final abstract result reads as follows.

Theorem 2.3.1. *In the setting given by Assumption 2.3.1, if* $(\mathbf{u},p,\mathbf{X},\boldsymbol{\lambda})$ *is solution of Problem 1.5.1 and* $(\mathbf{u}_h^\star,p_h^\star,\mathbf{X}_h^\star,\boldsymbol{\lambda}_h^\star)$ *is solution of Problem 2.3.1, then the following error estimate holds true*

$$\begin{aligned}
&\|\mathbf{u}-\mathbf{u}_h^\star\|_{1,\Omega}+\|p-p_h^\star\|_{0,\Omega}+\|\mathbf{X}-\mathbf{X}_h^\star\|_{1,\mathcal{B}}+\|\boldsymbol{\lambda}-\boldsymbol{\lambda}_h^\star\|_{\underline{\boldsymbol{\Lambda}}}\\
&\qquad\le \|\mathbf{u}-\mathbf{u}_h\|_{1,\Omega}+\|p-p_h\|_{0,\Omega}+\|\mathbf{X}-\mathbf{X}_h\|_{1,\mathcal{B}}+\|\boldsymbol{\lambda}-\boldsymbol{\lambda}_h\|_{\underline{\boldsymbol{\Lambda}}}\\
&\qquad\quad + M\big\{\mathcal{A}_f(\mathbf{u}_h)+\mathcal{A}_s(\mathbf{X}_h)+\mathcal{C}_f(\boldsymbol{\lambda}_h)+\mathcal{C}_s(\boldsymbol{\lambda}_h)+\mathcal{C}_f(\mathbf{u}_h)\\
&\qquad\qquad +\mathcal{C}_s(\mathbf{X}_h)+\mathcal{H}(\mathbf{u}_h)+\mathcal{H}^\top(p_h)+\mathcal{F}+\mathcal{G}+\mathcal{D}\big\},
\end{aligned} \tag{2.31}$$

where $(\mathbf{u}_h, p_h, \mathbf{X}_h, \boldsymbol{\lambda}_h)$ *is solution of Problem 1.5.3 and* $\mathcal{A}_f, \mathcal{A}_s, \mathcal{C}_f, \mathcal{C}_s, \mathcal{H}, \mathcal{F}, \mathcal{G}, \mathcal{D}$ *are defined above.*

As observed in practice in Section 2.2.5, it is reasonable to assume that only the terms on the right hand side and the coupling terms of the form $\mathbf{c}(\boldsymbol{\mu}, \mathbf{v}(\overline{\mathbf{X}}))$ are inexactly computed, therefore we restrict our discussion to this particular case. Let us consider the following problem.

Problem 2.3.2. *Let* $\overline{\mathbf{X}} \in \mathbf{W}^{1,\infty}(\mathcal{B})$ *be invertible with Lipschitz inverse and* $\overline{\mathbf{u}} \in \mathbf{H}^1_0(\Omega)$ *such that* $\operatorname{div} \overline{\mathbf{u}} = 0$. *Given* $\mathbf{f} \in \mathbf{L}^2(\Omega)$, $\mathbf{g} \in \mathbf{L}^2(\mathcal{B})$ *and* $\mathbf{d} \in \mathbf{L}^2(\mathcal{B})$, *find* $(\mathbf{u}_h^\star, p_h^\star) \in \underline{\mathbf{V}}_h \times Q_h$, $\mathbf{X}_h^\star \in \underline{\mathbf{S}}_h$ *and* $\boldsymbol{\lambda}_h^\star \in \underline{\boldsymbol{\Lambda}}_h$, *such that*

$$\mathbf{a}_f(\mathbf{u}_h^\star, \mathbf{v}_h) - (\operatorname{div} \mathbf{v}_h, p_h^\star)_\Omega + \mathbf{c}_h(\boldsymbol{\lambda}_h^\star, \mathbf{v}_h(\overline{\mathbf{X}})) = (\mathbf{f}, \mathbf{v}_h)_{h,\Omega} \qquad \forall \mathbf{v}_h \in \underline{\mathbf{V}}_h \qquad (2.32a)$$

$$(\operatorname{div} \mathbf{u}_h^\star, q_h)_\Omega = 0 \qquad \forall q_h \in Q_h \qquad (2.32b)$$

$$\mathbf{a}_s(\mathbf{X}_h^\star, \mathbf{Y}_h) - \mathbf{c}(\boldsymbol{\lambda}_h^\star, \mathbf{Y}_h) = (\mathbf{g}, \mathbf{Y}_h)_{h,\mathcal{B}} \qquad \forall \mathbf{Y}_h \in \underline{\mathbf{S}}_h \qquad (2.32c)$$

$$\mathbf{c}_h(\boldsymbol{\mu}_h, \mathbf{u}_h^\star(\overline{\mathbf{X}})) - \mathbf{c}(\boldsymbol{\mu}_h, \mathbf{X}_h^\star) = \mathbf{c}_h(\boldsymbol{\mu}_h, \mathbf{d}) \qquad \forall \boldsymbol{\mu}_h \in \underline{\boldsymbol{\Lambda}}_h. \qquad (2.32d)$$

Still under Assumption 2.3.1, we can easily derive a corollary of Theorem 2.3.1 for the particular case of Problem 2.3.2.

Corollary 2.3.1. *Under Assumption 2.3.1, if* $(\mathbf{u}, p, \mathbf{X}, \boldsymbol{\lambda})$ *is solution of Problem 1.5.1 and* $(\mathbf{u}_h^\star, p_h^\star, \mathbf{X}_h^\star, \boldsymbol{\lambda}_h^\star)$ *is solution of Problem 2.3.2, then the following error estimate holds true*

$$\begin{aligned} \|\mathbf{u} - \mathbf{u}_h^\star\|_{1,\Omega} + \|p - p_h^\star\|_{0,\Omega} + \|\mathbf{X} - \mathbf{X}_h^\star\|_{1,\mathcal{B}} + \|\boldsymbol{\lambda} - \boldsymbol{\lambda}_h^\star\|_{\underline{\boldsymbol{\Lambda}}} \\ \leq \|\mathbf{u} - \mathbf{u}_h\|_{1,\Omega} + \|p - p_h\|_{0,\Omega} + \|\mathbf{X} - \mathbf{X}_h\|_{1,\mathcal{B}} + \|\boldsymbol{\lambda} - \boldsymbol{\lambda}_h\|_{\underline{\boldsymbol{\Lambda}}} \\ + M\big\{\mathcal{C}_f(\mathbf{u}_h) + \mathcal{C}_f^\top(\boldsymbol{\lambda}_h) + \mathcal{F} + \mathcal{G} + \mathcal{D}\big\}, \end{aligned} \qquad (2.33)$$

where $(\mathbf{u}_h, p_h, \mathbf{X}_h, \boldsymbol{\lambda}_h)$ *is solution of Problem 1.5.3 and* $\mathcal{C}_f, \mathcal{C}_f^\top, \mathcal{F}, \mathcal{G}, \mathcal{D}$ *are defined in* (2.29) *and* (2.30).

Let us observe that if $\mathbf{f}$, $\mathbf{g}$, and $\mathbf{d}$ are sufficiently regular and a quadrature rule is appropriately chosen, then the related consistency terms $\mathcal{F}, \mathcal{G}, \mathcal{D}$ can be estimated using classical finite element theory [41]. For this reason, in the next section, we are going to present quadrature error estimates only for the coupling terms $\mathcal{C}_f(\mathbf{u}_h)$ and $\mathcal{C}_f^\top(\boldsymbol{\lambda}_h)$.

Moreover, in Section 2.5, we will prove the inf-sup conditions stated in Assumption 2.3.1 for the particular case of Problem 2.3.2.

2.4 Error estimates for the inexact coupling term

In this section, we study quadrature error estimates for the coupling term $\mathbf{c}$ when assembled in approximate way. We focus our discussion on the two dimensional case with triangular meshes and we consider first order elements for the involved variables, i.e. velocity and Lagrange multiplier. Therefore, we set

$$\begin{aligned} \underline{\mathbf{V}}_h &= \{\mathbf{v} \in \mathbf{H}_0^1(\Omega) : \mathbf{v}_{|\mathrm{T}} \in (\mathcal{P}_1(\mathrm{T}))^2 \quad \forall \mathrm{T} \in \mathcal{T}_h^{\Omega}\} \\ \underline{\mathbf{\Lambda}}_h &= \{\boldsymbol{\mu} \in \mathbf{H}^1(\mathcal{B}) : \boldsymbol{\mu}_{|\mathrm{T}} \in (\mathcal{P}_1(\mathrm{T}))^2 \quad \forall \mathrm{T} \in \mathcal{T}_h^{\mathcal{B}}\} \end{aligned} \tag{2.34}$$

The choice of linear elements for the velocity is justified, for instance, by the Bercovier–Pironneau element we already used for the numerical investigation presented in Section 2.2. Moreover, we remind that in our discussion we are assuming that $\underline{\mathbf{S}}_h = \underline{\mathbf{\Lambda}}_h$.

In order to fix notation, before proving the mentioned error estimates, we recall the main features of affine finite elements and then we introduce the error functional $\mathcal{E}$ [41].

Let us consider the reference triangle $\widehat{\mathrm{T}}$. Given any generic finite element T, there exists a unique affine mapping

$$\begin{aligned} &F_{\mathrm{T}} : \widehat{\mathrm{T}} \longrightarrow \mathrm{T} \\ &F_{\mathrm{T}}(\widehat{\mathbf{x}}) = \mathsf{D}_{\mathrm{T}}\widehat{\mathbf{x}} + \mathsf{d}_{\mathrm{T}}, \end{aligned} \tag{2.35}$$

where D_{T} is an invertible 2×2 matrix and d_{T} a vector in $\mathbb{R}^2$. In particular, F_{T} maps the vertices of $\widehat{\mathrm{T}}$ to those of T. A schematic representation of the action of the map F_{T} is showed in Figure 2.14. In this setting, given a generic finite element function v defined on T, we have the following relation

$$\widehat{v}(\widehat{\mathbf{x}}) = v(F_{\mathrm{T}}(\widehat{\mathbf{x}})). \tag{2.36}$$

Moreover, we have

$$\|\mathsf{D}_{\mathrm{T}}\| \leq h_{\mathrm{T}}, \qquad \det \mathsf{D}_{\mathrm{T}} = \frac{|\mathrm{T}|}{|\widehat{\mathrm{T}}|}, \qquad |\mathrm{T}| \leq h_{\mathrm{T}}^2. \tag{2.37}$$

We now define the quadrature error functional.

Definition 2.4.1. *Given a generic function f and a quadrature rule with nodes and weights $\{(\mathbf{p}_k, \omega_k)\}_{k=1}^K$, the quadrature error functional $\mathcal{E}_{\mathrm{T}}$ over a generic element*

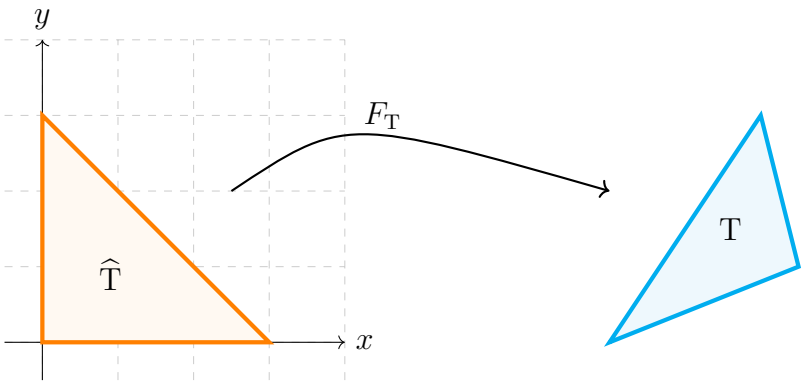


Figure 2.14: Example of affine finite element.

T *is defined as the difference between the exact integral and the numerical one, i.e.*

$$\mathcal{E}_{\mathrm{T}}(f) = \int_{\mathrm{T}} f(\mathbf{x})\,\mathrm{d}\mathbf{x} - |\mathrm{T}| \sum_{k=1}^{K} \omega_k f(\mathbf{p}_k). \tag{2.38}$$

We observe that a scaling argument for this functional holds true, indeed we have

$$\mathcal{E}_{\mathrm{T}}(f) = (\det \mathsf{D}_{\mathrm{T}})\, \widehat{\mathcal{E}}(\widehat{f}). \tag{2.39}$$

Moreover, we will denote by $\mathcal{E}_{\mathcal{B}}$ the quadrature error committed on the entire domain $\mathcal{B}$.

We start proving the following technical result.

Lemma 2.4.1. *Let us consider a triangle* $\mathrm{T} \in \mathcal{T}_h^{\mathcal{B}}$ *such that it is not included in an element of* $\mathcal{T}_h^{\Omega}$. *Let us assume that the map* $\overline{\mathbf{X}}$ *is linear in* T *so that* $\mathbf{v}_h(\overline{\mathbf{X}})$ *is continuous and piecewise linear in* T. *Then,* $\mathbf{v}_h(\overline{\mathbf{X}}) \in \mathbf{H}^{1+s}(\mathrm{T})$ *for* $0 \leq s < 1/2$ *and*

$$\| \boldsymbol{\nabla}_{\mathrm{s}}\, \mathbf{v}_h(\overline{\mathbf{X}})\|_{s,\mathrm{T}} \leq \frac{C}{1-2s} \| \boldsymbol{\nabla}_{\mathrm{s}}\, \mathbf{v}_h(\overline{\mathbf{X}})\|_{0,\mathrm{T}}.$$

Proof. We denote by Φ a generic component of $\boldsymbol{\nabla}_{\mathrm{s}}\, \mathbf{v}_h(\overline{\mathbf{X}})$. Since $\mathrm{T} \in \mathcal{T}_h^{\mathcal{B}}$ is chosen in such a way that $\overline{\mathbf{X}}(\mathrm{T})$ is not included in an element of $\mathcal{T}_h^{\Omega}$, we partition it in polygons P_j with $j = 1, \ldots, J$ so that

$$\mathrm{T} = \bigcup_{j=1}^{J} P_j$$

and $\mathbf{v}_h(\overline{\mathbf{X}})$ is linear in each P_j. We now denote by Φ_j the restriction of Φ in P_j, i.e. $\Phi_{|P_j}$. Thanks to the Sobolev inclusion $\mathrm{H}^1(P_j) \subset \mathrm{H}^s(P_j)$ for $0 \leq s < 1/2$, the extension by zero $\widetilde{\Phi_j}$ of $\Phi_j \in \mathrm{H}^s(P_j)$ belong to $\mathrm{H}^s(\mathrm{T})$ and the following estimate holds true

$$\|\widetilde{\Phi_j}\|_{s,\mathrm{T}} \leq \frac{C}{1-2s}\|\Phi_j\|_{s,P_j}.$$

This bound is proved applying [79, Chap. 1, Theo. 11.4] as done by Durán, Gastaldi and Lombardi in [52]. Consequently, $\Phi = \sum_{j=1}^{J} \widetilde{\Phi_j}$ belongs to $\mathrm{H}^s(\mathrm{T})$. Moreover, since Φ_j is constant, we can write

$$\|\Phi\|_{s,\mathrm{T}} \leq \frac{C}{1-2s}\left(\sum_{j=1}^{J}\|\Phi_j\|_{s,P_j}^2\right)^{1/2} = \frac{C}{1-2s}\left(\sum_{j=1}^{J}\|\Phi_j\|_{0,P_j}^2\right)^{1/2}$$

so that we have proved that $\boldsymbol{\nabla}_{\mathrm{s}}\,\mathbf{v}_h(\overline{\mathbf{X}}) \in \mathbf{H}^s(\mathrm{T})$ and

$$\|\,\boldsymbol{\nabla}_{\mathrm{s}}\,\mathbf{v}_h(\overline{\mathbf{X}})\|_{s,\mathrm{T}} \leq \frac{C}{1-2s}\|\,\boldsymbol{\nabla}_{\mathrm{s}}\,\mathbf{v}_h(\overline{\mathbf{X}})\|_{0,\mathrm{T}}.$$

□

With this tool, we are now able to prove quadrature error estimates for the $\mathbf{L}^2(\mathcal{B})$ scalar product and the $\mathbf{L}^2(\mathcal{B})$ scalar product of gradients.

Proposition 2.4.1. *Let us consider $\underline{\mathbf{V}}_h$ and $\underline{\boldsymbol{\Lambda}}_h$ as defined in* (2.34) *and assume $\overline{\mathbf{X}}$ to be linear. Given a quadrature rule $\{(\mathbf{p}_k^0, \omega_k^0)\}_{k=1}^{K_0}$ exact for quadratic polynomials, which means*

$$\widehat{\mathcal{E}}(\widehat{f}) = 0 \quad \forall \widehat{f} \in \mathcal{P}_2(\widehat{\mathrm{T}}), \tag{2.40}$$

the following estimate holds true

$$\left|\mathcal{E}_{\mathcal{B}}(\boldsymbol{\mu}_h \cdot \mathbf{v}_h(\overline{\mathbf{X}}))\right| \leq C h_{\mathcal{B}}^{3/2} |\log h_{\mathcal{B}}^{min}|\,\|\boldsymbol{\mu}_h\|_{0,\mathcal{B}}\,\|\mathbf{v}_h\|_{1,\Omega} \quad \forall \boldsymbol{\mu}_h \in \underline{\boldsymbol{\Lambda}}_h, \forall \mathbf{v}_h \in \underline{\mathbf{V}}_h, \tag{2.41}$$

where $h_{\mathcal{B}}^{min} = \min_{\mathrm{T}\in\mathcal{T}_h^{\mathcal{B}}} h_{\mathrm{T}}$.

Proof. The entire proof is done locally in an element $\mathrm{T} \in \mathcal{T}_h^{\mathcal{B}}$ and then the global estimate is derived summing on all the elements. Given $\mathrm{T} \in \mathcal{T}_h^{\mathcal{B}}$, let us notice that if T is included in an element of the fluid mesh $\mathcal{T}_h^{\Omega}$, then $\mathcal{E}_{\mathrm{T}}(\boldsymbol{\mu}_h \cdot \mathbf{v}_h(\overline{\mathbf{X}})) = 0$. Therefore, we can separate the elements of the solid mesh $\mathcal{T}_h^{\mathcal{B}}$ into two families

$$\begin{aligned} \mathcal{T}_{h,1}^{\mathcal{B}} &= \{\mathrm{T} \in \mathcal{T}_h^{\mathcal{B}} : \mathrm{T} \text{ is included in an element of } \mathcal{T}_h^{\Omega}\} \\ \mathcal{T}_{h,2}^{\mathcal{B}} &= \mathcal{T}_h^{\mathcal{B}} \setminus \mathcal{T}_{h,1}^{\mathcal{B}}, \end{aligned} \tag{2.42}$$

and consider $\mathrm{T} \in \mathcal{T}_{h,2}^{\mathcal{B}}$ because for the elements in $\mathcal{T}_{h,1}^{\mathcal{B}}$ no error occurs. Since $\mathbf{v}_h(\overline{\mathbf{X}})$ is piecewise linear in T, we introduce a decomposition of T into polygons, so that we have

$$\mathrm{T} = \bigcup_{j=1}^{J} P_j$$

and $\mathbf{v}_h(\overline{\mathbf{X}}) \in (\mathcal{P}_1(P_j))^2$ for $j = 1, \dots, J$. Notice that both $\boldsymbol{\mu}_h \in (\mathcal{P}_1(\mathrm{T}))^2$ and $\mathbf{v}_h(\overline{\mathbf{X}})$ are continuous in the element under consideration.

We now look for an estimate of the local quadrature error reading as

$$\mathcal{E}_{\mathrm{T}}(\boldsymbol{\mu}_h \cdot \mathbf{v}_h(\overline{\mathbf{X}})) = \int_{\mathrm{T}} \boldsymbol{\mu}_h \cdot \mathbf{v}_h(\overline{\mathbf{X}}) \, \mathrm{d}\mathbf{s} - |\mathrm{T}| \sum_{k=1}^{K_0} \omega_k^0 \, \boldsymbol{\mu}_h(\mathbf{p}_k^0) \cdot \mathbf{v}_h(\overline{\mathbf{X}}(\mathbf{p}_k^0)). \tag{2.43}$$

To this aim, we introduce the linear interpolant $\mathbf{v}_I \in (\mathcal{P}_1(\mathrm{T}))^2$ of $\mathbf{v}_h(\overline{\mathbf{X}})$ so that adding and subtracting equivalent terms from (2.43), we find

$$\begin{aligned} \mathcal{E}_{\mathrm{T}}(\boldsymbol{\mu}_h \cdot \mathbf{v}_h(\overline{\mathbf{X}})) = & \int_{\mathrm{T}} \boldsymbol{\mu}_h \cdot \big(\mathbf{v}_h(\overline{\mathbf{X}}) - \mathbf{v}_I\big) \, \mathrm{d}\mathbf{s} \\ & + \int_{\mathrm{T}} \boldsymbol{\mu}_h \cdot \mathbf{v}_I \, \mathrm{d}\mathbf{s} - |\mathrm{T}| \sum_{k=1}^{K_0} \omega_k^0 \, \boldsymbol{\mu}_h(\mathbf{p}_k^0) \cdot \mathbf{v}_I(\mathbf{p}_k^0) \\ & + |\mathrm{T}| \sum_{k=1}^{K_0} \omega_k^0 \, \boldsymbol{\mu}_h(\mathbf{p}_k^0) \cdot \big(\mathbf{v}_h(\overline{\mathbf{X}}(\mathbf{p}_k^0)) - \mathbf{v}_I(\mathbf{p}_k^0)\big) \end{aligned} \tag{2.44}$$

where each term can be studied separately.

Let us work on the first term. By Cauchy–Schwarz inequality and a classical interpolation result, we have

$$\begin{aligned} \left| \int_{\mathrm{T}} \boldsymbol{\mu}_h \cdot \big(\mathbf{v}_h(\overline{\mathbf{X}}) - \mathbf{v}_I\big) \, \mathrm{d}\mathbf{s} \right| &\le \|\boldsymbol{\mu}_h\|_{0,\mathrm{T}} \left\|\mathbf{v}_h(\overline{\mathbf{X}}) - \mathbf{v}_I\right\|_{0,\mathrm{T}} \\ &\le h_{\mathrm{T}}^{1+s} \, \|\boldsymbol{\mu}_h\|_{0,\mathrm{T}} \, |\mathbf{v}_h(\overline{\mathbf{X}})|_{1+s,\mathrm{T}} \end{aligned} \tag{2.45}$$

so that, applying Lemma 2.4.1, we finally obtain

$$\left| \int_{\mathrm{T}} \boldsymbol{\mu}_h \cdot \big(\mathbf{v}_h(\overline{\mathbf{X}}) - \mathbf{v}_I\big) \, \mathrm{d}\mathbf{s} \right| \le \frac{h_{\mathrm{T}}^{1+s}}{1-2s} \, \|\boldsymbol{\mu}_h\|_{0,\mathrm{T}} \, |\mathbf{v}_h(\overline{\mathbf{X}})|_{1,\mathrm{T}}. \tag{2.46}$$

The second term is nothing else that the quadrature error of the product $\boldsymbol{\mu}_h \cdot \mathbf{v}_I$: since both functions are linear in T, thanks to the choice of quadrature rule, we have

$$\mathcal{E}_{\mathrm{T}}(\boldsymbol{\mu}_h \cdot \mathbf{v}_I) = \int_{\mathrm{T}} \boldsymbol{\mu}_h \cdot \mathbf{v}_I \, \mathrm{d}\mathbf{s} - |\mathrm{T}| \sum_{k=1}^{K_0} \omega_k^0 \, \boldsymbol{\mu}_h(\mathbf{p}_k^0) \cdot \mathbf{v}_I(\mathbf{p}_k^0) = 0 \tag{2.47}$$

For the third term, we start applying the discrete Cauchy–Schwarz inequality; then, exploiting the precision of the quadrature rule under consideration, we can easily write

$$\begin{aligned}
&\left| |\mathrm{T}| \sum_{k=1}^{K_0} \omega_k^0 \, \boldsymbol{\mu}_h(\mathbf{p}_k^0) \cdot \left(\mathbf{v}_h(\overline{\mathbf{X}}(\mathbf{p}_k^0)) - \mathbf{v}_I(\mathbf{p}_k^0) \right) \right| \\
&\quad \leq \left(|\mathrm{T}| \sum_{k=1}^{K_0} \omega_k^0 \, \boldsymbol{\mu}_h(\mathbf{p}_k^0)^2 \right)^{1/2} \left(|\mathrm{T}| \sum_{k=1}^{K_0} \omega_k^0 \left(\mathbf{v}_h(\overline{\mathbf{X}}(\mathbf{p}_k^0)) - \mathbf{v}_I(\mathbf{p}_k^0) \right) \right)^{1/2} \\
&\quad = \left(\int_{\mathrm{T}} |\boldsymbol{\mu}_h|^2 \right)^{1/2} \left(|\mathrm{T}| \sum_{k=1}^{K_0} \omega_k^0 \left(\mathbf{v}_h(\overline{\mathbf{X}}(\mathbf{p}_k^0)) - \mathbf{v}_I(\mathbf{p}_k^0) \right) \right)^{1/2} \\
&\quad \leq K_0 \, |\mathrm{T}|^{1/2} \, \|\boldsymbol{\mu}_h\|_{0,\mathrm{T}} \left\| \mathbf{v}_h(\overline{\mathbf{X}}) - \mathbf{v}_I \right\|_{\infty,\mathrm{T}} .
\end{aligned} \tag{2.48}$$

In order to estimate $\left\| \mathbf{v}_h(\overline{\mathbf{X}}) - \mathbf{v}_I \right\|_{\infty,\mathrm{T}}$, we make use of some standard arguments in finite elements theory [41]. Indeed, moving to the reference element $\widehat{\mathrm{T}}$ through the affine map F_{T} defined in (2.35), we have

$$\left\| \mathbf{v}_h(\overline{\mathbf{X}}) - \mathbf{v}_I \right\|_{\infty,\mathrm{T}} \leq \left\| \widehat{\mathbf{v}_h(\overline{\mathbf{X}})} - \widehat{\mathbf{v}}_I \right\|_{\infty,\widehat{\mathrm{T}}} .$$

Now, since $\mathbf{v}_h(\overline{\mathbf{X}}) \in \mathbf{H}^{1+s}(\mathrm{T})$, we can exploit the inclusion $\mathrm{H}^{1+s}(\widehat{\mathrm{T}}) \subset \mathrm{L}^\infty(\widehat{\mathrm{T}})$ so that

$$\begin{aligned}
\left\| \widehat{\mathbf{v}_h(\overline{\mathbf{X}})} - \widehat{\mathbf{v}}_I \right\|_{\infty,\widehat{\mathrm{T}}} &\leq \left\| \mathrm{I} - \widehat{\Pi} \right\|_{\mathcal{L}(\mathbf{H}^{1+s}(\widehat{\mathrm{T}}),\mathbf{L}^\infty(\widehat{\mathrm{T}}))} \inf_{\widehat{\mathbf{q}} \in [\mathcal{P}_1(\widehat{\mathrm{T}})]^2} \left\| \widehat{\mathbf{v}_h(\overline{\mathbf{X}})} + \widehat{\mathbf{q}} \right\|_{1+s,\widehat{\mathrm{T}}} \\
&\leq C \left| \widehat{\mathbf{v}_h(\overline{\mathbf{X}})} \right|_{1+s,\widehat{\mathrm{T}}} ,
\end{aligned} \tag{2.49}$$

where $\mathcal{L}(\mathbf{H}^{1+s}(\widehat{\mathrm{T}}), \mathbf{L}^\infty(\widehat{\mathrm{T}}))$ denotes the space of linear functionals from $\mathbf{H}^{1+s}(\widehat{\mathrm{T}})$ to $\mathbf{L}^\infty(\widehat{\mathrm{T}})$ and $\widehat{\mathbf{v}_h(\overline{\mathbf{X}}(\widehat{\mathbf{s}}))} = \mathbf{v}_h(\overline{\mathbf{X}}(F_{\mathrm{T}}(\widehat{\mathbf{s}})))$. Moreover, I is the identity operator and $\widehat{\Pi}$ is the interpolation operator in $(\mathcal{P}_1(\widehat{\mathrm{T}}))^2$. An estimate for the last term can be found applying the definition of fractional Sobolev semi-norm and following the theory by Dupont and Scott [51]. In particular, we have

$$\begin{aligned}
&\left| \widehat{\mathbf{v}_h(\overline{\mathbf{X}})} \right|^2_{1+s,\widehat{\mathrm{T}}} = \left| \boldsymbol{\nabla}_{\mathrm{s}} \widehat{\mathbf{v}_h(\overline{\mathbf{X}})} \right|^2_{s,\widehat{\mathrm{T}}} = \int_{\widehat{\mathrm{T}}} \int_{\widehat{\mathrm{T}}} \frac{\left| \boldsymbol{\nabla}_{\mathrm{s}} \widehat{\mathbf{v}_h(\overline{\mathbf{X}}(\widehat{\mathbf{s}}_1))} - \boldsymbol{\nabla}_{\mathrm{s}} \widehat{\mathbf{v}_h(\overline{\mathbf{X}}(\widehat{\mathbf{s}}_2))} \right|^2}{|\widehat{\mathbf{s}}_1 - \widehat{\mathbf{s}}_2|^{2(s+1)}} \, \mathrm{d}\widehat{\mathbf{s}}_1 \mathrm{d}\widehat{\mathbf{s}}_2 \\
&= |\det \mathsf{D}_{\mathrm{T}}|^{-2} \int_{\mathrm{T}} \int_{\mathrm{T}} \frac{\left| \mathsf{D}_{\mathrm{T}} \left(\boldsymbol{\nabla}_{\mathrm{s}} \mathbf{v}_h(\overline{\mathbf{X}}(\mathbf{s}_1)) - \boldsymbol{\nabla}_{\mathrm{s}} \mathbf{v}_h(\overline{\mathbf{X}}(\mathbf{s}_2)) \right) \right|^2}{|\mathbf{s}_1 - \mathbf{s}_2|^{2(s+1)}} \left(\frac{|\mathbf{s}_1 - \mathbf{s}_2|}{\left| \mathsf{D}_{\mathrm{T}}^{-1}(\mathbf{s}_1 - \mathbf{s}_2) \right|} \right)^{2(s+1)} \mathrm{d}\mathbf{s}_1 \mathrm{d}\mathbf{s}_2 \\
&\leq \frac{\|\mathsf{D}_{\mathrm{T}}\|^{4+2s}}{|\det \mathsf{D}_{\mathrm{T}}|^2} \left| \boldsymbol{\nabla}_{\mathrm{s}} \mathbf{v}_h(\overline{\mathbf{X}}) \right|^2_{s,\mathrm{T}} ;
\end{aligned}$$

together with (2.37), this implies

$$\left|\widehat{\mathbf{v}_h(\overline{\mathbf{X}})}\right|^2_{1+s,\widehat{\mathrm{T}}} \leq h_{\mathrm{T}}^{2s} \left\|\boldsymbol{\nabla}_{\mathrm{s}}\, \mathbf{v}_h(\overline{\mathbf{X}})\right\|^2_{s,\mathrm{T}}. \tag{2.50}$$

Thanks again to Lemma 2.4.1, the third term is bounded as follows

$$\begin{aligned}\left||\mathrm{T}| \sum_{k=1}^{K_0} \omega_k\, \boldsymbol{\mu}_h(\mathbf{p}_k^0) \cdot \left(\mathbf{v}_h(\overline{\mathbf{X}}(\mathbf{p}_k^0)) - \mathbf{v}_I(\mathbf{p}_k^0)\right)\right| &\leq \frac{C}{1-2s}\, |\mathrm{T}|^{1/2}\, h_{\mathrm{T}}^s \left\|\boldsymbol{\mu}_h\right\|_{0,\mathrm{T}} \left\|\mathbf{v}_h(\overline{\mathbf{X}})\right\|_{1,\mathrm{T}} \\ &\leq C \frac{h_{\mathrm{T}}^{1+s}}{1-2s} \left\|\boldsymbol{\mu}_h\right\|_{0,\mathrm{T}} \left\|\boldsymbol{\nabla}_{\mathrm{s}}\, \mathbf{v}_h(\overline{\mathbf{X}})\right\|_{0,\mathrm{T}}.\end{aligned} \tag{2.51}$$

Putting together (2.46), (2.47) and (2.51), the local estimate reads

$$\left|\mathcal{E}_{\mathrm{T}}(\boldsymbol{\mu}_h \cdot \mathbf{v}_h(\overline{\mathbf{X}}))\right| \leq C \frac{h_{\mathrm{T}}^{1+s}}{1-2s} \left\|\boldsymbol{\mu}_h\right\|_{0,\mathrm{T}} \left\|\mathbf{v}_h(\overline{\mathbf{X}})\right\|_{1,\mathrm{T}} \tag{2.52}$$

for $0 \leq s < 1/2$. In particular, considering $s = \frac{1}{2} + \frac{1}{\log h_{\mathrm{T}}}$, by means of some easy manipulations, we have

$$\left|\mathcal{E}_{\mathrm{T}}(\boldsymbol{\mu}_h \cdot \mathbf{v}_h(\overline{\mathbf{X}}))\right| \leq C h_{\mathrm{T}}^{3/2} |\log h_{\mathrm{T}}| \left\|\boldsymbol{\mu}_h\right\|_{0,\mathrm{T}} \left\|\mathbf{v}_h(\overline{\mathbf{X}})\right\|_{1,\mathrm{T}}.$$

At this point, summing on all the triangles in $\mathcal{T}_{h,2}^{\mathcal{B}}$ and exploiting the inclusion $\overline{\mathbf{X}}(\mathcal{B}) \subset \Omega$, we finally obtain (2.41). □

Proposition 2.4.2. *Let us consider $\underline{\mathbf{V}}_h$ and $\underline{\boldsymbol{\Lambda}}_h$ as defined in* (2.34) *and assume $\overline{\mathbf{X}}$ to be linear. Given a quadrature rule $\{(\mathbf{p}_k^1, \omega_k^1)\}_{k=1}^{K_1}$ exact for constants, which means*

$$\widehat{\mathcal{E}}(\widehat{f}) = 0 \quad \forall \widehat{f} \in \mathcal{P}_0(\widehat{\mathrm{T}}) \tag{2.53}$$

and a quasi uniform mesh $\mathcal{T}_h^{\Omega}$, the following estimate holds true

$$\left|\mathcal{E}_{\mathcal{B}}(\boldsymbol{\nabla}_{\mathrm{s}}\, \boldsymbol{\mu}_h : \boldsymbol{\nabla}_{\mathrm{s}}\, \mathbf{v}_h(\overline{\mathbf{X}}))\right| \leq C \left(h_{\mathcal{B}}^{1/2} |\log h_{\mathcal{B}}^{min}| + \frac{h_{\mathcal{B}}}{h_{\Omega}} \right) \left\|\boldsymbol{\nabla}_{\mathrm{s}}\, \boldsymbol{\mu}_h\right\|_{0,\mathcal{B}} \left\|\boldsymbol{\nabla}\, \mathbf{v}_h\right\|_{0,\Omega} \tag{2.54}$$

for all $\boldsymbol{\mu}_h \in \underline{\boldsymbol{\Lambda}}_h$, $\mathbf{v}_h \in \underline{\mathbf{V}}_h$.

Proof. We start dividing the elements of the solid mesh $\mathcal{T}_h^{\mathcal{B}}$ into two families as already done in (2.42). We then work locally in $\mathrm{T} \in \mathcal{T}_{h,2}^{\mathcal{B}}$ with the aim of finding a bound for the local quadrature error

$$\begin{aligned}\mathcal{E}_{\mathrm{T}}(\boldsymbol{\nabla}_{\mathrm{s}}\, \boldsymbol{\mu}_h : \boldsymbol{\nabla}_{\mathrm{s}}\, \mathbf{v}_h(\overline{\mathbf{X}})) = &\int_{\mathrm{T}} \boldsymbol{\nabla}_{\mathrm{s}}\, \boldsymbol{\mu}_h : \boldsymbol{\nabla}_{\mathrm{s}}\, \mathbf{v}_h(\overline{\mathbf{X}})\, \mathrm{d}\mathbf{s} \\ &- |\mathrm{T}| \sum_{k=1}^{K_1} \omega_k^1\, \boldsymbol{\nabla}_{\mathrm{s}}\, \boldsymbol{\mu}_h(\mathbf{p}_k^1) : \boldsymbol{\nabla}_{\mathrm{s}}\, \mathbf{v}_h(\overline{\mathbf{X}}(\mathbf{p}_k^1)).\end{aligned} \tag{2.55}$$

Let us notice that $\boldsymbol{\nabla}_{\mathrm{s}}\boldsymbol{\mu}_h$ is constant in T, whereas $\boldsymbol{\nabla}_{\mathrm{s}}\mathbf{v}_h(\overline{\mathbf{X}})$ is discontinuous piecewise constant in T. Introducing again the linear interpolant $\mathbf{v}_I$ of $\mathbf{v}_h(\overline{\mathbf{X}})$, (2.55) can be reformulated as

$$\begin{aligned}\mathcal{E}_{\mathrm{T}}(\boldsymbol{\nabla}_{\mathrm{s}}\boldsymbol{\mu}_h : \boldsymbol{\nabla}_{\mathrm{s}}\mathbf{v}_h(\overline{\mathbf{X}})) = &\int_{\mathrm{T}} \boldsymbol{\nabla}_{\mathrm{s}}\boldsymbol{\mu}_h : \big(\boldsymbol{\nabla}_{\mathrm{s}}\mathbf{v}_h(\overline{\mathbf{X}}) - \boldsymbol{\nabla}_{\mathrm{s}}\mathbf{v}_I\big)\,\mathrm{d}\mathbf{s}\\ &+ \int_{\mathrm{T}} \boldsymbol{\nabla}_{\mathrm{s}}\boldsymbol{\mu}_h : \boldsymbol{\nabla}_{\mathrm{s}}\mathbf{v}_I\,\mathrm{d}\mathbf{s} - |\mathrm{T}|\sum_{k=1}^{K_1}\omega_k^1\,\boldsymbol{\nabla}_{\mathrm{s}}\boldsymbol{\mu}_h(\mathbf{p}_k^1) : \boldsymbol{\nabla}_{\mathrm{s}}\mathbf{v}_I(\mathbf{p}_k^1)\\ &+ |\mathrm{T}|\sum_{k=1}^{K_1}\omega_k^1\,\boldsymbol{\nabla}_{\mathrm{s}}\boldsymbol{\mu}_h(\mathbf{p}_k^1) : \big(\boldsymbol{\nabla}_{\mathrm{s}}\mathbf{v}_h(\overline{\mathbf{X}}(\mathbf{p}_k^1)) - \boldsymbol{\nabla}_{\mathrm{s}}\mathbf{v}_I(\mathbf{p}_k^1)\big),\end{aligned} \tag{2.56}$$

so that each term can be studied independently from the others.

Looking at the first term, we can apply the Cauchy–Schwarz inequality to obtain

$$\begin{aligned}&\left|\int_{\mathrm{T}} \boldsymbol{\nabla}_{\mathrm{s}}\boldsymbol{\mu}_h : \big(\boldsymbol{\nabla}_{\mathrm{s}}\mathbf{v}_h(\overline{\mathbf{X}}) - \boldsymbol{\nabla}_{\mathrm{s}}\mathbf{v}_I\big)\,\mathrm{d}\mathbf{s}\right|\\ &\qquad\le \|\boldsymbol{\nabla}_{\mathrm{s}}\boldsymbol{\mu}_h\|_{0,\mathrm{T}}\left\|\boldsymbol{\nabla}_{\mathrm{s}}\mathbf{v}_h(\overline{\mathbf{X}}) - \boldsymbol{\nabla}_{\mathrm{s}}\mathbf{v}_I\right\|_{0,\mathrm{T}}.\end{aligned} \tag{2.57}$$

Exploiting Lemma 2.4.1, we can write

$$\left\|\boldsymbol{\nabla}_{\mathrm{s}}\mathbf{v}_h(\overline{\mathbf{X}}) - \boldsymbol{\nabla}_{\mathrm{s}}\mathbf{v}_I\right\|_{0,\mathrm{T}} \le h_{\mathrm{T}}^s\left\|\boldsymbol{\nabla}_{\mathrm{s}}\mathbf{v}_h(\overline{\mathbf{X}})\right\|_{s,\mathrm{T}} \le \frac{h_{\mathrm{T}}^s}{1-2s}\left\|\boldsymbol{\nabla}_{\mathrm{s}}\mathbf{v}_h(\overline{\mathbf{X}})\right\|_{0,\mathrm{T}}$$

so that, setting $s = \frac{1}{2} + \frac{1}{\log h_{\mathrm{T}}}$, we find

$$\begin{aligned}&\left|\int_{\mathrm{T}} \boldsymbol{\nabla}_{\mathrm{s}}\boldsymbol{\mu}_h : \big(\boldsymbol{\nabla}_{\mathrm{s}}\mathbf{v}_h(\overline{\mathbf{X}}) - \boldsymbol{\nabla}_{\mathrm{s}}\mathbf{v}_I\big)\,\mathrm{d}\mathbf{s}\right|\\ &\qquad\le Ch_{\mathrm{T}}^{1/2}\,|\log h_{\mathrm{T}}|\,\|\boldsymbol{\nabla}_{\mathrm{s}}\boldsymbol{\mu}_h\|_{0,\mathrm{T}}\left\|\boldsymbol{\nabla}_{\mathrm{s}}\mathbf{v}_h(\overline{\mathbf{X}})\right\|_{0,\mathrm{T}}.\end{aligned} \tag{2.58}$$

The second term is the quadrature error for the product $\boldsymbol{\nabla}_{\mathrm{s}}\boldsymbol{\mu}_h : \boldsymbol{\nabla}_{\mathrm{s}}\mathbf{v}_I$, which is zero since the integrand function is constant, therefore

$$\begin{aligned}\mathcal{E}_{\mathrm{T}}(\boldsymbol{\nabla}_{\mathrm{s}}\boldsymbol{\mu}_h : \boldsymbol{\nabla}_{\mathrm{s}}\mathbf{v}_I) = &\int_{\mathrm{T}} \boldsymbol{\nabla}_{\mathrm{s}}\boldsymbol{\mu}_h : \boldsymbol{\nabla}_{\mathrm{s}}\mathbf{v}_I\,\mathrm{d}\mathbf{s}\\ &- |\mathrm{T}|\sum_{k=1}^{K_1}\omega_k^1\,\boldsymbol{\nabla}_{\mathrm{s}}\boldsymbol{\mu}_h(\mathbf{p}_k^1) : \boldsymbol{\nabla}_{\mathrm{s}}\mathbf{v}_I(\mathbf{p}_k^1) = 0.\end{aligned} \tag{2.59}$$

The last term can be treated by Cauchy–Schwarz inequality and taking into

account the precision of the quadrature rule as in (2.48), we get

$$\begin{aligned}\left|\,|\mathrm{T}|\sum_{k=1}^{K_1}\omega_k^1\,\boldsymbol{\nabla}_{\mathrm{s}}\,\boldsymbol{\mu}_h(\mathbf{p}_k^1):\big(\,\boldsymbol{\nabla}_{\mathrm{s}}\,\mathbf{v}_h(\overline{\mathbf{X}}(\mathbf{p}_k^1))-\boldsymbol{\nabla}_{\mathrm{s}}\,\mathbf{v}_I(\mathbf{p}_k^1)\big)\right| \\ \leq K_1\,|\mathrm{T}|^{1/2}\,\|\boldsymbol{\nabla}_{\mathrm{s}}\,\boldsymbol{\mu}_h\|_{0,\mathrm{T}}\,\big\|\boldsymbol{\nabla}_{\mathrm{s}}\,\mathbf{v}_h(\overline{\mathbf{X}})-\boldsymbol{\nabla}_{\mathrm{s}}\,\mathbf{v}_I\big\|_{\infty,\mathrm{T}}\,.\end{aligned} \tag{2.60}$$

We need to estimate the pointwise error $\big\|\boldsymbol{\nabla}_{\mathrm{s}}\,\mathbf{v}_h(\overline{\mathbf{X}})-\boldsymbol{\nabla}_{\mathrm{s}}\,\mathbf{v}_I\big\|_{\infty,\mathrm{T}}$, which requires attention since $\boldsymbol{\nabla}_{\mathrm{s}}\,\mathbf{v}_h(\overline{\mathbf{X}})$ is discontinuous in T. To this aim, we partition the triangle into polygons P_j for $j=1,\ldots,N$, so that $\mathrm{T}=\bigcup_{j=1}^{J}P_j$ and $\boldsymbol{\nabla}_{\mathrm{s}}\,\mathbf{v}_h(\overline{\mathbf{X}})$ is constant in each P_j. Therefore, we have

$$\big\|\boldsymbol{\nabla}_{\mathrm{s}}\,\mathbf{v}_h(\overline{\mathbf{X}})-\boldsymbol{\nabla}_{\mathrm{s}}\,\mathbf{v}_I\big\|_{\infty,\mathrm{T}}=\max_{j=1,\ldots,J}\big\|\boldsymbol{\nabla}_{\mathrm{s}}\,\mathbf{v}_h(\overline{\mathbf{X}})-\boldsymbol{\nabla}_{\mathrm{s}}\,\mathbf{v}_I\big\|_{\infty,P_j}$$

and, using $\|\boldsymbol{\nabla}_{\mathrm{s}}\,\mathbf{v}_I\|_{\infty,\mathrm{T}}\leq\big\|\boldsymbol{\nabla}_{\mathrm{s}}\,\mathbf{v}_h(\overline{\mathbf{X}})\big\|_{\infty,\mathrm{T}}$, we obtain

$$\big\|\boldsymbol{\nabla}_{\mathrm{s}}\,\mathbf{v}_h(\overline{\mathbf{X}})-\boldsymbol{\nabla}_{\mathrm{s}}\,\mathbf{v}_I\big\|_{\infty,\mathrm{T}}\leq 2\max_{j=1,\ldots,J}\big\|\boldsymbol{\nabla}_{\mathrm{s}}\,\mathbf{v}_h(\overline{\mathbf{X}})\big\|_{\infty,P_j}\,. \tag{2.61}$$

In order to treat $\big\|\boldsymbol{\nabla}_{\mathrm{s}}\,\mathbf{v}_h(\overline{\mathbf{X}})\big\|_{\infty,P_j}$, we use a macroelement technique. Notice that each polygon P_j is also defined by intersecting $\overline{\mathbf{X}}(\mathrm{T})$ with a fluid element $\mathrm{T}_{f,j}\in\mathcal{T}_h^{\Omega}$, this is equivalent to write

$$P_j=\overline{\mathbf{X}}^{-1}\big(\overline{\mathbf{X}}(\mathrm{T})\cap\mathrm{T}_{f,j}\big).$$

Applying an inverse inequality and introducing $\mathfrak{O}_{\mathrm{T}}=\bigcup_{j=1}^{J}\mathrm{T}_{f,j}$, which is the macroelement of all the fluid triangles $\mathrm{T}_{f,j}\in\mathcal{T}_h^{\Omega}$ such that $\overline{\mathbf{X}}(\mathrm{T})\cap\mathrm{T}_{f,j}\neq\emptyset$, we can write

$$\begin{aligned}\max_{j=1,\ldots,J}\big\|\boldsymbol{\nabla}_{\mathrm{s}}\,\mathbf{v}_h(\overline{\mathbf{X}})\big\|_{\infty,P_j}&\leq\max_{j=1,\ldots,J}\|\boldsymbol{\nabla}\,\mathbf{v}_h\|_{\infty,\mathrm{T}_{f,j}} \\ &\leq\max_{j=1,\ldots,J}\frac{1}{h_{\mathrm{T}_{f,j}}}\,\|\boldsymbol{\nabla}\,\mathbf{v}_h\|_{0,\mathrm{T}_{f,j}}\leq\frac{1}{h_{\Omega}}\,\|\boldsymbol{\nabla}\,\mathbf{v}_h\|_{0,\mathfrak{O}_{\mathrm{T}}}\,.\end{aligned} \tag{2.62}$$

Hence,

$$\begin{aligned}\left|\,|\mathrm{T}|\sum_{k=1}^{K_1}\omega_k^1\,\boldsymbol{\nabla}_{\mathrm{s}}\,\boldsymbol{\mu}_h(\mathbf{p}_k^1):\big(\,\boldsymbol{\nabla}_{\mathrm{s}}\,\mathbf{v}_h(\overline{\mathbf{X}}(\mathbf{p}_k^1))-\boldsymbol{\nabla}_{\mathrm{s}}\,\mathbf{v}_I(\mathbf{p}_k^1)\big)\right| \\ \leq\frac{|\mathrm{T}|^{1/2}}{h_{\Omega}}\,\|\boldsymbol{\nabla}_{\mathrm{s}}\,\boldsymbol{\mu}_h\|_{0,\mathrm{T}}\,\|\boldsymbol{\nabla}\,\mathbf{v}_h\|_{0,\mathfrak{O}_{\mathrm{T}}}\;\leq\frac{h_{\mathcal{B}}}{h_{\Omega}}\,\|\boldsymbol{\nabla}_{\mathrm{s}}\,\boldsymbol{\mu}_h\|_{0,\mathrm{T}}\,\|\boldsymbol{\nabla}\,\mathbf{v}_h\|_{0,\mathfrak{O}_{\mathrm{T}}}\,.\end{aligned} \tag{2.63}$$

Putting together (2.58), (2.59) and (2.63), the local estimate reads

$$\mathcal{E}_{\mathrm{T}}(\boldsymbol{\nabla}_{\mathrm{s}}\,\boldsymbol{\mu}_h : \boldsymbol{\nabla}_{\mathrm{s}}\,\mathbf{v}_h(\overline{\mathbf{X}})) \leq C\left(h_{\mathrm{T}}^{1/2}\,|\log h_{\mathrm{T}}| + \frac{h_{\mathcal{B}}}{h_\Omega}\right)\|\boldsymbol{\mu}_h\|_{0,\mathrm{T}}\left\|\mathbf{v}_h(\overline{\mathbf{X}})\right\|_{1,\mathrm{T}} \tag{2.64}$$

so that, summing on over all $\mathrm{T} \in \mathcal{T}_{h,2}^{\mathcal{B}}$, and using $\overline{\mathbf{X}}(\mathcal{B}) \subset \Omega$, the global estimate (2.54) is obtained.

□

The following results for the coupling term are direct consequence of the two propositions we have just proved.

Proposition 2.4.3. *With the same hypotheses of Proposition 2.4.1, if the coupling bilinear form is defined as*

$$\mathbf{c}(\boldsymbol{\mu}_h, \mathbf{Y}_h) = (\boldsymbol{\mu}_h, \mathbf{Y}_h)_{\mathcal{B}} \quad \forall \boldsymbol{\mu}_h \in \underline{\boldsymbol{\Lambda}}_h, \forall \mathbf{Y}_h \in \underline{\mathbf{S}}_h,$$

then the following quadrature error estimate holds true

$$\left|\mathbf{c}(\boldsymbol{\mu}_h, \mathbf{v}_h(\overline{\mathbf{X}})) - \mathbf{c}_h(\boldsymbol{\mu}_h, \mathbf{v}_h(\overline{\mathbf{X}}))\right| \leq C h_{\mathcal{B}}^{3/2} |\log h_{\mathcal{B}}^{min}|\,\|\boldsymbol{\mu}_h\|_{0,\mathcal{B}}\,\|\mathbf{v}_h\|_{1,\Omega} \tag{2.65}$$

for all $\boldsymbol{\mu}_h \in \underline{\boldsymbol{\Lambda}}_h$, $\mathbf{v}_h \in \underline{\mathbf{V}}_h$.

Remark 2.4.1. *This estimate suggests that the* $\mathbf{L}^2(\mathcal{B})$ *coupling term produces an optimal method also when assembled without mesh intersection provided that the solid mesh size* $h_{\mathcal{B}}$ *decreases to zero. This suggestion will be confirmed in the numerical tests we are going to show in the next section. In any case, we should remember that the use of the* $\mathbf{L}^2(\mathcal{B})$ *scalar product at discrete level is a replacement of the duality pairing, which means that at continuous level* $\boldsymbol{\mu} \in (\mathbf{H}^1(\mathcal{B}))'$. *In order to derive the dual norm in the right hand side of* (2.65)*, we should apply an inverse estimate, finally obtaining a factor* $h_{\mathcal{B}}^{1/2}|\log h_{\mathcal{B}}^{min}|$.

Proposition 2.4.4. *With the same hypotheses of Propositions 2.4.1 and 2.4.2, if the coupling bilinear form is defined as*

$$\mathbf{c}(\boldsymbol{\mu}_h, \mathbf{Y}_h) = (\boldsymbol{\mu}_h, \mathbf{Y}_h)_{\mathcal{B}} + (\boldsymbol{\nabla}_{\mathrm{s}}\,\boldsymbol{\mu}_h, \boldsymbol{\nabla}_{\mathrm{s}}\,\mathbf{Y}_h)_{\mathcal{B}} \quad \forall \boldsymbol{\mu}_h \in \underline{\boldsymbol{\Lambda}}_h, \forall \mathbf{Y}_h \in \underline{\mathbf{S}}_h,$$

then the following quadrature error estimate holds true

$$\begin{aligned}&\left|\mathbf{c}(\boldsymbol{\mu}_h, \mathbf{v}_h(\overline{\mathbf{X}})) - \mathbf{c}_h(\boldsymbol{\mu}_h, \mathbf{v}_h(\overline{\mathbf{X}}))\right| \\ &\qquad \leq C\left(\left(h_{\mathcal{B}}^{3/2} + h_{\mathcal{B}}^{1/2}\right)|\log h_{\mathcal{B}}^{min}| + \frac{h_{\mathcal{B}}}{h_\Omega}\right)\|\boldsymbol{\mu}_h\|_{1,\mathcal{B}}\,\|\mathbf{v}_h\|_{1,\Omega}\end{aligned} \tag{2.66}$$

for all $\boldsymbol{\mu}_h \in \underline{\boldsymbol{\Lambda}}_h$, $\mathbf{v}_h \in \underline{\mathbf{V}}_h$.

Remark 2.4.2. *The estimate in Proposition 2.4.4 is perfectly aligned with the results we obtained in the numerical investigation we presented in Section 2.2. Indeed, all the tests were performed maintaining the ratio $h_{\mathcal{B}}/h_\Omega$ constant and showed a loss of convergence for the method assembled without mesh intersection. This theoretical result says that both $h_{\mathcal{B}}$ and $h_{\mathcal{B}}/h_\Omega$ are required to decrease to zero if we want to obtain an optimal method with approximate integration of the $\mathbf{H}^1(\mathcal{B})$ scalar product.*

2.4.1 Numerical tests

In this section we present two numerical tests with a twofold scope: we validate the theoretical estimates we have just proved and we compare the behavior of the $\mathbf{L}^2(\mathcal{B})$ and $\mathbf{H}^1(\mathcal{B})$ coupling terms. In particular, we are going to solve the following stationary problem, already considered in Section 2.2 (see Test 8), with two combinations of $h_{\mathcal{B}}/h_\Omega$.

Problem 2.4.1. *Let $\overline{\mathbf{X}} \in \mathbf{W}^{1,\infty}(\mathcal{B})$ be invertible with Lipschitz inverse. Given $\mathbf{f} \in \mathbf{L}^2(\Omega)$, $\mathbf{g} \in \mathbf{L}^2(\mathcal{B})$ and $\mathbf{d} \in \mathbf{L}^2(\mathcal{B})$, find $(\mathbf{u}, p) \in \mathbf{H}^1_0(\Omega) \times \mathrm{L}^2_0(\Omega)$, $\mathbf{X} \in \mathbf{H}^1(\mathcal{B})$ and $\boldsymbol{\lambda} \in \underline{\boldsymbol{\Lambda}}$, such that*

$$
\begin{aligned}
&(\boldsymbol{\nabla}\,\mathbf{u}, \boldsymbol{\nabla}\,\mathbf{v})_\Omega - (\operatorname{div}\mathbf{v}, p)_\Omega + \mathbf{c}(\boldsymbol{\lambda}, \mathbf{v}(\overline{\mathbf{X}})) = (\mathbf{f}, \mathbf{v})_\Omega && \forall \mathbf{v} \in \mathbf{H}^1_0(\Omega) && \text{(2.67a)}\\
&(\operatorname{div}\mathbf{u}, q)_\Omega = 0 && \forall q \in \mathrm{L}^2_0(\Omega) && \text{(2.67b)}\\
&(\boldsymbol{\nabla}_{\mathrm{s}}\,\mathbf{X}, \boldsymbol{\nabla}_{\mathrm{s}}\,\mathbf{Y})_{\mathcal{B}} - \mathbf{c}(\boldsymbol{\lambda}, \mathbf{Y}) = (\mathbf{g}, \mathbf{Y})_{\mathcal{B}} && \forall \mathbf{Y} \in \mathbf{H}^1(\mathcal{B}) && \text{(2.67c)}\\
&\mathbf{c}(\boldsymbol{\mu}, \mathbf{u}(\overline{\mathbf{X}}) - \mathbf{X}) = \mathbf{c}(\boldsymbol{\mu}, \mathbf{d}) && \forall \boldsymbol{\mu} \in \underline{\boldsymbol{\Lambda}}. && \text{(2.67d)}
\end{aligned}
$$

In particular, we set $\Omega = [-2, 2]^2$ and $\mathcal{B} = [0, 1]^2$. The actual configuration of the solid is given by the square $\Omega_s = [-0.62, 1.38]^2$ and is defined through the map

$$
\overline{\mathbf{X}}(\mathbf{s}) = (-0.62 + 2s_1,\ -0.62 + 2s_2), \qquad \mathbf{s} = (s_1, s_2);
$$

the right hand sides $\mathbf{f}$, $\mathbf{g}$ and $\mathbf{d}$ are computed accordingly to the following choices of solution

$$
\begin{aligned}
&\mathbf{u}(x, y) = \operatorname{curl}\left((4 - x^2)^2(4 - y^2)^2\right)\\
&\mathbf{u}_{|\partial\Omega} = \mathbf{0}\\
&p(x, y) = 150\sin(x)\\
&\mathbf{X}(x, y) = \mathbf{u}(x, y)\\
&\boldsymbol{\lambda}(x, y) = \left(e^x, e^y\right).
\end{aligned}
$$

The considered domains are discretized with uniform triangulations: in particular, the fluid domain Ω is partitioned with a right-oriented mesh $\mathcal{T}_h^\Omega$, while the solid reference domain is partitioned with a left-oriented mesh $\mathcal{T}_h^\mathcal{B}$. The geometric configuration of the problem is depicted in Figure 2.6h. In terms of finite elements spaces, the choice falls again on the standard Bercovier–Pironneau element for fluid variables, while for solid variables we use continuous piecewise linear elements. Information about the considered spaces are recalled in Section 2.2.

The coupling terms are integrated both exactly, i.e. with mesh intersection, and approximately with a quadrature rule which is exact for polynomials of degree two. Since the coupling is defined between velocity functions $\mathbf{v}$ and Lagrange multiplier $\boldsymbol{\lambda}$, in the the remainder of this section, we denote by h_Ω the mesh size of the velocity sub-triangulation $\mathcal{T}_{h/2}^\Omega$. Moreover, for the $\mathbf{L}^2(\mathcal{B})$ case, the convergence of the Lagrange multiplier is studied with respect to the norm of the dual space $(\mathbf{H}^1(\mathcal{B}))'$, computed solving the associated Poisson equation.

Test 1

In this test we consider $h_\mathcal{B} \to 0$ and $h_\mathcal{B}/h_\Omega$ constant. The results are showed in Figure 2.15, where the left column is related to the $\mathbf{L}^2(\mathcal{B})$ coupling, while the right column is related to the $\mathbf{H}^1(\mathcal{B})$ coupling. As expected, since only the solid mesh size is decreasing to zero, the choice of the $\mathbf{L}^2(\mathcal{B})$ scalar product provides optimal results also when the interface matrix is assembled without mesh intersection. For the $\mathbf{H}^1(\mathcal{B})$ scalar product the behavior is the one already observed in Section 2.2, where only the method constructed with exact integration is optimal.

Test 2

For this test we choose $h_\mathcal{B} \to 0$ and also $h_\mathcal{B}/h_\Omega \to 0$, in particular $h_\mathcal{B} = h_\Omega^{3/2}$. The results are collected in Figure 2.16, with the same format of the previous test. In this case both choices of coupling terms present equivalent behaviors with respect to exact and inexact integration. For the $\mathbf{H}^1(\mathcal{B})$ case computed without mesh intersection, we can observe some oscillations of the convergence curve, but the overall behavior is consistent with the exact case and the theoretical estimates. Finally, let us remark that the suboptimal rate of convergence of the solid variables is 1/3, which is consequence of our choices of mesh sizes.

Remark 2.4.3. *A first comparison between the performance provided by the* $\mathbf{L}^2(\mathcal{B})$

and $\mathbf{H}^1(\mathcal{B})$ scalar products has been discussed by Boffi, Ruggeri and Gastaldi [30]. This preliminary study was conducted on two dimensional elliptic interface problems with discontinuous coefficients considering several choices of $h_\mathcal{B}/h_\Omega$ and with exact integration of the coupling term. In particular, the $\mathbf{L}^2(\mathcal{B})$ case showed some instabilities when $h_\mathcal{B}/h_\Omega \leq 1$, whereas the $\mathbf{H}^1(\mathcal{B})$ scalar product worked generally well. This means that the choice of coupling term cannot be done only looking at the efficiency of the method. Clearly, from a computational point of view, the choice of the $\mathbf{L}^2(\mathcal{B})$ scalar product appears convenient because it produces good convergence properties also when assembled with an approximate and cheap procedure, but, in certain situations, accurate results may be obtained only by means of the $\mathbf{H}^1(\mathcal{B})$ scalar product computed with mesh intersection.

2.5 Inf–sup conditions for inexact coupling

In the previous sections, we studied quadrature error estimates for the coupling term $\mathbf{c}$ when inexactly assembled without mesh intersection. Since the study has been performed assuming the well-posedness of the problem, we dedicate this section to the proof of the inf-sup conditions. To this aim, let us recall Problem 2.3.2 in matrix form

$$\left[\begin{array}{ccc|c} \mathsf{A}_f & \mathsf{B}^\top & 0 & \mathsf{C}_{f,h}^\top \\ \mathsf{B} & 0 & 0 & 0 \\ 0 & 0 & \mathsf{A}_s & -\mathsf{C}_s^\top \\ \hline \mathsf{C}_{f,h} & 0 & -\mathsf{C}_s & 0 \end{array}\right] \left[\begin{array}{c} \mathbf{u}_h^\star \\ p_h^\star \\ \mathbf{X}_h^\star \\ \hline \boldsymbol{\lambda}_h^\star \end{array}\right] = \left[\begin{array}{c} \mathbf{f} \\ \mathbf{0} \\ \mathbf{g} \\ \hline \mathbf{d} \end{array}\right]. \tag{2.68}$$

In this case, the bilinear form $\mathscr{B}_h$ defined in (2.21) coincides with the continuous $\mathscr{B}(\mathbf{V}_h, q_h) = (\operatorname{div} \mathbf{v}_h, q_h)_\Omega$, while $\mathscr{A}_h$ is defined as

$$\begin{aligned} \mathscr{A}_h(\mathbf{U}_h, \mathbf{V}_h) = \mathbf{a}_f(\mathbf{u}_h, \mathbf{v}_h) + \mathbf{a}_s(\mathbf{X}_h, \mathbf{Y}_h) \\ + \mathbf{c}_h(\boldsymbol{\lambda}_h, \mathbf{v}_h(\overline{\mathbf{X}})) - \mathbf{c}(\boldsymbol{\lambda}_h, \mathbf{Y}_h) - \mathbf{c}_h(\boldsymbol{\mu}_h, \mathbf{u}_h(\overline{\mathbf{X}})) + \mathbf{c}(\boldsymbol{\mu}_h, \mathbf{X}_h) \end{aligned} \tag{2.69}$$

for all $\mathbf{U}_h, \mathbf{V}_h \in \boldsymbol{\mathcal{V}}_h$, $q_h \in Q_h$.

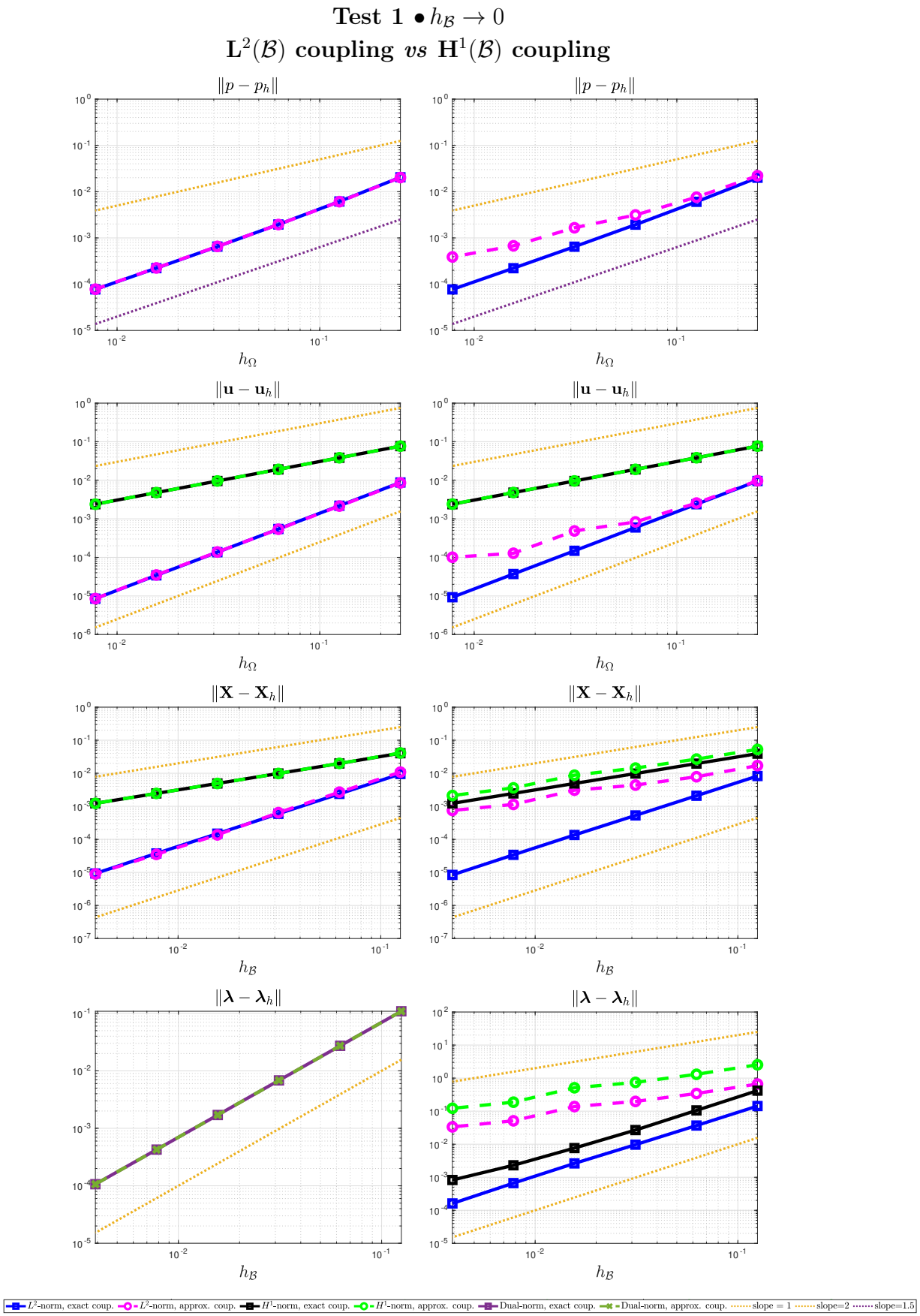


Figure 2.15: Convergence plots of Test 1. Comparison between the $\mathbf{L}^2(\mathcal{B})$ coupling (left column) and $\mathbf{H}^1(\mathcal{B})$ coupling (right column).

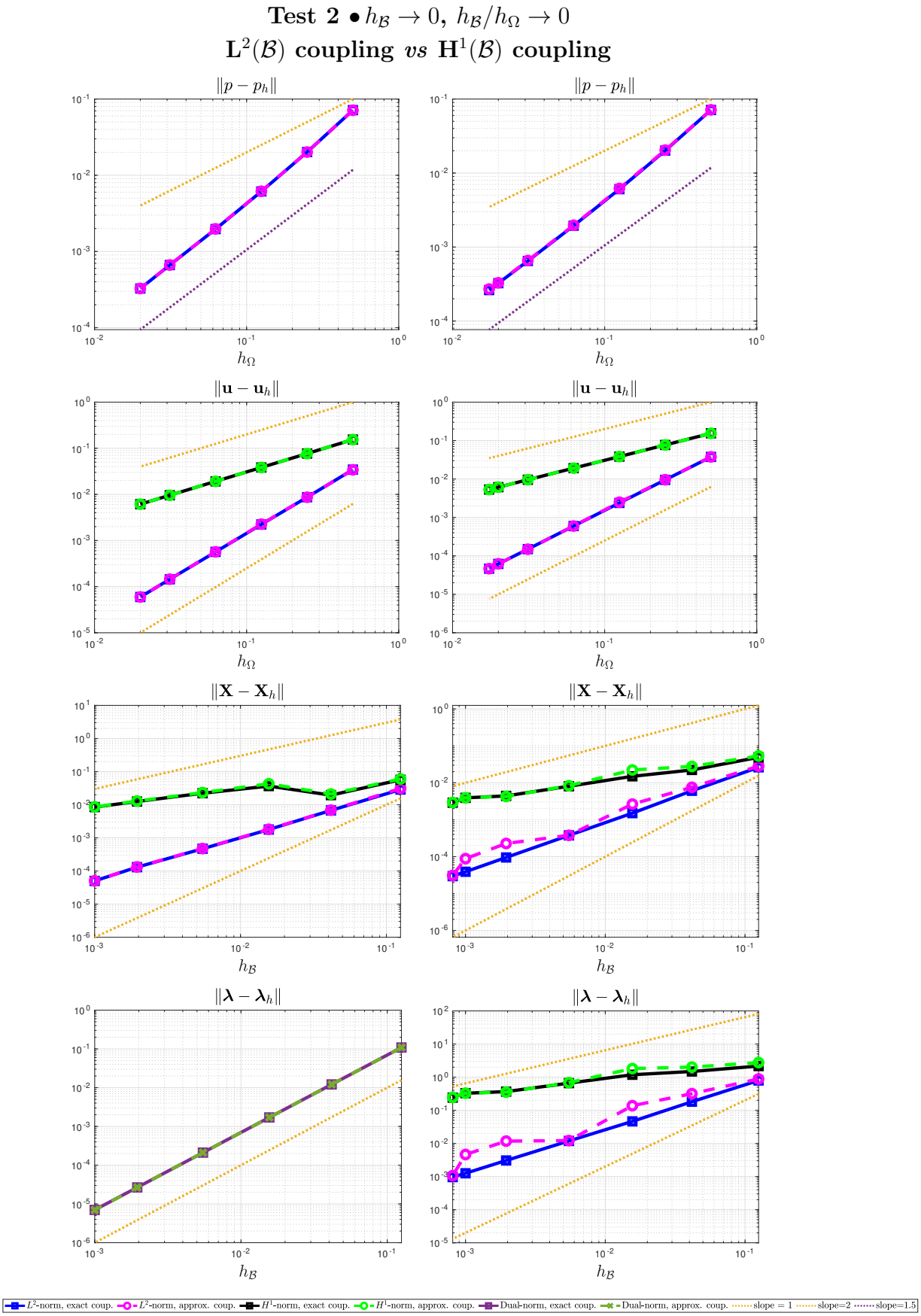


Figure 2.16: Convergence plots of Test 2. Comparison between the $\mathbf{L}^2(\mathcal{B})$ coupling (left column) and $\mathbf{H}^1(\mathcal{B})$ coupling (right column).

Our problem is actually a double saddle point problem, indeed rearranging variables, we can write

$$\left[\begin{array}{ccc|c} \mathsf{A}_f & 0 & \mathsf{C}_{f,h}^\top & \mathsf{B}^\top \\ 0 & \mathsf{A}_s & -\mathsf{C}_s^\top & 0 \\ \mathsf{C}_{f,h} & -\mathsf{C}_s & 0 & 0 \\ \hline \mathsf{B}^\top & 0 & 0 & 0 \end{array}\right] \left[\begin{array}{c} \mathbf{u}_h^\star \\ \mathbf{X}_h^\star \\ \boldsymbol{\lambda}_h^\star \\ \hline p_h^\star \end{array}\right] = \left[\begin{array}{c} \mathbf{f} \\ \mathbf{g} \\ \mathbf{d} \\ \hline \mathbf{0} \end{array}\right]; \tag{2.70}$$

and observe that

$$\left[\begin{array}{cc:c} \mathsf{A}_f & 0 & \mathsf{C}_{f,h}^\top \\ 0 & \mathsf{A}_s & -\mathsf{C}_s^\top \\ \hdashline \mathsf{C}_{f,h} & -\mathsf{C}_s & 0 \end{array}\right].$$

Consequently, exploiting this structure, the proof of well-posedness can be carried out in three steps, as already done in [24], applying results from [96].

- **Step 1.** We prove the inf-sup condition for $\mathscr{B}$.

This first step is trivial, because from the existing theory $\mathscr{B}$ satisfies the inf-sup condition recalled in Proposition 1.5.2. Then, since our aim is to prove that $\mathscr{A}_h$ is invertible in the discrete kernel of $\mathscr{B}$

$$\boldsymbol{\mathcal{K}}_h[\mathscr{B}] = \{\mathbf{V}_h \in \boldsymbol{\mathcal{V}}_h : \mathscr{B}(\mathbf{V}_h, q_h) = 0 \quad \forall q_h \in Q_h\}, \tag{2.71}$$

we introduce the space

$$\underline{\mathbf{V}}_{0,h} = \{\mathbf{v}_h \in \underline{\mathbf{V}}_h : (\operatorname{div} \mathbf{v}_h, q_h)_\Omega = 0 \quad \forall q_h \in Q_h\} \tag{2.72}$$

and we proceed as follows.

- **Step 2.** We prove the inf-sup condition for the bilinear form $\mathscr{C}_h$ associated to the operator $[\mathsf{C}_{f,h}, -\mathsf{C}_s]$ on the space $\underline{\mathbf{V}}_{0,h} \times \underline{\mathbf{S}}_h$. We focus on the particular case with $\underline{\boldsymbol{\Lambda}}_h \subset \underline{\mathbf{S}}_h$.

- **Step 3.** We prove the ellipticity of the operator $\begin{bmatrix} \mathsf{A}_f & 0 \\ 0 & \mathsf{A}_s \end{bmatrix}$ in the kernel of $\mathscr{C}_h$.

Before starting the analysis, let us introduce the L^2 projection onto $\underline{\mathbf{S}}_h$

$$\Pi^0 : \mathbf{H}^1(\mathcal{B}) \longrightarrow \underline{\mathbf{S}}_h \tag{2.73}$$

such that for any $\mathbf{Y} \in \mathbf{H}^1(\mathcal{B})$, the projection $\Pi^0\mathbf{Y}_h$ satisfies

$$\int_{\mathcal{B}} \big(\mathbf{Y} - \Pi^0\mathbf{Y}\big) \cdot \mathbf{Z}_h \, \mathrm{d}\mathbf{s} = 0 \qquad \forall \mathbf{Z}_h \in \underline{\mathbf{S}}_h. \tag{2.74}$$

From standard theory, the following property holds true

$$\big\|\Pi^0\mathbf{Y}\big\|_{0,\mathcal{B}} \leq C \, \|\mathbf{Y}\|_{0,\mathcal{B}} \qquad \forall \mathbf{Y} \in \mathbf{H}^1(\mathcal{B}). \tag{2.75}$$

We start proving the inf-sup for $\mathscr{C}_h$.

Proposition 2.5.1. *Let us assume that the $\mathbf{L}^2(\mathcal{B})$ term of $\mathbf{c}_h$ is computed with a quadrature rule $\{(\mathbf{p}_k^0, \omega_k^0)\}_{k=1}^{K_0}$ which is exact for quadratic polynomials, while the $\mathbf{L}^2(\mathcal{B})$ scalar product of gradients is approximated with a quadrature rule $\{(\mathbf{p}_k^1, \omega_k^1)\}_{k=1}^{K_1}$ exact for constants. If $\underline{\mathbf{\Lambda}}_h \subset \underline{\mathbf{S}}_h$, then there exists a constant $\zeta_{\mathbf{c}} > 0$ such that the following condition holds true*

$$\sup_{(\mathbf{v}_h, \mathbf{Y}_h) \in \underline{\mathbf{V}}_{0,h} \times \underline{\mathbf{S}}_h} \frac{\mathbf{c}_h(\boldsymbol{\mu}_h, \mathbf{v}_h(\overline{\mathbf{X}}) - \mathbf{Y}_h)}{\|\mathbf{v}_h\|_{1,\Omega} + \|\mathbf{Y}_h\|_{1,\mathcal{B}}} \geq \zeta_{\mathbf{c}} \, \|\boldsymbol{\mu}_h\|_{\underline{\mathbf{\Lambda}}} \qquad \forall \boldsymbol{\mu}_h \in \underline{\mathbf{\Lambda}}_h. \tag{2.76}$$

Proof. Case 1. We consider the $\mathbf{L}^2(\mathcal{B})$ coupling term (1.53) computed without mesh intersection as described in Section 2.1.2. For simplicity, we recall its exact and inexact definition

$$\mathbf{c}(\boldsymbol{\mu}_h, \mathbf{Y}_h) = (\boldsymbol{\mu}_h, \mathbf{Y}_h)_{\mathcal{B}} \qquad \forall \boldsymbol{\mu}_h \in \underline{\mathbf{\Lambda}}_h, \forall \mathbf{Y}_h \in \underline{\mathbf{S}}_h,$$

$$\mathbf{c}_h(\boldsymbol{\mu}_h, \mathbf{Y}_h) = \sum_{\mathrm{T}_s \in \mathcal{T}_h^{\mathcal{B}}} |\mathrm{T}_s| \sum_{k=1}^{K_0} \omega_k^0 \boldsymbol{\mu}_h(\mathbf{p}_k^0) \cdot \mathbf{Y}_h(\mathbf{p}_k^0) \qquad \forall \boldsymbol{\mu}_h \in \underline{\mathbf{\Lambda}}_h, \forall \mathbf{Y}_h \in \underline{\mathbf{S}}_h.$$

Thanks to the definition of norm in the dual space $\underline{\mathbf{\Lambda}} = (\mathbf{H}^1(\mathcal{B}))'$, there exists $\tilde{\mathbf{Y}} \in \mathbf{H}^1(\mathcal{B})$ realizing the supremum at continuous level, hence we have

$$\|\boldsymbol{\mu}_h\|_{\underline{\mathbf{\Lambda}}} = \sup_{\mathbf{Y} \in \mathbf{H}^1(\mathcal{B})} \frac{(\boldsymbol{\mu}_h, \mathbf{Y})_{\mathcal{B}}}{\|\mathbf{Y}\|_{1,\mathcal{B}}} = \sup_{\mathbf{Y} \in \mathbf{H}^1(\mathcal{B})} \frac{\mathbf{c}(\boldsymbol{\mu}_h, \mathbf{Y})}{\|\mathbf{Y}\|_{1,\mathcal{B}}} = \frac{\mathbf{c}(\boldsymbol{\mu}_h, \tilde{\mathbf{Y}})}{\|\tilde{\mathbf{Y}}\|_{1,\mathcal{B}}}. \tag{2.77}$$

Now, making use of the projection Π^0 with its property (2.75), we can write

$$\frac{\mathbf{c}(\boldsymbol{\mu}_h, \tilde{\mathbf{Y}})}{\|\tilde{\mathbf{Y}}\|_{1,\mathcal{B}}} = \frac{\mathbf{c}(\boldsymbol{\mu}_h, \Pi^0\tilde{\mathbf{Y}})}{\|\tilde{\mathbf{Y}}\|_{1,\mathcal{B}}} \leq C \frac{\mathbf{c}(\boldsymbol{\mu}_h, \Pi^0\tilde{\mathbf{Y}})}{\|\Pi^0\tilde{\mathbf{Y}}\|_{1,\mathcal{B}}} \leq \sup_{\mathbf{Y}_h \in \underline{\mathbf{S}}_h} \frac{\mathbf{c}(\boldsymbol{\mu}_h, \mathbf{Y}_h)}{\|\mathbf{Y}_h\|_{1,\mathcal{B}}}. \tag{2.78}$$

Since we are assuming that the considered quadrature rule is exact for quadratic polynomial, we have that $\mathbf{c}$ and $\mathbf{c}_h$ are equivalent when both integrand functions are defined on the solid domain $\mathcal{B}$, therefore we can write

$$\|\boldsymbol{\mu}_h\|_{\underline{\boldsymbol{\Lambda}}} \leq \sup_{\mathbf{Y}_h \in \underline{\mathbf{S}}_h} \frac{\mathbf{c}(\boldsymbol{\mu}_h, \mathbf{Y}_h)}{\|\mathbf{Y}_h\|_{1,\mathcal{B}}} = \sup_{\mathbf{Y}_h \in \underline{\mathbf{S}}_h} \frac{\mathbf{c}_h(\boldsymbol{\mu}_h, \mathbf{Y}_h)}{\|\mathbf{Y}_h\|_{1,\mathcal{B}}} \tag{2.79}$$

and, finally,

$$\sup_{\mathbf{Y}_h \in \underline{\mathbf{S}}_h} \frac{\mathbf{c}_h(\boldsymbol{\mu}_h, \mathbf{Y}_h)}{\|\mathbf{Y}_h\|_{1,\mathcal{B}}} \leq \sup_{(\mathbf{v}_h, \mathbf{Y}_h) \in \underline{\mathbf{V}}_{0,h} \times \underline{\mathbf{S}}_h} \frac{\mathbf{c}_h(\boldsymbol{\mu}_h, \mathbf{v}_h(\overline{\mathbf{X}}) - \mathbf{Y}_h)}{\|\mathbf{v}_h\|_{1,\Omega} + \|\mathbf{Y}_h\|_{1,\mathcal{B}}}. \tag{2.80}$$

Case 2. Now we consider the coupling term (1.41) defined as the scalar product of $\mathbf{H}^1(\mathcal{B})$ and discretized without mesh intersection (see Section 2.1.2). We recall both definitions:

$$\mathbf{c}(\boldsymbol{\mu}_h, \mathbf{Y}_h) = (\boldsymbol{\mu}_h, \mathbf{Y}_h)_{\mathcal{B}} + (\boldsymbol{\nabla}_{\mathrm{s}} \boldsymbol{\mu}_h, \boldsymbol{\nabla}_{\mathrm{s}} \mathbf{Y}_h)_{\mathcal{B}},$$

$$\mathbf{c}_h(\boldsymbol{\mu}_h, \mathbf{Y}_h) = \sum_{\mathrm{T}_s \in \mathcal{T}_h^{\mathcal{B}} \mathcal{T}_h^{\Omega}} |\mathrm{T}_s| \left[\sum_{k=1}^{K_0} \omega_k^0 \boldsymbol{\mu}_h(\mathbf{p}_k^0) \cdot \mathbf{Y}_h(\mathbf{p}_k^0) + \sum_{k=1}^{K_1} \omega_k^1 \, \boldsymbol{\nabla}_{\mathrm{s}} \boldsymbol{\mu}_h(\mathbf{p}_k) : \boldsymbol{\nabla}_{\mathrm{s}} \mathbf{Y}_h(\mathbf{p}_k) \right] \tag{2.81}$$

for all $\boldsymbol{\mu}_h \in \underline{\boldsymbol{\Lambda}}_h, \forall \mathbf{Y}_h \in \underline{\mathbf{S}}_h$. Given $\boldsymbol{\mu}_h \in \underline{\boldsymbol{\Lambda}}_h$, we can choose $\mathbf{Y}_h = \boldsymbol{\mu}_h$ so that

$$\begin{aligned} \|\boldsymbol{\mu}_h\|_{\underline{\boldsymbol{\Lambda}}} &= \frac{(\boldsymbol{\mu}_h, \mathbf{Y}_h)_{\mathcal{B}} + (\boldsymbol{\nabla}_{\mathrm{s}} \boldsymbol{\mu}_h, \boldsymbol{\nabla}_{\mathrm{s}} \mathbf{Y}_h)_{\mathcal{B}}}{\|\mathbf{Y}_h\|_{1,\mathcal{B}}} \\ &= \frac{\mathbf{c}(\boldsymbol{\mu}_h, \mathbf{Y}_h)}{\|\mathbf{Y}_h\|_{1,\mathcal{B}}} \leq \sup_{\mathbf{Y}_h \in \underline{\mathbf{S}}_h} \frac{\mathbf{c}(\boldsymbol{\mu}_h, \mathbf{Y}_h)}{\|\mathbf{Y}_h\|_{1,\mathcal{B}}}. \end{aligned} \tag{2.82}$$

At this point, with the same computation we did in (2.79), the result is proved.

□

In the following proposition, we prove that $\mathscr{A}_h$ is coercive on the kernel of $\mathscr{C}_h$.

Proposition 2.5.2. *Let us assume that the $\mathbf{L}^2(\mathcal{B})$ term of $\mathbf{c}_h$ is computed with a quadrature rule $\{(\mathbf{p}_k^0, \omega_k^0)\}_{k=1}^{K_0}$ which is exact for quadratic polynomials, while the $\mathbf{L}^2(\mathcal{B})$ scalar product of gradients is approximated with a quadrature rule $\{(\mathbf{p}_k^1, \omega_k^1)\}_{k=1}^{K_1}$ exact for constants. There exists $\zeta_{\mathbf{a}} > 0$ independent on the mesh size such that*

$$\mathbf{a}_f(\mathbf{u}_h, \mathbf{u}_h) + \mathbf{a}_s(\mathbf{X}_h, \mathbf{X}_h) \geq \zeta_{\mathbf{a}}(\|\mathbf{u}_h\|_{1,\Omega}^2 + \|\mathbf{X}_h\|_{1,\mathcal{B}}^2) \tag{2.83}$$

for all pairs $(\mathbf{u}_h, \mathbf{X}_h)$ in the kernel of $\mathscr{C}_h$ defined as

$$\boldsymbol{\mathcal{K}}[\mathscr{C}_h] = \{(\mathbf{v}_h, \mathbf{Y}_h) \in \underline{\mathbf{V}}_{0,h} \times \mathbf{S}_h : \mathbf{c}_h(\boldsymbol{\mu}_h, \mathbf{v}_h(\overline{\mathbf{X}})) - \mathbf{c}(\boldsymbol{\mu}_h, \mathbf{Y}_h) = 0 \quad \forall \boldsymbol{\mu}_h \in \underline{\boldsymbol{\Lambda}}_h\}.$$

Proof. Before starting the proof, we recall the definition of $\mathbf{a}_f$ and $\mathbf{a}_s$ presented in (1.54), assuming that $\mathbb{P}(\mathbb{F}) = \kappa\mathbb{F}$. In particular, we consider $\mathbf{a}_f$ in a simplified version without the nonlinear convective term:

$$\begin{aligned} \mathbf{a}_f(\mathbf{u}, \mathbf{v}) &= \alpha(\mathbf{u}, \mathbf{v})_\Omega + a(\mathbf{u}, \mathbf{v}) \\ \mathbf{a}_s(\mathbf{X}, \mathbf{Y}) &= \beta(\mathbf{X}, \mathbf{Y})_\mathcal{B} + \gamma(\boldsymbol{\nabla}_\mathrm{s}\, \mathbf{X}, \boldsymbol{\nabla}_\mathrm{s}\, \mathbf{Y})_\mathcal{B}. \end{aligned} \tag{2.84}$$

We remark that we can prove the proposition once for both options of $\mathbf{c}$. Let us observe that, if $\beta > 0$, we have

$$\begin{aligned} \mathbf{a}_f(\mathbf{u}_h, \mathbf{u}_h) + \mathbf{a}_s(\mathbf{X}_h, \mathbf{X}_h) &\geq C \left\|\mathbf{u}_h\right\|_{1,\Omega}^2 + \beta \left\|\mathbf{X}_h\right\|_{0,\mathcal{B}}^2 + \kappa \left\|\boldsymbol{\nabla}_\mathrm{s}\, \mathbf{X}_h\right\|_{0,\mathcal{B}}^2 \\ &\geq C \left\|\mathbf{u}_h\right\|_{1,\Omega}^2 + \min\{\beta, \kappa\} \left\|\mathbf{X}_h\right\|_{1,\mathcal{B}}^2 \end{aligned} \tag{2.85}$$

so that the result is proved. Conversely, if $\beta = 0$, then

$$\mathbf{a}_f(\mathbf{u}_h, \mathbf{u}_h) + \mathbf{a}_s(\mathbf{X}_h, \mathbf{X}_h) \geq C \left\|\mathbf{u}_h\right\|_{1,\Omega}^2 + \kappa \left\|\boldsymbol{\nabla}_\mathrm{s}\, \mathbf{X}_h\right\|_{0,\mathcal{B}}^2 . \tag{2.86}$$

In this case, we missed the term $\|\mathbf{X}_h\|_{0,\mathcal{B}}^2$, therefore we need to recover it. We start introducing the mean of $\mathbf{X}_h$ over $\mathcal{B}$

$$\mathring{\mathbf{X}}_h = |\mathcal{B}|^{-1} \int_\mathcal{B} \mathbf{X}_h \, \mathrm{d}\mathbf{s} \tag{2.87}$$

so that, by Poincaré–Wirtinger inequality, we have

$$\|\mathbf{X}_h - \mathring{\mathbf{X}}_h\|_{0,\mathcal{B}} \leq C \left\|\boldsymbol{\nabla}_\mathrm{s}\, \mathbf{X}_h\right\|_{0,\mathcal{B}} . \tag{2.88}$$

Applying the triangle inequality, we find

$$\|\mathbf{X}_h\|_{0,\mathcal{B}} \leq \|\mathring{\mathbf{X}}_h\|_{0,\mathcal{B}} + \|\mathbf{X}_h - \mathring{\mathbf{X}}_h\|_{0,\mathcal{B}} \leq \|\mathring{\mathbf{X}}_h\|_{0,\mathcal{B}} + C \left\|\boldsymbol{\nabla}_\mathrm{s}\, \mathbf{X}_h\right\|_{0,\mathcal{B}} . \tag{2.89}$$

Exploiting that the pair $(\mathbf{u}_h, \mathbf{X}_h)$ belongs to the kernel $\boldsymbol{\mathcal{K}}[\mathscr{C}_h]$, with an easy manipulation we write

$$\mathbf{c}(\boldsymbol{\mu}_h, \mathring{\mathbf{X}}_h) = \mathbf{c}_h(\boldsymbol{\mu}_h, \mathbf{u}_h(\overline{\mathbf{X}})) - \mathbf{c}(\boldsymbol{\mu}_h, \mathbf{X}_h - \mathring{\mathbf{X}}_h) \quad \forall \boldsymbol{\mu}_h \in \underline{\boldsymbol{\Lambda}}_h \tag{2.90}$$

so that, taking $\boldsymbol{\mu}_h = \mathring{\mathbf{X}}_h$, we have

$$\|\mathring{\mathbf{X}}_h\|_{0,\mathcal{B}}^2 = \mathbf{c}(\mathring{\mathbf{X}}_h, \mathring{\mathbf{X}}_h) = \mathbf{c}_h(\mathring{\mathbf{X}}_h, \mathbf{u}_h(\overline{\mathbf{X}})) - \mathbf{c}(\mathring{\mathbf{X}}_h, \mathbf{X}_h - \mathring{\mathbf{X}}_h). \tag{2.91}$$

Now, taking into account the definition of $\mathring{\mathbf{X}}_h$, the term $\mathbf{c}(\mathring{\mathbf{X}}_h, \mathbf{X}_h - \mathring{\mathbf{X}}_h)$ vanishes: by linearity and thanks to the fact that $\boldsymbol{\nabla}_\mathrm{s}\, \mathring{\mathbf{X}}_h = \mathbf{0}$, we can write

$$\mathbf{c}(\mathring{\mathbf{X}}_h, \mathbf{X}_h - \mathring{\mathbf{X}}_h) = \int_\mathcal{B} \mathring{\mathbf{X}}_h \cdot \mathbf{X}_h \, \mathrm{d}\mathbf{s} - \int_\mathcal{B} \mathring{\mathbf{X}}_h^2 \, \mathrm{d}\mathbf{s}; \tag{2.92}$$

factorizing $\mathring{\mathbf{X}}_h$ and exploiting the precision of the chosen quadrature rule, we find

$$\int_{\mathcal{B}} \mathring{\mathbf{X}}_h \cdot \mathbf{X}_h \, \mathrm{d}\mathbf{s} - \int_{\mathcal{B}} \mathring{\mathbf{X}}_h^2 \, \mathrm{d}\mathbf{s} = \mathring{\mathbf{X}}_h \left(\int_{\mathcal{B}} \mathbf{X}_h \, \mathrm{d}\mathbf{s} - |\mathcal{B}| \, \mathring{\mathbf{X}}_h \right) = 0. \tag{2.93}$$

At this point, we look for a bound for $\mathbf{c}_h(\mathring{\mathbf{X}}_h, \mathbf{u}_h(\overline{\mathbf{X}}))$. Adding and subtracting the same quantity, the following equality holds

$$\mathbf{c}_h(\mathring{\mathbf{X}}_h, \mathbf{u}_h(\overline{\mathbf{X}})) = \mathbf{c}_h(\mathring{\mathbf{X}}_h, \mathbf{u}_h(\overline{\mathbf{X}})) - \mathbf{c}(\mathring{\mathbf{X}}_h, \mathbf{u}_h(\overline{\mathbf{X}})) + \mathbf{c}(\mathring{\mathbf{X}}_h, \mathbf{u}_h(\overline{\mathbf{X}})) \tag{2.94}$$

and, by continuity, we get

$$\mathbf{c}(\mathring{\mathbf{X}}_h, \mathbf{u}_h(\overline{\mathbf{X}})) \leq \|\mathring{\mathbf{X}}_h\|_{0,\mathcal{B}} \|\mathbf{u}_h(\overline{\mathbf{X}})\|_{0,\mathcal{B}}. \tag{2.95}$$

Using the quadrature estimate of Proposition 2.4.1, we obtain

$$\mathbf{c}_h(\mathring{\mathbf{X}}_h, \mathbf{u}_h(\overline{\mathbf{X}})) - \mathbf{c}(\mathring{\mathbf{X}}_h, \mathbf{u}_h(\overline{\mathbf{X}})) \leq C h_{\mathcal{B}}^{3/2} |\log h_{\mathcal{B}}^{min}| \|\mathring{\mathbf{X}}_h\|_{0,\mathcal{B}} \|\mathbf{u}_h\|_{1,\Omega} \tag{2.96}$$

so that, in combination with (2.92) and (2.93), we find

$$\|\mathring{\mathbf{X}}_h\|_{0,\mathcal{B}} \leq C(1 + h_{\mathcal{B}}^{3/2} |\log h_{\mathcal{B}}^{min}|) \, \|\mathbf{u}_h\|_{1,\Omega} \, . \tag{2.97}$$

Finally, putting together (2.97) and (2.89), we get

$$\|\mathbf{X}_h\|_{0,\mathcal{B}} \leq C(1 + h_{\mathcal{B}}^{3/2} |\log h_{\mathcal{B}}^{min}|) \, \|\mathbf{u}_h\|_{1,\Omega} + C \, \|\boldsymbol{\nabla}_{\mathrm{s}} \mathbf{X}_h\|_{0,\mathcal{B}} \tag{2.98}$$

and this concludes the proof since we have a constant $\zeta_{\mathbf{a}}$ such that

$$\mathbf{a}_f(\mathbf{u}_h, \mathbf{u}_h) + \mathbf{a}_s(\mathbf{X}_h, \mathbf{X}_h) \geq \zeta_{\mathbf{a}} (\|\mathbf{u}_h\|_{1,\Omega}^2 + \|\mathbf{X}_h\|_{1,\mathcal{B}}^2). \tag{2.99}$$

□

Finally, combining Propositions 2.5.1 and 2.5.2, the well-posedness of $\mathscr{A}_h$ is immediately proved accordingly with [96].

Proposition 2.5.3. *There exists a positive constant $\theta^\star$ such that the following inf-sup condition holds true*

$$\inf_{\mathbf{U} \in \boldsymbol{\mathcal{K}}_h[\mathscr{B}]} \sup_{\mathbf{V} \in \boldsymbol{\mathcal{K}}_h[\mathscr{B}]} \frac{\mathscr{A}_h(\mathbf{U}, \mathbf{V})}{|||\mathbf{U}||| \, |||\mathbf{V}|||} \geq \theta^\star. \tag{2.100}$$

Chapter 3

A parallel solver

The numerical simulation of complex and large scale problems, such as fluid-structure interactions, requires huge computational resources. During the last decades, the advent of modern supercomputers contributed to the development of parallel software and architectures. Nevertheless, the available resources are clearly limited, therefore the design of efficient parallel solvers is fundamental to balance accuracy and computational costs, also in terms of run time.

Several studies have been already focused on parallel solvers for the solution of discrete systems arising from fluid-structure interaction problems. We mention the works by Deparis, Quarteroni and collaborators [11, 44, 47], Barker and Cai [12] and Wu and Cai [95] with applications in cardiac simulations and blood flow modeling. In [63], Klawonn and collaborators focus on two-levels overlapping Schwartz method, while Jodlbauer, Langer and Wick propose in [71] parallel block preconditioners for a monolithic solver. We finally recall the recent work by Wichrowski, Heltai and collaborators [94], based on high contrast Stokes preconditioners for incompressible FSI problems. We remark that in all these cases, the ALE formulation has been considered. On the other hand, in [76, 77], the authors introduce preconditioners for XFEM formulations and in [74], Krause and Zulian propose a parallel approach to deal with variational transfer of information between non overlapping meshes in a mortar-like setting.

This chapter is focused on the preliminary study about a parallel solver for fluid-structure interaction problems with fictitious domain approach we presented in [21, 22] for two dimensional problems. In our work, the problem is solved monolithically, which is a rather challenging task. Consequently, exploiting the block structure of our system, we introduce and analyze from the numerical point of

view two block preconditioners we combine with the GMRES solver. This kind of design is also motivated by the mixed nature of the equations, which allows the use of preconditioners taking into account specific features of each sub-problem. In particular, our Fortran implementation is based on the library PETSc from Argonne National Laboratory [9, 10] and makes use of the direct solver Mumps [3, 4] to invert the diagonal blocks of the preconditioners under consideration. A challenging feature is of course represented by the computation of the interface matrix, topic widely discussed in the previous chapter. Indeed, the motion of the immersed solid body may lead to many possible data distribution scenarios which are not easy to predict and, moreover, the track of all the element by element interaction has to be done at each discrete time, with consequent reassembly of the coupling matrix.

Our analysis regards several properties of the parallel solver and it is applied to both linear and nonlinear models for the solid material, while the fluid model is simplified removing the convective term of the Navier–Stokes equations. Moreover, we assume that fluid and solid materials share the same density and viscosity, as reasonably assumed in classical IBM setting [87].

As first step towards the design of an effective parallel solver for our formulation, we conduct a numerical investigation on two academic problems which should be helpful for understanding all the computational challenges posed by the approach. We analyze the optimality of the proposed algorithms, their strong and weak scalability and robustness with respect to different choices of time step. Moreover, in order to check the admissibility of the results, we study the variation in volume of the solid body, which should be theoretically zero thanks to the incompressibility of both fluid and solid materials.

The chapter is divided into three sections: in the first section, after presenting the problem we are going to solve, we discuss the main features of our numerical method such as finite element spaces, assembly procedure for the interface matrix and preconditioners. In particular, the time discretization is realized by means of the semi-implicit modified Backward Euler scheme. The Stokes equation is discretized with $\mathcal{Q}_2 - \mathcal{P}_1$ elements, while the solid variables are approximated by $\mathcal{Q}_1$ elements. In Section 3.2, we present a wide range of numerical results considering first a linear and then a nonlinear problem; all the tests have been performed on Linux clusters. Finally, in the last section, a discussion about the limits of the proposed solver is carried out by considering also possible future research directions.

Of course, the design of highly scalable algorithms with respect to the total run time is not trivial and several factors have to be taken into account.

3.1 The numerical method

In this section we consider a simplified version of the full discrete problem we analyzed in Section 1.4. Indeed, let us consider Problem 1.4.1: we neglect the convective term from the fluid equation and we assume that fluid and solid densities have the same value, so that $\delta\rho = 0$.

This choice about densities may appear as a limitation of our method, but it is not; indeed, this is the particular situation in which instabilities may occur due to the added mass effect. For instance, this phenomenon has been discussed by Causin, Gerbeau and Nobile in [36] for FSI problems modeled by ALE: non implicit time advancing schemes are unconditionally unstable when fluid and solid have same densities and regardless the other discrete parameters. It has been also shown that such critical behavior can be alleviated with some appropriate treatments for transmission conditions [46, 8].

Moreover, as described in Remark 1.4.1, we set the coupling term to be the scalar product in $\mathbf{L}^2(\mathcal{B})$. Therefore, we are going to study the following problem in two dimensions.

Problem 3.1.1. *Given* $\mathbf{u}_h^0 \in \underline{\mathbf{V}}_h$ *and* $\mathbf{X}_h^0 \in \mathbf{W}^{1,\infty}(\mathcal{B})$*, for* $n = 1, \dots, N$ *find* $\mathbf{u}_h^n \in \underline{\mathbf{V}}_h$*,* $p_h^n \in Q_h$*,* $\mathbf{X}_h^n \in \underline{\mathbf{S}}_h$*, and* $\boldsymbol{\lambda}_h^n \in \underline{\boldsymbol{\Lambda}}_h$*, such that*

$$\rho_f\left(\frac{\mathbf{u}_h^{n+1}-\mathbf{u}_h^n}{\Delta t},\mathbf{v}\right)_\Omega + a(\mathbf{u}_h^{n+1},\mathbf{v}) - (\operatorname{div}\mathbf{v}, p_h^{n+1})_\Omega + \left(\boldsymbol{\lambda}_h^{n+1}, \mathbf{v}(\mathbf{X}_h^n)\right)_\mathcal{B} = 0 \qquad \forall \mathbf{v} \in \underline{\mathbf{V}}_h \tag{3.1a}$$

$$(\operatorname{div}\mathbf{u}_h^{n+1}, q)_\Omega = 0 \qquad \forall q \in Q_h \tag{3.1b}$$

$$(\mathbb{P}(\mathbb{F}_h^{n+1}), \boldsymbol{\nabla}_{\mathrm{s}}\,\mathbf{Y})_\mathcal{B} - \left(\boldsymbol{\lambda}_h^{n+1}, \mathbf{Y}\right)_\mathcal{B} = 0 \qquad \forall \mathbf{Y} \in \underline{\mathbf{S}}_h \tag{3.1c}$$

$$\left(\boldsymbol{\mu}, \mathbf{u}_h^{n+1}(\mathbf{X}_h^n) - \frac{\mathbf{X}_h^{n+1}-\mathbf{X}_h^n}{\Delta t}\right)_\mathcal{B} = 0 \qquad \forall \boldsymbol{\mu} \in \underline{\boldsymbol{\Lambda}}_h \tag{3.1d}$$

We define the four discrete spaces on quadrilateral meshes. We set $\underline{\mathbf{V}}_h$ and Q_h to be the popular $\mathcal{Q}_2 - \mathcal{P}_1$ pair, characterized by discontinuous pressure

$$\begin{aligned} \underline{\mathbf{V}}_h &= \{\mathbf{v} \in \mathbf{H}_0^1(\Omega) : \mathbf{v}|_E \in (\mathcal{Q}_2(E))^2 \quad \forall E \in \mathcal{T}_h^\Omega\} \\ Q_h &= \{q \in \mathrm{L}_0^2(\Omega) : q|_E \in \mathcal{P}_1(E) \quad \forall E \in \mathcal{T}_h^\Omega\}. \end{aligned} \tag{3.2}$$

This Stokes element has been studied in [59, 92]; historically, the use of a $\mathcal{P}_1$ pressure on quadrilateral is not a standard choice, but it is the solution for the instability of the $\mathcal{Q}_2 - \mathcal{Q}_1$ pair.

On the other hand, for the solid variables $\mathbf{X}_h$ and $\boldsymbol{\lambda}_h$, we set $\underline{\mathbf{S}}_h = \underline{\boldsymbol{\Lambda}}_h$ to be the space of continuous bilinear functions $\mathcal{Q}_1$, i.e.

$$\underline{\mathbf{S}}_h = \underline{\boldsymbol{\Lambda}}_h = \{\mathbf{Y} \in \mathbf{H}^1(\mathcal{B}) : \mathbf{Y}|_E \in (\mathcal{Q}_1(E))^2 \quad \forall E \in \mathcal{T}_h^{\mathcal{B}}\}. \tag{3.3}$$

We recall that if we consider a linear constitutive law for the solid material, which means $\mathbb{P}(\mathbb{F}) = \kappa\mathbb{F}$, then Problem 3.1.1 can be easily written in matrix form

$$\left[\begin{array}{cc|cc} \frac{\rho_f}{\Delta t}\mathsf{M}_f + \mathsf{K} & \mathsf{B}^\top & 0 & \mathsf{C}_f(\mathbf{X}_h^n)^\top \\ \mathsf{B} & 0 & 0 & 0 \\ \hline 0 & 0 & \mathsf{A}_s & -\mathsf{C}_s^\top \\ \mathsf{C}_f(\mathbf{X}_h^n) & 0 & -\frac{1}{\Delta t}\mathsf{C}_s & 0 \end{array}\right] \left[\begin{array}{c} \mathbf{u}_h^{n+1} \\ p_h^{n+1} \\ \hline \mathbf{X}_h^{n+1} \\ \boldsymbol{\lambda}_h^{n+1} \end{array}\right] = \left[\begin{array}{c} \mathbf{f}^n \\ 0 \\ \hline \mathbf{g}^n \\ \mathbf{d}^n \end{array}\right] = \left[\begin{array}{c} \frac{\rho_f}{\Delta t}\mathsf{M}_f\mathbf{u}_h^n \\ 0 \\ \hline 0 \\ -\frac{1}{\Delta t}\mathsf{C}_s\mathbf{X}_h^n \end{array}\right] \tag{3.4}$$

In the remainder of the chapter, we use the notation $\mathsf{A}_f = \frac{\rho_f}{\Delta t}\mathsf{M}_f + \mathsf{K}$. We remark that in more general situations, $\mathbb{P}(\mathbb{F})$ appears to be nonlinear since there is a dependence of the matrix A_s on the variable $\mathbf{X}_h^n$. If this is the case, the system has to be solved making use of a nonlinear solver such as the Newton method or a fixed point iterator. Since fixed point iterators may show poor convergence properties, in our numerical simulation, we consider the Newton method as described, for instance, in [56].

Therefore, in presence of nonlinear constitutive laws, at each time step, we compute a sequence of approximations of the solution until convergence to prescribed tolerance. We denote by $\mathbf{U}_{h,k}^n$ the global solution vector at time t_n related to the k$^{\text{th}}$ Newton iteration, i.e.

$$\mathbf{U}_{h,k}^n = \left[\begin{array}{c} \mathbf{u}_{h,k}^n \\ p_{h,k}^n \\ \mathbf{X}_{h,k}^n \\ \boldsymbol{\lambda}_{h,k}^n \end{array}\right]$$

In order to start the process, a good choice of initial approximation $\mathbf{U}_{h,0}^n$ is given by zero velocity, pressure and Lagrange multiplier, while $\mathbf{X}_{h,0}^n$ is set either to the

position of the solid computed at the previous time step or the initial configuration if we need to solve the first time step.

The computation of $\mathbf{U}^n_{h,k+1}$ is performed as follows. Given $\mathbf{U}^n_{h,k}$, we first define the residual

$$\mathbf{R}^n_k = \begin{bmatrix} \mathsf{A}_f \mathbf{u}^n_{h,k} + \mathsf{B}^\top p^n_{h,k} + \mathsf{C}_f(\mathbf{X}^n_h)^\top \boldsymbol{\lambda}^n_{h,k} \\ \mathsf{B}\mathbf{u}^n_{h,k} \\ \mathsf{JA}_s(\mathbf{X}^n_{h,k}) - \mathsf{C}_s^\top \boldsymbol{\lambda}^n_{h,k} \\ \mathsf{C}_f(\mathbf{X}^n_h)\mathbf{u}^n_{h,k} - \frac{1}{\Delta t}\mathsf{C}_s \boldsymbol{\lambda}^n_{h,k} \end{bmatrix} - \begin{bmatrix} \frac{\rho_f}{\Delta t}\mathsf{M}_f \mathbf{u}^n_h \\ 0 \\ 0 \\ -\frac{1}{\Delta t}\mathsf{C}_s \mathbf{X}^n_h \end{bmatrix} \tag{3.5}$$

and then compute the Newton correction vector $\delta\mathbf{U}^n_k$ by solving the linear system

$$\mathcal{J}\,\delta\mathbf{U}^n_k = \mathbf{R}^n_k, \tag{3.6}$$

where $\mathcal{J}$ is the global Jacobian matrix defined as

$$\mathcal{J} = \left[\begin{array}{cc|cc} \mathsf{A}_f & \mathsf{B}^\top & 0 & \mathsf{C}_f(\mathbf{X}^n_h)^\top \\ \mathsf{B} & 0 & 0 & 0 \\ \hline 0 & 0 & \mathsf{JA}_s(\mathbf{X}^n_{h,k}) & -\mathsf{C}_s^\top \\ \mathsf{C}_f(\mathbf{X}^n_h) & 0 & -\frac{1}{\Delta t}\mathsf{C}_s & 0 \end{array}\right] \tag{3.7}$$

In particular, $\mathsf{JA}_s(\mathbf{X}^n_{h,k})$ is the Jacobian related to the nonlinearity of $(\mathbb{P}(\mathbb{F}^{n+1}_h), \boldsymbol{\nabla}_{\mathrm{s}}\,\mathbf{Y})_{\mathcal{B}}$. Then, the solution is updated computing

$$\mathbf{U}^n_{h,k+1} = \mathbf{U}^n_{h,k} - \delta\mathbf{U}^n_k. \tag{3.8}$$

In our implementation, the procedure stops when a 10^{-6} reduction of the relative residual in Euclidean norm is reached, i.e.

$$\frac{|\mathbf{R}^n_k|}{|\mathbf{R}^n_0|} \leq 10^{-6} \tag{3.9}$$

where $\mathbf{R}^n_0$ is the residual at the first Newton iteration.

3.1.1 Parallel preconditioners

Before presenting two preconditioners for our parallel solver, we notice that the matrix in (3.4), and similarly the Jacobian matrix (3.7), can be split into four

blocks as follows

$$\mathbb{A}_{11} = \begin{bmatrix} \mathsf{A}_f & \mathsf{B}^\top \\ \mathsf{B} & 0 \end{bmatrix} \qquad \mathbb{A}_{12} = \begin{bmatrix} 0 & \mathsf{C}_f(\mathbf{X}_h^n)^\top \\ 0 & 0 \end{bmatrix}$$
$$\mathbb{A}_{21} = \begin{bmatrix} 0 & 0 \\ \mathsf{C}_f(\mathbf{X}_h^n) & 0 \end{bmatrix} \qquad \mathbb{A}_{22} = \begin{bmatrix} \mathsf{A}_s & -\mathsf{C}_s^\top \\ -\frac{1}{\Delta t}\mathsf{C}_s & 0 \end{bmatrix}$$

where $\mathbb{A}_{11}$ represents the fluid equations, $\mathbb{A}_{22}$ the solid contribution and the remaining two blocks $\mathbb{A}_{12} = \mathbb{A}_{21}^\top$ contain the interface matrix.

In order to solve the linear system (3.4) and the Jacobian system (3.6), we choose the parallel GMRES solver provided by the PETSc library. In particular, the following stopping criterion is adopted: we check the reduction of the Euclidean norm of the relative residual with a tolerance of 10^{-8} and restart parameter of 200. Moreover, we endow the solver with the following choices of preconditioners, so that the solution may be accelerated: we call **block–diag** the preconditioner defined as

$$\begin{bmatrix} \mathbb{A}_{11} & 0 \\ 0 & \mathbb{A}_{22} \end{bmatrix} \tag{3.10}$$

while, we call **block–tri** the triangular preconditioner

$$\begin{bmatrix} \mathbb{A}_{11} & 0 \\ \mathbb{A}_{21} & \mathbb{A}_{22} \end{bmatrix}. \tag{3.11}$$

The action of these two preconditioners is carried out by the exact inversion of the diagonal blocks $\mathbb{A}_{11}$ and $\mathbb{A}_{22}$. In particular, this operation is performed by means of the multifrontal direct solver Mumps [3, 4].

3.1.2 The interface matrix

As widely discussed in Chapter 2, the assembly of the coupling term is not trivial and it can be carried out in two possible ways since one can perform exact integration by computing the intersection between fluid and solid mesh or, alternatively, one can work with approximated integrals. In particular, we discussed in Section 2.4 that the $\mathbf{L}^2(\mathcal{B})$ coupling term provides good results with both the exact and the approximate assembly techniques, therefore we design and test our parallel solver taking into account both possibilities.

For the case of inexact integration, integrals are approximated using a first order quadrature rule for quadrilaterals: the nodes are the vertices $\mathbf{q}_1, \ldots, \mathbf{q}_4$ of the element $E \in \mathcal{T}_h^{\mathcal{B}}$ under consideration, while the weights are set equal to $1/4$. Consequently, the local coupling can be easily written as

$$\int_E \boldsymbol{\mu}_h \cdot \mathbf{v}_h(\mathbf{X}_h^n)\, d\mathbf{s} \approx \frac{|E|}{4} \sum_{k=1}^{4} \boldsymbol{\mu}_h(\mathbf{q}_k) \cdot \mathbf{v}_h(\mathbf{X}_h^n(\mathbf{q}_k)) \quad \forall E \in \mathcal{T}_h^{\mathcal{B}}. \tag{3.12}$$

On the other hand, for implementing exact integration, we partition each $E \in \mathcal{T}_h^{\mathcal{B}}$ into a certain number of polygons according to its position with respect to the fluid mesh $\mathcal{T}_h^{\Omega}$; if a polygon is not a triangle, it is triangulated for computational purpose, by connecting each vertex with the barycenter. Hence, if $E = \bigcup_{j=1}^{J} P_j$, we have that

$$\begin{aligned}
\int_E \boldsymbol{\mu}_h \cdot \mathbf{v}_h(\mathbf{X}_h^n)\, d\mathbf{s} &= \sum_{j=1}^{J} \int_{P_j} \boldsymbol{\mu}_h \cdot \mathbf{v}_h(\mathbf{X}_h^n)\, \mathrm{d}\mathbf{s} \\
&= \sum_{j=1}^{J} \sum_{i=1}^{N_j} \left[\int_{\mathrm{T}_i} \boldsymbol{\mu}_h \cdot \mathbf{v}_h(\mathbf{X}_h^n)\, \mathrm{d}\mathbf{s} \right] \\
&= \sum_{j=1}^{J} \sum_{i=1}^{N_j} |\mathrm{T}_i| \left[\sum_{k=1}^{4} \omega_k \boldsymbol{\mu}_h(\mathbf{p}_k) \cdot \mathbf{v}_h(\mathbf{X}_h^n(\mathbf{p}_k)) \right],
\end{aligned} \tag{3.13}$$

where $\{(\mathbf{p}_k, \omega_k)\}_{k=1}^{4}$ are nodes and weights of the third order Gaussian quadrature rule on triangles we introduced in Definition 2.2.3 (see Sec. 2.2.5).

An example of the two described approaches is depicted in Figure 3.1.

We remark that element-by-element intersections are computed by means of the Sutherland–Hodgman algorithm.

It is important to notice that the choice of assembly technique affects the sparsity pattern of the coupling matrix: when the composite quadrature rule is considered, a larger number of fluid degrees of freedom is involved so that the matrix appears more dense than in the case of approximate integration. An example is reported in Figure 3.2. We are going to see that this feature affects the solver, which generally requires a slightly large number of iterations to solve the linear system when $C_f(\mathbf{X}_h^n)$ is assembled exactly.

Finally, we point out that these operations are challenging and an efficient parallel implementation is not easy to be carried out. In particular, in our implementation, the coupling term is assembled using two nested loops: the first loop

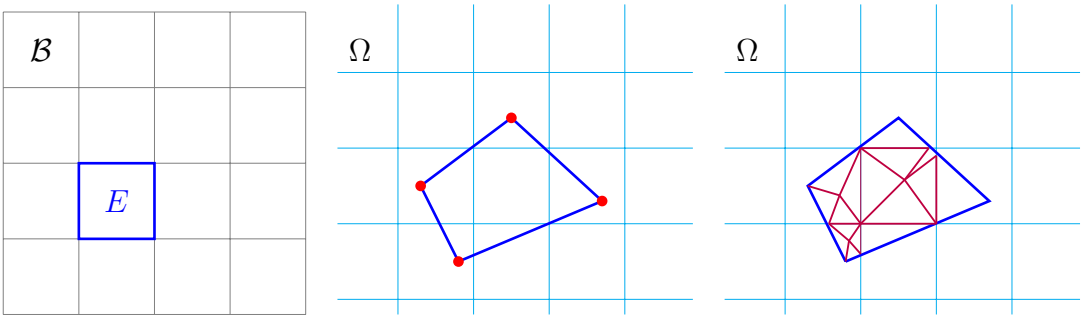


Figure 3.1: A schematic representation of the geometric aspects of the coupling operations. From the left hand side: a portion of the solid mesh $\mathcal{T}_h^{\mathcal{B}}$ with a particular element under consideration, the immersed counterpart of E associated with the quadrature rule of the vertices, the same immersed element partitioned by computing the intersection with the background fluid mesh.

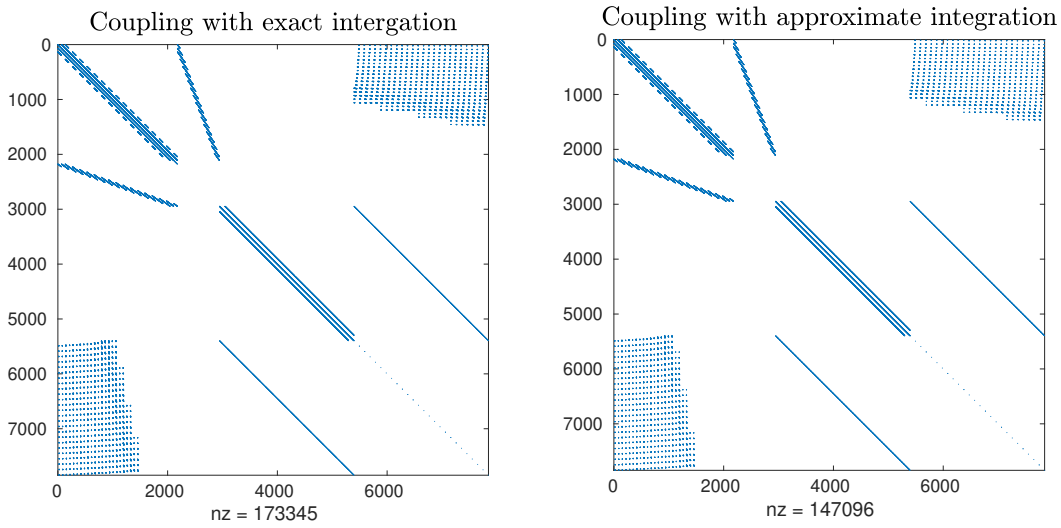


Figure 3.2: Sparsity patterns for the global matrix (3.4) when the interface term $C_f(\mathbf{X}_h^n)$ is computed with the two assembly techniques. The matrix is related to the linear elastic model described in Section 3.2.1 and discretized with 7,846 global DOFs. The number of nonzero entries is denoted by nz. When the interface matrix is assembled exactly, the total amount of nz is 173,345; on the other hand, the approximate assembly produces in total 147,095 nz.

is distributed over the processors and iterates through the solid elements, whereas the nested one is performed in serial over all the fluid elements. In particular, we are going to see that this kind of design produces a lack of weak scalability.

3.2 Numerical results

In this section we present several numerical tests with the aim of analyzing the performance of the proposed parallel solver. The study is done on two test cases: for the first one, the solid is modeled by a linear constitutive law, while for the second one a nonlinear model is chosen. Moreover, we remark that for each test, we consider and compare the two assembly techniques for the coupling term. In particular, our discussion is primarily focused on the three following features.

- **Mesh refinement.** We study robustness of the method when both fluid and solid meshes are refined, whereas the number of cores on which we distribute the computational load is kept constant. This kind of test is also called *optimality test*.

- **Weak scalability.** In this case, the size of the problem, also known as workload, assigned to each core is kept constant. To this aim, each simulation is performed by maintaining approximately constant the ratio $\frac{\#DOFs}{\#cores}$. A weakly scalable solver should maintain constant its run time.

- **Strong scalability.** We fix the number of degrees of freedom and then we solve the problem increasing the number of processors. A measure of scalability is given by the so-called *speed up factor* S^p, which is computed dividing the total run time of the reference test by the total run time of the m processors test.

Since in fluid-structure interactions several physical parameters are involved, besides parallel performance and scalability, also other kinds of test may be carried out: we focus on robustness with respect to time step refinement and conservation of mass.

- **Time step refinement.** The choice of time step affects the performance of the solver. For example, the choice of a small step produces small variations in the configuration of the problem, therefore some operations are faster

thanks to the so-called *dynamic allocation*. Moreover, the choice of time step may affect the convergence of the adopted nonlinear iterator.

- **Volume loss.** Since we are considering problems in which fluids and solids are incompressible, a good parallel solver should present slight variations of the volume of the involved entities. In particular, due to our choice of finite element spaces, the divergence free constraint is not exactly imposed.

The results we present in this section are obtained by running the parallel solver on two Linux clusters: Shaheen, provided by the Extreme Computing Research Center at King Abdullah University of Science and Technology (Saudi Arabia), and EOS, provided by the Department of Mathematics of the University of Pavia (Italy). Shaheen is a Cray XC40 cluster constituted by 6,174 dual sockets compute nodes, based on 16 core Intel Haswell processors running at 2.3GHz. Each node has 128GB of DDR4 memory running at 2300MHz. EOS is a Linux Infiniband cluster with 21 nodes, each carrying two 16 cores Intel Xeon Gold 6130 processors running at 2.1 GHz.

3.2.1 Linear solid model

The first problem we address in our numerical tests involves an incompressible solid body with linear constitutive law. This is a particular case of neo–Hookean material: the first Piola–Kirchhoff stress tensor expressed with Lagrangian description is formulated as

$$\mathbb{P}(\mathbb{F}) = \kappa\,\mathbb{F}. \tag{3.14}$$

As a consequence, the associated energy density reads as

$$W(\mathbb{F}) = \frac{\kappa}{2}\,\mathbb{F} : \mathbb{F}, \tag{3.15}$$

while the potential energy is given by

$$\mathscr{E}(\mathbf{X}) = \frac{\kappa}{2}\int_{\mathcal{B}} W(\mathbb{F})\,\mathrm{d}\mathbf{s} = \frac{\kappa}{2}\int_{\mathcal{B}} |\,\boldsymbol{\nabla}_{\mathrm{s}}\,\mathbf{X}|^2\,\mathrm{d}\mathbf{s}. \tag{3.16}$$

In particular, for our system we consider a viscous incompressible fluid filling the square $[-1,1]^2$. Then, we immerse the elastic annulus described, at rest, by

$$\{\mathbf{x} \in \mathbb{R}^2 : 0.3 \leq |\mathbf{x}| \leq 0.5\}.$$

At the initial configuration, we consider the fluid at rest and a stretched version of the annulus. When the solid body is stretched, its internal forces act with the aim of bringing it back to the resting configuration and the surrounding fluid starts its motion.

Since the geometry of the problem is symmetric, we can reduce the simulation to a quarter of the mentioned domain so that we set

$$\Omega = [0,1]^2 \qquad \text{and} \qquad \mathcal{B} = \{\mathbf{s} = (s_1, s_2) \in \mathbb{R}^2 : s_1, s_2 \geq 0,\ 0.3 \leq |\mathbf{s}| \leq 0.5\}. \tag{3.17}$$

In this reduced system, the annulus touches the left and the lower side of Ω and both fluid and solid are allowed to move along the tangential direction. On the other hand, on the upper and right sides of Ω, we impose no-slip boundary conditions for the velocity $\mathbf{u}$.

In order to complete our equations, we consider the following initial conditions at time instant $t = 0$

$$\mathbf{u}(\mathbf{x}, 0) = 0$$
$$\mathbf{X}(\mathbf{s}, 0) = \left(\frac{s_1}{1.4}, 1.4\, s_2\right).$$

Moreover, we set fluid and solid densities equal to 1 and we also assume that the two materials have same viscosity by setting $\nu_f = \nu_s = 0.1$. The initial shear modulus κ is equal to 10.

In particular, the validation of the solver is done simulating the system until $T = 2$. We collect in Figure 3.3 some snapshots of the evolution with final time set to $T = 20$.

Mesh refinement

In order to study robustness of the parallel solver with respect to mesh refinement, we solve the problem setting $\Delta t = 0.01$ and we distribute the computational load over 128 processors. Then, we solve the problem refining both fluid and solid meshes six times: the smallest problem is solved with a total amount of 30,534 DOFs, while the largest has 1,460,934 total DOFs.

Results are collected in Table 3.1 for coupling with mesh intersection and in Table 3.2 for coupling without mesh intersection; except for the assembly time T_{asm} of mass and stiffness matrices and the total time T_{tot}, all the reported run times are averaged over the 200 time instants of simulation.

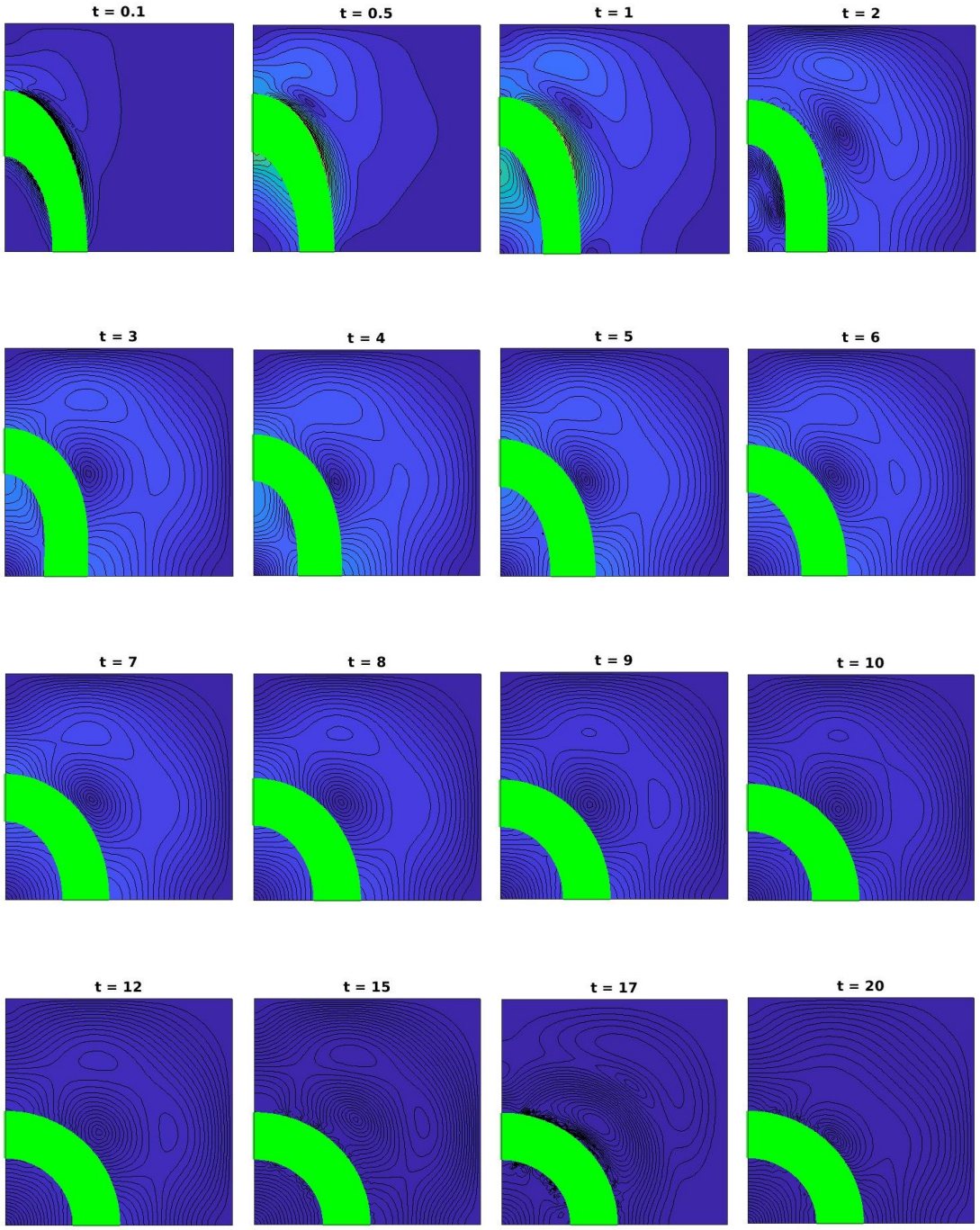


Figure 3.3: Some snapshots of the evolution in time of the elastic annulus (linear solid model).

First, we observe that, for both coupling techniques, the volume loss reduces when meshes are refined. In particular, when the exact coupling is carried out, the values remain below 0.3% if we look at the three finest cases. On the other hand, for the approximate coupling, the percentage of volume loss is always below 0.1%, showing a better mass conservation for the immersed body.

Looking at the assembly time T_{asm}, we can see that it increases moderately when we increase the number of degrees of freedom, whereas T_{coup} grows superlinearly.

Concerning the two preconditioners, from Table 3.1, we observe that block–diag is not optimal since the GMRES iteration count grows rapidly from 124 to 394. As a consequence, T_{sol} and T_{tot} grow accordingly. Conversely, block–tri is optimal since the number of iterations slightly increases, but remaining bounded by 16. Therefore, T_{sol} shows a moderate increase. A comparison between the two preconditioners can be done looking at T_{sol} in correspondence of the 746,566 DOFs case: for block–diag it is almost 27 times larger than for block–tri.

Moving to Table 3.2, the two preconditioners have comparable behaviors. Indeed, also block–diag is optimal: the number of GMRES iterations is bounded by 10. For block–tri, the good behavior is confirmed, with its bounded by 6. Therefore T_{sol} slightly increases for both choices of preconditioners, which are basically equivalent in terms of total run time.

Weak scalability

For studying the weak scalability, we fix only the temporal parameters: $T = 2$ and $\Delta t = 0.01$; then, we vary number of processors and number of DOFs by almost doubling them at each test. Theoretically, a weakly scalable parallel solver should maintain constant its total run time. In particular, for the smallest simulation, we consider 4 processors and 68,070 total degrees of freedom, while, for the largest one, we consider 128 processors and 2,152,614 total DOFs. We remark that each mesh refinement involves both $\mathcal{T}_h^{\Omega}$ and $\mathcal{T}_h^{\mathcal{B}}$.

We collect our results in Table 3.3, for the case with mesh intersection, and in Table 3.4 for the case without mesh intersection.

Of course, the assembly time T_{asm} remains basically constant because the procedure is carried out making use only of PETSc routines, which are scalable by construction. On the other hand, the time T_{coup} we need to assembly the interface matrix is not scalable due to the design of the algorithm: as mentioned in Sec-

Linear solid model – Mesh refinement test *Coupling with mesh intersection* procs = 128, T = 2, Δt = 0.01									
DOFs	vol. loss (%)	$T_{asm}(s)$	$T_{coup}(s)$	block-diag			block-tri		
				its	$T_{sol}(s)$	$T_{tot}(s)$	its	$T_{sol}(s)$	$T_{tot}(s)$
30,534	1.17	2.56e-3	2.09e-1	12	1.38	3.18e+2	7	8.08e-1	2.03e+2
120,454	5.06e-1	1.02e-2	8.94e-1	31	4.69	1.12e+3	9	1.34	4.43e+2
269,766	3.18e-1	2.35e-2	3.10	86	16.12	3.85e+3	11	2.06	1.04e+3
478,470	2.33e-1	4.01e-2	8.71	160	37.53	9.25e+3	12	3.02	2.30e+3
746,566	1.85e-1	6.50e-2	19.52	394	1.20e+2	2.79e+4	13	4.33	4.72e+3
1,074,054	1.54e-1	9.44e-2	37.72	-	-	-	14	5.51	8.65e+3
1,460,934	1.27e-1	1.27e-1	67.10	-	-	-	16	7.18	1.49e+4

Table 3.1: Refining the mesh in the linear solid model, coupling with mesh intersection. The simulations are run on the Shaheen cluster. procs = number of processors; DOFs = degrees of freedom; vol. loss = loss of structure volume in percentage; T_{asm} = CPU time to assemble the stiffness and mass matrices; T_{coup} = CPU time to assemble the coupling term; its = GMRES iterations; T_{sol} = CPU time to solve the linear system; T_{tot} = total simulation CPU time. The quantities T_{coup}, its and T_{sol} are averaged over the time steps. All CPU times are reported in seconds.

Linear solid model – Mesh refinement test *Coupling without mesh intersection* procs = 128, T = 2, Δt = 0.01									
DOFs	vol. loss (%)	$T_{asm}(s)$	$T_{coup}(s)$	block-diag			block-tri		
				its	$T_{sol}(s)$	$T_{tot}(s)$	its	$T_{sol}(s)$	$T_{tot}(s)$
30,534	6.99e-2	2.50e-3	1.68e-1	9	1.04	2.42e+2	5	6.57e-1	1.65e+2
120,454	6.89e-2	9.06e-3	2.50e-1	9	1.53	3.57e+2	6	1.01	2.53e+2
269,766	4.87e-2	2.33e-2	9.96e-1	10	2.10	6.19e+2	6	1.31	4.65e+2
478,470	4.24e-2	4.13e-2	3.70	10	2.65	1.27e+3	6	1.63	1.06e+3
746,566	4.09e-2	6.50e-2	9.90	10	3.30	2.64e+3	6	1.97	2.25e+3
1,074,054	3.69e-2	9.46e-2	20.68	10	3.93	4.92e+3	6	2.55	4.66e+3
1,460,934	3.52e-2	1.29e-1	45.39	10	4.66	1.00e+4	6	3.15	9.86e+3

Table 3.2: Refining the mesh in the linear solid model, coupling without mesh intersection. The simulations are run on the Shaheen cluster. Same format as Table 3.1.

Linear solid model – Weak scalability test *Coupling with mesh intersection* T = 2, Δt = 0.01									
procs	DOFs	$T_{asm}(s)$	$T_{coup}(s)$	block-diag			block-tri		
				its	$T_{sol}(s)$	$T_{tot}(s)$	its	$T_{sol}(s)$	$T_{tot}(s)$
4	68,070	8.55e-2	3.95	22	6.25e-1	933.43	8	2.24e-1	833.44
8	135,870	1.00e-1	5.23	38	2.16	1.48e+3	9	4.41e-1	1.13e+3
16	269,766	1.01e-1	8.77	111	10.23	3.80e+3	11	9.70e-1	1.95e+3
32	539,926	9.24e-2	59.27	706	108.05	2.50e+4	18	2.91	1.24e+4
64	1,074,054	1.90e-1	48.00	429	113.59	3.24e+4	14	3.90	1.04e+4
128	2,152,614	1.90e-1	98.63	-	-	-	18	11.43	2.20e+4

Table 3.3: Weak scalability for the linear solid model, coupling with mesh intersection. The simulations are run on the Shaheen cluster. Same format as Table 3.1.

tion 3.1.2, this procedure consists of two nested loops, on solid and fluid elements respectively, but only the outer one is distributed over the processors. Therefore, when we refine the meshes, we increase the workload without applying a proper distribution. Anyway, if we compare T_{coup} for the two assembly techniques, we see that for the inexact case values are doubled with respect to the exact one.

Looking at the performance information of the two preconditioners, we notice different behaviors. From Table 3.3, it is clear that block–diag combined with composite integration for the coupling term is not convenient: the average of GMRES iterations increases from 22 to 429 affecting T_{sol} and T_{tot}. For block–tri we have better results: the number of iterations its is bounded by 18, but the time T_{sol} spent to solve the linear system is not scalable.

From Table 3.4, we see that both block–diag and block–tri need few iterations to solve the system: in the first case its is equal to 9 or 10, while for the second preconditioner it is always equal to 6. This fact reflects on T_{sol} which assumes lower values than in Table 3.3. Despite this, the two preconditioners have a lack of weak scalability.

Strong scalability

The simulations for the strong scalability test are done with fixed spaced discretization; in particular, we discretize the system with a total amount of 478,470 DOFs.

Linear solid model – Weak scalability test *Coupling without mesh intersection* T = 2, $\Delta t = 0.01$									
procs	DOFs	$T_{asm}(s)$	$T_{coup}(s)$	block-diag			block-tri		
				its	$T_{sol}(s)$	$T_{tot}(s)$	its	$T_{sol}(s)$	$T_{tot}(s)$
4	68,070	8.80e-2	5.73	9	2.65e-1	1.22e+3	6	1.76e-1	1.18e+3
8	135,870	9.90e-2	13.63	10	5.54e-1	2.84e+3	6	3.50e-1	2.80e+3
16	269,766	1.04e-1	15.94	10	9.30e-1	3.35e+3	6	5.87e-1	3.31e+3
32	539,926	1.94e-1	28.00	10	1.54	5.90e+3	6	1.01	5.80e+3
64	1,074,054	1.89e-1	59.62	10	2.85	1.22e+4	6	1.78	1.27e+4
128	2,152,614	1.88e-1	198.75	10	7.53	5.94e+4	6	4.30	3.88e+4

Table 3.4: Weak scalability for the linear solid model, coupling without mesh intersection. The simulations are run on the Shaheen cluster. Same format as Table 3.1.

The time step is also fixed to $\Delta t = 0.01$.

We remark that the computation of the speed up factor S^p is performed considering the total time T_{tot} of the 4 processors run as reference value.

The results are reported in Table 3.5 and Table 3.6 for exact and approximate coupling respectively. From both tables we can see that T_{coup} is perfectly scalable as also graphically represented in Figure 3.4a. For both techniques the logarithm of T_{coup} scales almost linearly with respect to the number of processors.

When we use the composite quadrature rule for assembling $\mathsf{C}_f(\mathbf{X}_h^n)$, the two preconditioners have different behaviors. For block–diag we have a large number of GMRES iterations, always bigger than 140, which remains almost constant when we increase the number of processors. Thus, the time T_{sol} for solving the linear system is not scalable and the actual speed up values are far from the theoretical estimate. On the other hand, block–tri is more robust: the average number *its* is small, always equal to 12 or 13, producing small values for T_{sol} too. Even if T_{sol} is not perfectly scalable, the solver is globally scalable since T_{coup} is scalable and T_{coup} is remarkably larger than T_{sol}. In Figure 3.4b, we compare the speed up values for the two preconditioners. We can see that the blue line, corresponding to block–diag, is very far from the ideal case (dashed line) in contrast with the red line, which is related to block–tri and under estimated only for the 256 processors run.

From Table 3.6, we can analyze our solver when approximate integration for

Linear solid model – Strong scalability test

Coupling with mesh intersection

DOFs = 478470, T = 2, $\Delta t = 0.01$

procs	$T_{asm}(s)$	$T_{coup}(s)$	block-diag				block-tri			
			its	$T_{sol}(s)$	T_{tot} (s)	S^p	its	$T_{sol}(s)$	T_{tot} (s)	S^p
4	6.41e-1	498.08	192	41.04	1.08e+5	-	12	2.52	9.85e+4	-
8	3.47e-1	169.69	168	21.60	3.83e+4	2.82 (2)	13	1.49	3.42e+4	2.88 (2)
16	1.78e-1	89.18	180	18.09	2.15e+4	5.02 (4)	12	1.20	1.79e+4	5.50 (4)
32	1.65e-1	26.45	192	19.42	9.17e+3	11.78 (8)	12	1.22	5.53e+3	17.81 (8)
64	8.32e-2	17.38	165	23.09	8.09e+3	13.35 (16)	13	1.57	3.64e+3	27.06 (16)
128	4.12e-2	8.52	170	40.58	9.82e+3	11.00 (32)	12	3.02	2.35e+3	41.91 (32)
256	2.03e-2	4.08	144	68.31	1.45e+4	7.45 (64)	12	5.91	2.00e+3	49.25 (64)

Table 3.5: Strong scalability for the linear solid model, coupling with mesh intersection. The simulations are run on the Shaheen cluster. DOFs = degrees of freedom; procs = number of processors; T_{asm} = CPU time to assemble the stiffness and mass matrices; T_{coup} = CPU time to assemble the coupling term; its = GMRES iterations; T_{sol} = CPU time to solve the linear system; T_{tot} = total simulation time; S^p = parallel speedup computed with respect to the 4 processors run. The theoretical speed up is reported between brackets. The quantities T_{coup}, its and T_{sol} are averaged over the time steps. All CPU times are reported in seconds.

the coupling matrix is carried out. In this case, both preconditioners exhibit very small values of average GMRES iterations, 10 for block–diag and 6 for block–tri, remaining constant when the number of processes increases. As before, the solution time T_{sol} is not scalable, but thanks to the excellent scalability of T_{coup}, the solver appears to be strongly scalable in general. A plot showing the evolution of speed up values is reported in Figure 3.4c. For both preconditioners the actual factors are near or larger than the ideal ones.

Time step refinement

For this test we solve the linear problem with $478,470$ degrees of freedom and splitting the computational load over 64 processors. Five different choices of time step are considered: in particular, the largest value is $\Delta t = 0.02$, while the smallest one is $\Delta t = 0.001$.

The results are collected in Table 3.7 for the case of exact coupling, while for the approximate case we refer to Table 3.8. In both cases, the volume loss decreases

Linear solid model – Strong scalability test *Coupling with mesh intersection* DOFs = 478470, T = 2, $\Delta t = 0.01$										
procs	$T_{asm}(s)$	$T_{coup}(s)$	block-diag				block-tri			
			its	$T_{sol}(s)$	T_{tot} (s)	S^p	its	$T_{sol}(s)$	T_{tot} (s)	S^p
4	6.77e-1	1.68e+3	10	2.60	3.36e+5	-	6	1.69	3.37e+5	-
8	3.89e-1	741.99	10	1.87	1.50e+5	2.24 (2)	6	1.22	1.49e+5	2.26 (2)
16	1.79e-1	287.02	10	9.76e-1	5.76e+4	5.83 (4)	6	6.35e-1	5.75e+4	5.86 (4)
32	1.65e-1	108.92	10	9.95e-1	2.20e+4	15.27 (8)	6	6.21e-1	2.17e+4	15.53 (8)
64	7.70e-2	28.66	10	1.29	5.99e+3	56.09 (16)	6	8.68e-1	6.02e+3	55.98 (16)
128	4.04e-2	4.32	10	87.20	1.83e+4	18.36 (32)	6	58.70	1.26e+4	26.75 (32)
256	2.05e-2	1.63	10	5.04	1.33e+3	252.63(64)	6	3.05	951.11	354.32 (64)

Table 3.6: Strong scalability for the linear solid model, coupling without mesh intersection. The simulations are run on the Shaheen cluster. Same format as Table 3.5.

when Δt is refined. Moreover, since the spatial parameters do not change, T_{ass} remains constant with respect to the time step refinement. The time we need to assemble $\mathsf{C}_f(\mathbf{X}_h^n)$, T_{coup}, decreases when small values of Δt are chosen, since the geometric configuration does not significantly change between a time instant and the subsequent. This happens independently of the considered technique and thanks to the dynamic allocation of matrices since the sparsity pattern remains basically unchanged.

Moving on analyzing the performance of the two preconditioners, we have dissimilar results between Table 3.7 and Table 3.8. In Table 3.7, we can see that block–diag is not robust when coarse values of Δt are used, indeed for $\Delta t = 0.02$, the number of iterations its is around $1,300$, which is huge, especially if compared with the value related to block–tri, which is bounded by 19. On the other hand, from Table 3.8 both preconditioners are good since the number of iterations is bounded by 12 for block–diag and 7 for block–tri. Clearly, the average time T_{sol} to solve the system and the total T_{tot} directly depend on the number of iterations.

Volume loss

In this paragraph, we study how the choice of meshes, preconditioner and assembly technique for the interface matrix affect the deformation of the immersed solid body. In order to check if admissible results are produced, we analyze the

Linear solid model – Time step refinement test

Coupling with mesh intersection

DOFs = 478470, procs = 64, T = 2, Δt = 0.01

Δt	vol. loss (%)	$T_{asm}(s)$	$T_{coup}(s)$	block-diag			block-tri		
				its	$T_{sol}(s)$	$T_{tot}(s)$	its	$T_{sol}(s)$	$T_{tot}(s)$
0.02	2.55e-1	1.01e-1	15.06	1364	190.66	2.05e+4	19	3.56	1.86e+3
0.01	2.33e-1	1.01e-1	11.83	170	23.69	7.48e+3	12	2.33	2.83e+3
0.005	2.04e-1	1.01e-1	10.86	30	5.70	6.68e+3	9	1.91	5.12e+3
0.002	1.88e-1	1.01e-1	9.90	12	2.61	1.26e+4	7	1.44	1.13e+4
0.001	1.81e-1	1.01e-1	9.91	8	1.73	2.28e+4	5	1.09	2.20e+4

Table 3.7: Refining the time step in the linear solid model, coupling with mesh intersection. The simulations are run on the Shaheen cluster. Same format as Table 3.1.

Linear solid model – Time step refinement test

Coupling without mesh intersection

DOFs = 478470, procs = 64, T = 2, Δt = 0.01

Δt	vol. loss (%)	$T_{asm}(s)$	$T_{coup}(s)$	block-diag			block-tri		
				its	$T_{sol}(s)$	$T_{tot}(s)$	its	$T_{sol}(s)$	$T_{tot}(s)$
0.02	6.27e-2	1.01e-1	45.78	12	2.41	4.79e+3	7	1.45	4.72e+3
0.01	4.24e-2	1.01e-1	32.52	10	2.15	7.50e+3	6	1.36	6.78e+3
0.005	3.23e-2	1.01e-1	23.55	9	1.85	1.04e+4	5	1.17	9.91e+3
0.002	2.58e-2	1.02e-1	14.57	7	1.57	1.60e+4	4	9.62e-1	1.55e+4
0.001	2.37e-2	1.01e-1	10.07	6	1.42	2.32e+4	4	9.60e-1	2.21e+4

Table 3.8: Refining the time step in the linear solid model, coupling without mesh intersection. The simulations are run on the Shaheen cluster. Same format as in Table 3.1.

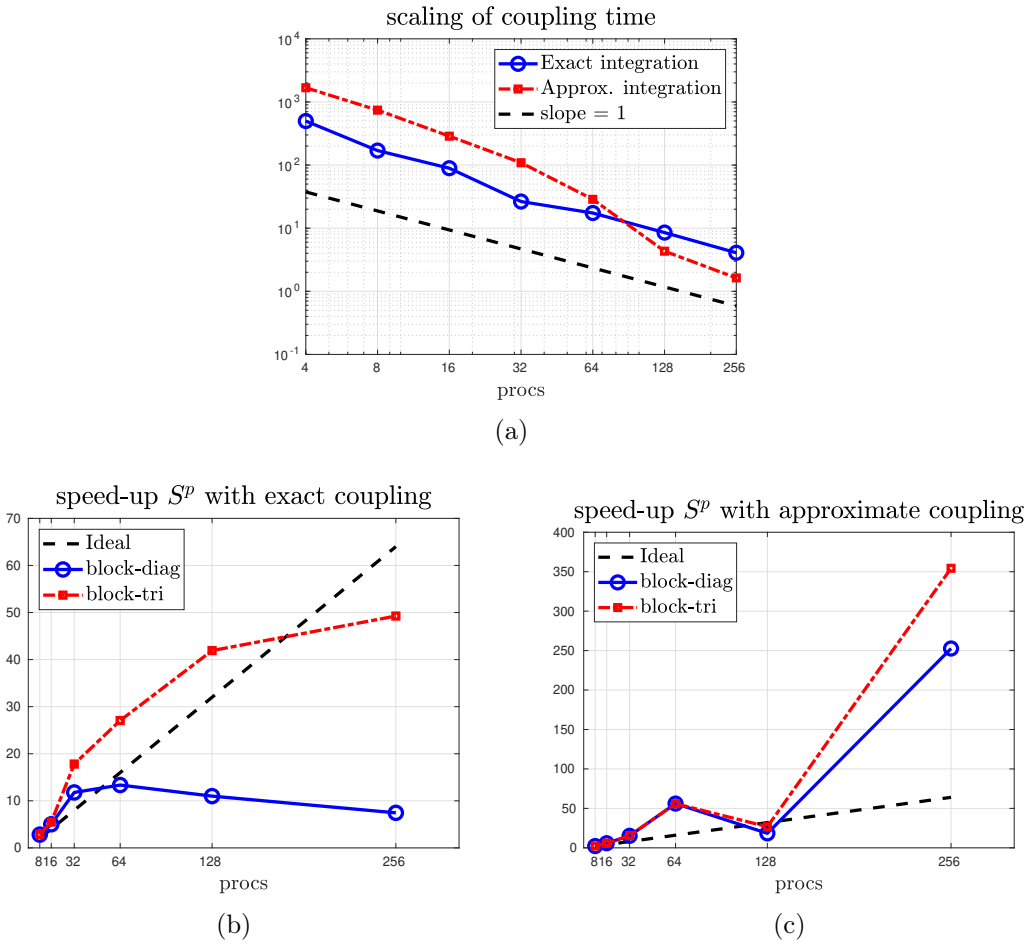


Figure 3.4: Graphic representation of the strong scalability tests reported in Table 3.5 and Table 3.6. In the first plot, the scaling of T_{coup} is represented in logarithmic scale: the blue line refers to the case with exact integration, while the red line is related to the approximate integration procedure. In both cases the decay rate is 1, accordingly to the theoretical expectation. In the second line, we report the speed up factor with respect to the number of processes: on left hand side we have the exact coupling case, on the right the approximate one. Blue lines are related to block–diag, while red lines are related to block–tri. The ideal speed up is represented by the black dashed line.

percentage of volume loss between the last and the first simulated time instants.

In particular, for our study we fix the solid mesh and we refine the fluid one.

For the first test, the solid mesh $\mathcal{T}_h^{\mathcal{B}}$ is made of 384×192 elements, while for the second one, we have 192×96 solid elements. On the other hand, for the fluid we start with a 64×64 mesh and then we refine three times so that we have 128×128, 256×256 and finally 512×512 elements.

As we already previously observed, we have confirmation that the approximate computation of the coupling matrix produces a better mass conservation than the exact approach. Moreover, if $\mathcal{T}_h^{\mathcal{B}}$ is 384×192, we see that the parallel solver does not work when the block–diag preconditioner is combined with the exact construction of $\mathsf{C}_f(\mathbf{X}_h^n)$ and coarse fluid meshes are considered (64×64 and 128×128). Indeed, the results have no physical meaning since the volume loss is of 99.94% and 72.15% respectively. Conversely, for the other choice of $\mathcal{T}_h^{\mathcal{B}}$, we notice that, even if the inexact coupling performs better in terms of mass conservation, it presents a slight increase of volume loss when the fluid mesh is significantly finer than the solid discretization, i.e. when $\mathcal{T}_h^{\Omega}$ is made up of 512×512 elements. The use of block–tri does not produce degenerate situations and for all test cases the results are valid.

Linear solid model – Volume loss (%)				
procs = 64, T = 2, $\Delta t = 0.01$				
solid mesh 384×192				
fluid mesh	*with mesh intersection*		*without mesh intersection*	
	block-diag	block-tri	block-diag	block-tri
64×64	99.94	2.51e-1	6.59e-2	6.59e-2
128×128	72.15	2.24e-1	6.69e-2	6.69e-2
256×256	2.33e-1	2.34e-1	4.24e-2	4.24e-2
512×512	2.14e-1	2.14e-1	4.15e-2	4.15e-2
solid mesh 192×96				
fluid mesh	*with mesh intersection*		*without mesh intersection*	
	block-diag	block-tri	block-diag	block-tri
64×64	4.95e-1	4.95e-1	6.61e-2	6.62e-2
128×128	5.06e-1	5.06e-1	6.89e-2	6.89e-2
256×256	4.60e-1	4.60e-1	5.76e-2	5.76e-2
512×512	5.48e-1	5.48e-1	2.66e-1	2.67e-1

Table 3.9: Refining the fluid mesh keeping fixed the solid one. Loss of the structure volume in percentage.

3.2.2 Nonlinear solid model

In this test, the immersed solid is an isotropic hyperelastic material described by the exponential strain energy density

$$W(\mathbb{F}) = \frac{\gamma}{2\eta} \exp\big(\eta[\, I_1 - 2]\big), \tag{3.18}$$

where I_1 denotes the first invariant of the right Cauchy–Green deformation tensor $\mathbb{F}^\top\mathbb{F}$, i.e. $I_1 = trace(\mathbb{F}^\top\mathbb{F})$, while γ and η are two constant parameters representing material properties. In particular, $\gamma > 0$ is a stress-like parameter, while $\eta > 0$ is a non-dimensional constant. This kind of materials are commonly considered in cardiac simulations to model arterial walls and have been widely studied in [45]. We remark that thanks to the monotonic increase of the exponential, then the density W appears to be strictly local convex with respect to $\mathbb{F}^\top\mathbb{F}$ [68, 83].

For our test, we set the fluid domain Ω to be the unit square and we immerse the elastic bar described by

$$\Omega_0^s = \mathcal{B} = [0, 0.4] \times [0.45, 0.55] \tag{3.19}$$

We emphasize that the solid reference domain $\mathcal{B}$ corresponds to the initial resting configuration of the structure. In particular, the bar is anchored to the left edge of the fluid domain and the dynamics of the system is generated by a force pulling down the solid body. This force is applied to the right edge of Ω_0^s during the time interval $[0, 1]$. Once the force is released, the elastic properties of the material steer the structure back to rest.

We impose null Dirichlet conditions for $\mathbf{u}$ on the whole boundary $\partial\Omega$, while at initial time we set

$$\mathbf{u}(\mathbf{x}, 0) = 0$$
$$\mathbf{X}(\mathbf{s}, 0) = \mathbf{s}.$$

As for the linear model, fluid and solid densities assume same value 1, for the viscosities we choose $\nu_f = \nu_s = 0.2$. We set the physical parameters γ and η involved in the energy density W to be $\gamma = 1.333$ and $\eta = 9.242$.

The analysis of the parallel solver is carried out simulating the evolution of the system during the time interval $[0, 2]$ for mesh refinement test, strong scalability and time step refinement. The weak scalability is discussed simulating the interaction until final time $T = 0.1$. In Figure 3.5 we report some snapshots of the evolution until $T = 5$.

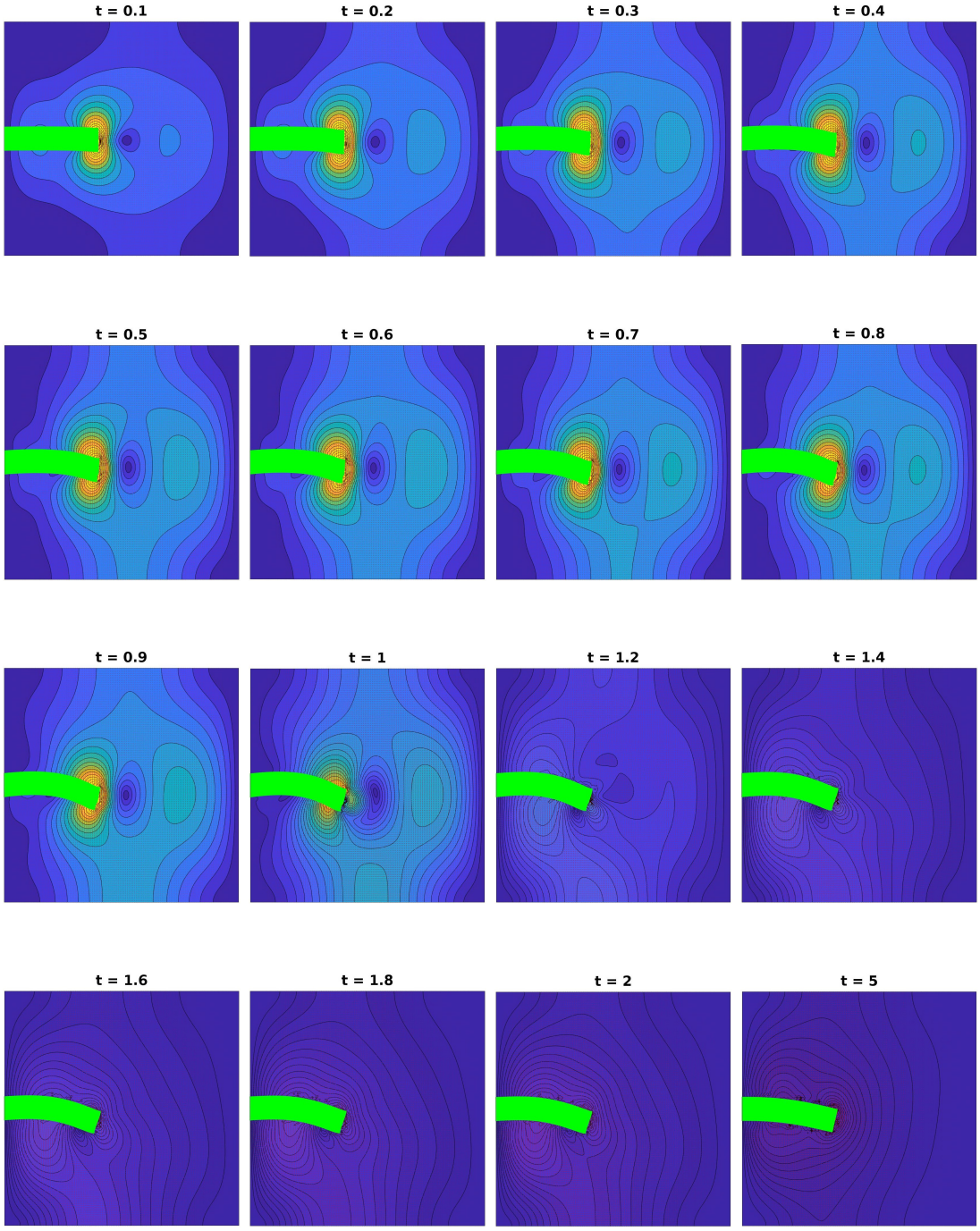


Figure 3.5: Some snapshots of the evolution in time of the elastic bar (nonlinear solid model).

Mesh refinement

We study the optimality of the solver for two different choices of time steps ($\Delta t = 0.01$ and $\Delta t = 0.002$) and number of processors (32 and 64, respectively). We initially consider a discretization consisting in 21,222 total DOFs and we refine both fluid and solid mesh five times, so that the finest tests is performed using a total amount of 741,702 DOFs.

We collect in Table 3.10 and Table 3.11 the results related to coupling with and without mesh intersection respectively computed by considering $\Delta t = 0.01$ and $procs = 32$. When $\mathsf{C}_f(\mathbf{X}_h^n)$ is computed exactly, the volume loss assumes more significant values: indeed, from the second test, it stabilizes at 1.47%, in contrast with the approximate coupling case for which we register a reduction of one order of magnitude between the first and the last simulation (from $3.50 \times 10^{-1}\%$ to $5.34 \times 10^{-2}\%$). For both techniques, T_{asm} exhibits a moderate increase, while T_{coup} grows superlinearly.

Analyzing the performance of the preconditioners, first we notice that the average number of Newton iterations nit is always bounded by 3, however in the case of coupling with mesh intersection it fails when the two finest spatial discretizations are considered. From both tables we can conclude that block–diag is not robust: the average number of GMRES linear iterations we need to solve the Jacobian linear system reaches 406 in Table 3.10 and even 3,001 in Table 3.11 when the problem is solved with 330,630 DOFs. Therefore, the solution time T_{sol} significantly grows with clear impact on T_{tot}. For block–tri the situation is different. The number of GMRES iteration its moderately increases, but remaining bounded by 30 and also T_{sol} is always smaller than $30\,s$. Therefore, we can say that block–tri is robust with respect to mesh refinement.

If we now consider a different time step, $\Delta t = 0.002$, and we distribute the computational load over 64 processors, the two preconditioners have slightly different behaviors: the simulation results are reported in Table 3.12 (with mesh intersection) and Table 3.13 (without mesh intersection). First, we notice that block–tri confirms its robustness since in both tables the related averaged iteration count per nonlinear iteration is bounded by 10 for the exact coupling and by 24 for the approximate case. On the other hand, block–diag becomes robust when the coupling is computed with mesh intersection: the GMRES iteration count is bounded by 15, which is very far from 406 we have when $\Delta t = 0.01$ is considered. Therefore, T_{sol} only moderately grows from $4.29 \times 10^{-1}\,s$ to $18.58\,s$. Unfortunately this

Nonlinear solid model – Mesh refinement test *Coupling with mesh intersection* procs = 32, T = 2, $\Delta t = 0.01$											
DOFs	vol. loss (%)	$T_{asm}(s)$	$T_{coup}(s)$	block-diag				block-tri			
				nit	its	$T_{sol}(s)$	$T_{tot}(s)$	nit	its	$T_{sol}(s)$	$T_{tot}(s)$
21,222	1.70	7.24e-3	6.08e-2	2	245	13.28	2.67e+3	2	21	1.56	3.24e+2
83,398	1.47	2.87e-2	8.41e-1	2	269	47.78	9.72e+3	2	23	5.57	1.28e+3
186,534	1.47	6.34e-2	4.10	3	388	1.55e+2	3.19e+4	3	26	14.38	3.71e+3
330,630	1.47	1.13e-1	9.05	3	406	2.59e+2	5.37e+4	3	27	23.95	6.62e+3
515,686	-	-	-	-	-	-	-	-	-	-	-
741,702	-	-	-	-	-	-	-	-	-	-	-

Table 3.10: Refining the mesh for the nonlinear solid model, coupling with mesh intersection. The simulations are run on the Shaheen cluster. procs = number of processors; DOFs = degrees of freedom; vol. loss = loss of structure volume in percentage; T_{asm} = CPU time to assemble the stiffness and mass matrices; T_{coup} = CPU time to assemble the coupling term; nit = Newton iterations; its = GMRES iterations to solve the Jacobian system; T_{sol} = CPU time to solve the Jacobian system; T_{tot} = total simulation CPU time. The quantities T_{coup} and nit are averaged over the time steps, whereas the quantities its and T_{sol} are averaged over the Newton iterations and the time steps. All CPU times are reported in seconds.

improvement is not confirmed for the non intersection case, for which we reach an average number of 516 linear iterations per Newton iteration, which is still large even if very far from the $3,000\, its$ we observed in Table 3.11.

Weak scalability

In order to study the weak scalability, we set the final time T to be just 0.1 so that considering $\Delta t = 0.002$, we post-process data of 50 time instants. The results of our simulations are reported in Table 3.14, for the case with mesh intersection, and in Table 3.15 for the case without mesh intersection. The number of processors is doubled four times starting from 4, while the number of DOFs goes from 64,014 to 1,008,678 accordingly with the same law, so that $\frac{\#DOFs}{\#cores}$ is almost kept constant.

As we already observed for the linear case, the assembly time T_{coup} of the interface matrix is not weakly scalable in general: when the exact quadrature is considered, it goes from $1.30\, s$ up to $41.81\, s$; on the other hand, for the approximate

Nonlinear solid model – Mesh refinement test											
Coupling without mesh intersection											
procs = 32, T = 2, Δt = 0.01											
DOFs	vol. loss (%)	$T_{asm}(s)$	$T_{coup}(s)$	block-diag				block-tri			
				nit	its	$T_{sol}(s)$	$T_{tot}(s)$	nit	its	$T_{sol}(s)$	$T_{tot}(s)$
21,222	3.50e-1	4.90e-3	1.61e-2	3	245	7.26	4.49e+3	3	11	5.47e-1	331.91
83,398	2.07e-1	2.04e-2	6.89e-2	3	498	43.32	2.70e+4	3	14	1.99	1.24e+3
186,534	1.36e-1	4.56e-2	2.15e-1	3	1,572	308.33	1.85e+5	3	16	4.92	3.18e+3
330,630	9.97e-2	8.20e-2	6.42e-1	3	3,001	924.25	5.55e+5	3	18	8.63	5.95e+3
515,686	7.16e-2	1.26e-1	1.85	-	-	-	-	3	25	17.71	1.31e+4
741,702	5.34e-2	3.19e-1	5.45	-	-	-	-	3	30	28.59	2.19e+4

Table 3.11: Refining the mesh in the nonlinear solid model, coupling without mesh intersection. The simulations are run on the EOS cluster. Same format as Table 3.10.

Nonlinear solid model – Mesh refinement test											
Coupling with mesh intersection											
procs = 64, T = 2, Δt = 0.001											
DOFs	vol. loss (%)	$T_{asm}(s)$	$T_{coup}(s)$	block-diag				block-tri			
				nit	its	$T_{sol}(s)$	$T_{tot}(s)$	nit	its	$T_{sol}(s)$	$T_{tot}(s)$
21,222	1.71	4.04e-3	3.89e-2	2	11	4.29e-1	9.35e+2	2	8	3.93e-1	8.64e+2
83,398	1.47	1.68e-2	3.60e-1	2	12	1.67	4.06e+3	2	8	1.57	3.86e+3
186,534	1.49	3.80e-2	1.57	2	14	4.23	1.16e+4	2	9	4.00	1.11e+4
330,630	1.48	6.68e-2	4.77	2	14	7.71	2.49e+4	2	10	7.07	2.37e+4
515,686	1.44	1.05e-1	11.40	2	15	13.03	4.92e+4	2	10	11.48	4.58e+4
741,702	1.42	1.52e-1	23.23	2	15	18.58	8.49e+4	2	10	16.63	7.98e+4

Table 3.12: Refining the mesh in the nonlinear solid model, coupling with mesh intersection. The simulations are run on the Shaheen cluster. Same format as Table 3.10.

Nonlinear solid model – Mesh refinement test											
Coupling without mesh intersection											
procs = 64, T = 2, Δt = 0.001											
DOFs	vol. loss (%)	$T_{asm}(s)$	$T_{coup}(s)$	block-diag				block-tri			
				nit	its	$T_{sol}(s)$	$T_{tot}(s)$	nit	its	$T_{sol}(s)$	$T_{tot}(s)$
21,222	4.09e-1	4.33e-3	2.65e-2	2	21	6.21e-1	1.30e+3	2	12	4.95e-1	1.04e+3
83,398	2.70e-1	1.70e-2	1.45e-1	3	87	5.50	1.13e+4	3	14	2.13	4.55e+3
186,534	1.98e-1	3.81e-2	4.75e-1	3	270	30.83	6.26e+4	3	18	5.77	1.25e+4
330,630	1.58e-1	6.69e-2	1.58	3	516	1.00e+2	2.04e+5	3	20	10.66	2.45e+4
515,686	1.31e-1	1.05e-1	8.38	-	-	-	-	3	23	17.62	5.20e+4
741,702	1.15e-1	1.52e-1	32.37	-	-	-	-	3	24	26.46	1.18e+5

Table 3.13: Refining the mesh in the nonlinear solid model, coupling without mesh intersection. The simulations are run on the Shaheen cluster. Same format as Table 3.10.

technique we get better results, not so far to be scalable, since the minimum value assumed by T_{coup} is $2.82 \times 10^{-1}\, s$, while the maximum value is $3.47\, s$.

In relation to the preconditioners, the results of the previous tests are confirmed. Block–diag cannot be used when the coupling term is inexactly constructed since the average number of GMRES iterations per nonlinear iteration assumes large values, such as $its = 1,952$ when $procs = 8$. Better results are reported in Table 3.14: the number of GMRES iterations grows from 23 to 68 affecting T_{sol}, which increases from $5.22\, s$ to $196.66\, s$: it is evident that, even if the results in this case seem to be better, this preconditioner is not scalable.

For block–tri the situation is different. When the coupling term is assembled with intersection, even if it is not perfectly scalable if we look at the solution time, the average number of linear iterations per Newton iteration is almost constant, increasing only from 15 to 18. The lack of scalability is evident for the other assembly technique: indeed, from Table 3.15, we can see that its doubles its value, from 20 to 40 between the first and the last simulation.

Strong scalability

The study of strong scalability is only performed on the block–tri preconditioner since we observed poor performance for block–diag. For this test, we fix both fluid and solid meshes so that the total amount of degrees of freedom corresponds to 515,686. The simulations are done increasing the number of processors from 4 to

Nonlinear solid model – Weak scalability test											
Coupling with mesh intersection											
T = 0.1, Δt = 0.002											
procs	DOFs	$T_{asm}(s)$	$T_{coup}(s)$	block-diag				block-tri			
				nit	its	$T_{sol}(s)$	$T_{tot}(s)$	nit	its	$T_{sol}(s)$	$T_{tot}(s)$
4	64,014	1.36e-1	1.30	3	23	5.22	326.29	3	15	4.17	273.29
8	129,846	9.45e-2	2.51	3	29	10.76	663.37	3	15	7.35	493.16
16	253,462	9.74e-2	5.17	3	41	26.16	1.57e+3	3	16	13.72	944.30
32	515,686	1.70e-1	13.53	3	58	77.10	4.53e+3	3	17	33.14	2.33e+3
64	1,008,678	1.77e-1	41.81	3	68	196.66	1.19e+4	3	18	78.42	6.01e+3

Table 3.14: Weak scalability for the nonlinear solid model, coupling with mesh intersection. The simulations are run on the Shaheen cluster. Same format as Table 3.10.

Nonlinear solid model – Weak scalability test											
Coupling without mesh intersection											
T = 0.1, Δt = 0.002											
procs	DOFs	$T_{asm}(s)$	$T_{coup}(s)$	block-diag				block-tri			
				nit	its	$T_{sol}(s)$	$T_{tot}(s)$	nit	its	$T_{sol}(s)$	$T_{tot}(s)$
4	64,014	8.27e-2	3.40e-1	3	89	14.46	739.84	3	20	5.16	275.04
8	129,846	9.52e-2	2.82e-1	3	1952	502.61	2.51e+4	3	29	11.34	581.25
16	253,462	9.61e-2	4.21e-1	3	722	361.60	1.81e+4	3	26	19.79	1.01e+3
32	515,686	1.81e-1	1.14	-	-	-	-	3	57	79.67	4.04e+3
64	1,008,678	1.74e-1	3.47	-	-	-	-	3	40	136.61	7.00e+3

Table 3.15: Weak scalability for the nonlinear solid model, coupling without mesh intersection. The simulations are run on the Shaheen cluster. Same format as Table 3.10.

64. As for the linear problem, the speed up factor S^p is computed referring to the total time T_{tot} of the 4 processors run.

Table 3.16 shows the results of the simulation carried out with exact coupling. The final time is set to $T = 2$ and the time step is $\Delta t = 0.002$. On the other hand, in Table 3.17 we collect the results for the simulation with approximate coupling: for this case, we have again $T = 2$, but we choose a different time step, $\Delta t = 0.01$.

For both tests, the values of T_{coup} confirm the behavior we already observed with the linear model: in the case with intersection it scales linearly as we increase the number of processors, while for the non intersection test the rate is even higher. Moreover, the number of Newton iterations nit and the average number of GMRES iterations present a scalable behavior since they remain bounded: in particular, nit is always bounded by 3, while its is bounded by 6 in Table 3.16 and by 33 in Table 3.17. However, the solution time T_{sol} is not scalable, therefore T_{sol} impairs the global performance of the solver producing speed up values far from the theoretical estimates.

Time step refinement

As already done for the linear solid model, we discuss how the solver behaves when several choices of time step are considered; in particular the coarsest case corresponds to $\Delta t = 0.02$, while the finest one is related to $\Delta t = 0.001$. We set again $T = 2$ and we discretize the problem with a total amount of 515,686 degrees of freedom.

The results of our simulations are reported in Table 3.18 (coupling with mesh intersection) and Table 3.19 (coupling without mesh intersection). We notice that, for the first case, the volume loss is constant, while in the second case, it increases by one order of magnitude when the time step is refined. This results are in line with all the previous tests.

Looking at the assembly times, T_{asm} keeps a constant value. On the other hand, T_{coup} is in average almost constant when $\mathsf{C}_f(\mathbf{X}_h^n)$ is assembled exactly, assuming values between $11\,s$ and $12\,s$, whereas it drastically decreases (from $51.20\,s$ to $8.62\,s$) when $\mathsf{C}_f(\mathbf{X}_h^n)$ is computed with approximated integrals.

In terms of preconditioners, we see again that block–diag is not robust. From Table 3.18, it behaves well only when a very fine time step is considered (for $\Delta t = 0.001$ the average number of linear iterations per nonlinear iteration is 16), but this is not anymore true when the interface matrix is assembled without mesh

Nonlinear solid model – Strong scalability test *Coupling with mesh intersection*							
DOFs = 515686, T = 2, $\Delta t = 0.002$							
procs	$T_{asm}(s)$	$T_{coup}(s)$	block-tri				
			nit	its	$T_{sol}(s)$	T_{tot} (s)	S^p
4	6.79e-1	80.89	2	6	11.17	1.05e+5	-
8	3.96e-1	38.71	2	6	9.02	5.79e+4	1.81 (2)
16	2.71e-1	21.10	2	6	9.33	4.10e+4	2.56 (4)
32	1.71e-1	12.53	2	6	7.55	2.86e+4	3.67 (8)
64	1.00e-1	8.44	2	6	13.29	3.68e+4	2.85 (16)

Table 3.16: Strong scalability for the nonlinear solid model, coupling with mesh intersection. The simulations are run on the EOS cluster. DOFs = degrees of freedom; procs = number of processors; T_{asm} = CPU time to assemble the stiffness and mass matrices; T_{coup} = CPU time to assemble the coupling term; nit = Newton iterations; its = GMRES iterations to solve the Jacobian system; T_{sol} = CPU time to solve the Jacobian system; T_{tot} = total simulation CPU time; S^p = parallel speedup computed with respect to the 4 processors run. The theoretical speed up is reported between brackets. The quantities T_{coup} and nit are averaged over the time steps, whereas the quantities its and T_{sol} are averaged over the Newton iterations and the time steps. All CPU times are reported in seconds.

intersection. Conversely, block–tri seems more robust: for the intersection case, its is bounded by 19 and it reduces when Δt decreases, whereas it assumes larger values for simulations without mesh intersection: in this case, its decreases from 193 ($\Delta t = 0.02$) to 23 ($\Delta t = 0.001$).

We remark that in some cases the solver fails due to the nonlinear Newton iterator.

3.3 Final remarks

As already mentioned before, the design of efficient parallel solvers for coupled problems is not an easy task under several aspects. For instance, the discretization of the coupling term requires computations over non matching meshes. In our

Nonlinear solid model – Strong scalability test							
Coupling without mesh intersection							
DOFs = 515686, T = 2, Δt = 0.01							
procs	$T_{asm}(s)$	$T_{coup}(s)$	block-tri				
			nit	its	$T_{sol}(s)$	T_{tot} (s)	S^p
4	6.31e-1	185.50	3	33	30.49	5.93e+4	-
8	3.75e-1	22.85	3	33	27.62	2.48e+4	2.39 (2)
16	1.79e-1	4.74	3	29	24.79	1.91e+4	3.10 (4)
32	1.19e-1	2.51	3	25	18.26	1.36e+4	4.36 (8)
64	1.05e-1	8.77e-1	3	22	25.24	1.81e+4	3.28 (16)

Table 3.17: Strong scalability for the nonlinear solid model, coupling without mesh intersection. The simulations are run on the EOS cluster. Same format as Table 3.16.

Nonlinear solid model – Time step refinement test											
Coupling with mesh intersection											
DOFs = 515686, procs = 64, T = 2											
Δt	vol. loss (%)	$T_{asm}(s)$	$T_{coup}(s)$	block-diag				block-tri			
				nit	its	$T_{sol}(s)$	$T_{tot}(s)$	nit	its	$T_{sol}(s)$	$T_{tot}(s)$
0.02	-	-	-	-	-	-	failed	-	-	-	failed
0.01	-	-	-	-	-	-	failed	-	-	-	failed
0.005	1.44	1.06e-1	12.04	2	227	71.35	3.34e+4	2	19	14.97	1.08e+4
0.002	1.44	1.06e-1	11.43	2	59	25.49	3.70e+4	2	13	12.53	2.40e+4
0.001	1.44	1.06e-1	11.24	2	16	13.14	4.91e+4	2	10	11.44	4.54e+4

Table 3.18: Refining the time step Δt for the nonlinear solid model, coupling with mesh intersection. The simulations are run on Shaheen cluster. Same format as Table 3.10.

Nonlinear solid model – Time step refinement test											
Coupling without mesh intersection											
DOFs = 515686, procs = 64, T = 2											
Δt	vol. loss (%)	$T_{asm}(s)$	$T_{coup}(s)$	block-diag				block-tri			
				nit	its	$T_{sol}(s)$	$T_{tot}(s)$	nit	its	$T_{sol}(s)$	$T_{tot}(s)$
0.02	1.08e-2	1.06e-1	51.20	-	-	-	failed	4	193	65.66	1.17e+4
0.01	7.16e-2	1.06e-1	35.86	-	-	-	failed	3	80	37.16	1.46e+4
0.005	1.04e-1	1.06e-1	24.28	-	-	-	failed	3	48	26.25	2.03e+4
0.002	1.24e-1	1.05e-1	13.49	3	955	283.82	2.95e+5	3	30	20.69	3.42e+4
0.001	1.31e-1	8.66e-2	8.62	3	710	214.62	4.46e+5	3	23	17.89	5.30e+4

Table 3.19: Refining the time step Δt for the nonlinear solid model, coupling without mesh intersection. The simulations are run on Shaheen cluster. Same format as Table 3.18.

implementation we have chosen the easiest algorithm one can consider: it is based on two nested loops with only the external one distributed over the processors so that each solid element is tested against all fluid elements. This choice affects the properties of the solver: indeed, the procedure is strongly scalable but not weakly scalable. In this second case, the increase of run time is significant. In order to improve this procedure, other techniques may be considered for future studies: for instance, one can keep track of the position of the solid body at the previous time step to reduce the number of element by element pairs to be tested; an alternative approach may be designed considering a *K–nearest neighbors* algorithm so that at each time step a solid element is tested with few fluid elements in dependence of its position.

Regarding the preconditioners, it is clear that block–diag is not suitable under several aspects: in the case of the linear solid model, it suffers when combined with the exact integration of the coupling term, showing an iteration count larger than 100. For the nonlinear solid model, it performs poorly when combined with the inexact assembly of C_f, since the order of magnitude of the average GMRES iteration count is 10^3. Conversely, the overall behavior of block–tri is acceptable since its performance is not strongly influenced by the choice of assembly technique for the interface matrix. Block–tri behaves well for both class of problems, while is strongly scalable only when applied to linear problems. The lack of weak scalability is evident, but more acceptable than for block–diag. Another limitation may be represented by the exact inversion of the diagonal blocks: this operation is

slow when very fine meshes are considered. This choice has been done since some preliminary tests showed bad results when the solid block $\mathbb{A}_{22}$ is inexactly inverted. This is probably the main aspect we should improve in future studies.

Finally, from the modeling point of view, several extensions are possible. For instance, we may consider the full Navier–Stokes problem reintroducing the non-linear convective term, while for the structure, we may consider also anisotropic constitutive laws. Finally, the biggest challenge is represented by the simulation of three dimensional problems related to real world phenomena.

Bibliography

[1] F. Alauzet, B. Fabrèges, M. A. Fernández, and M. Landajuela. Nitsche-XFEM for the coupling of an incompressible fluid with immersed thin-walled structures. *Computer Methods in Applied Mechanics and Engineering*, 301:300–335, 2016.

[2] N. Alshehri, D. Boffi, and L. Gastaldi. Unfitted mixed finite element methods for elliptic interface problems. *Numerical Methods for Partial Differential Equations*, 1-24, 2023.

[3] P. R. Amestoy, I. S. Duff, J.-Y. L'Excellent, and J. Koster. A fully asynchronous multifrontal solver using distributed dynamic scheduling. *SIAM J. Matr. Anal. Appl.*, 23(1):15–41, 2001.

[4] P. R. Amestoy, A. Guermouche, J.-Y. L'Excellent, and S. Pralet. Hybrid scheduling for the parallel solution of linear systems. *Paral. Comput.*, 32(2):136–156, 2006.

[5] D. Arndt, W. Bangerth, M. Feder, M. Fehling, R. Gassmöller, T. Heister, L. Heltai, M. Kronbichler, M. Maier, P. Munch, et al. The deal.II library, version 9.4. *Journal of Numerical Mathematics*, 30(3):231–246, 2022.

[6] P. J. Atzberger, P. R. Kramer, and C. S. Peskin. A stochastic immersed boundary method for fluid-structure dynamics at microscopic length scales. *Journal of Computational Physics*, 224(2):1255–1292, 2007.

[7] I. Babuška. The finite element method with Lagrangian multipliers. *Numerische Mathematik*, 20(3):179–192, 1973.

[8] S. Badia, F. Nobile, and C. Vergara. Fluid–structure partitioned procedures based on Robin transmission conditions. *Journal of Computational Physics*, 227(14):7027–7051, 2008.

[9] S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, D. A. May, L. C. McInnes, R. T. Mills, T. Munson, K. Rupp, P. Sanan, B. F. Smith, S. Zampini, H. Zhang, and H. Zhang. PETSc users manual. Technical Report ANL-95/11 - Revision 3.9, Argonne National Laboratory, 2018.

[10] S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, D. A. May, L. C. McInnes, R. T. Mills, T. Munson, K. Rupp, P. Sanan, B. F. Smith, S. Zampini, H. Zhang, and H. Zhang. PETSc Web page. http://www.mcs.anl.gov/petsc, 2018.

[11] D. Balzani, S. Deparis, S. Fausten, D. Forti, A. Heinlein, A. Klawonn, A. Quarteroni, O. Rheinbach, and J. Schroeder. Numerical modeling of fluid–structure interaction in arteries with anisotropic polyconvex hyperelastic and anisotropic viscoelastic material models at finite strains. *International Journal for Numerical Methods in Biomedical Engineering*, page e02756, 2016.

[12] A. T. Barker and X.-C. Cai. Scalable parallel methods for monolithic coupling in fluid–structure interaction with application to blood flow modeling. *Journal of Computational Physics*, 229:642–659, 2010.

[13] M. Bercovier and O. Pironneau. Error estimates for finite element method solution of the Stokes problem in the primitive variables. *Numerische Mathematik*, 33(2):211–224, 1979.

[14] R. P. Beyer. A computational model of the cochlea using the immersed boundary method. *Journal of Computational Physics*, 98(1):145–162, 1992.

[15] D. Boffi, F. Brezzi, and M. Fortin. *Mixed finite element methods and applications*, volume 44. Springer, 2013.

[16] D. Boffi, A. Cangiani, M. Feder, L. Gastaldi, and L. Heltai. A comparison of non-matching techniques for the finite element approximation of interface problems. *Comput. Math. Appl.*, 151:101–115, 2023.

[17] D. Boffi, N. Cavallini, F. Gardini, and L. Gastaldi. Local mass conservation of Stokes finite elements. *Journal of scientific computing*, 52(2):383–400, 2012.

[18] D. Boffi, N. Cavallini, and L. Gastaldi. Finite element approach to immersed boundary method with different fluid and solid densities. *Mathematical Models and Methods in Applied Sciences*, 21(12):2523–2550, 2011.

[19] D. Boffi, N. Cavallini, and L. Gastaldi. The finite element immersed boundary method with distributed Lagrange multiplier. *SIAM Journal on Numerical Analysis*, 53(6):2584–2604, 2015.

[20] D. Boffi, F. Credali, and L. Gastaldi. On the interface matrix for fluid–structure interaction problems with fictitious domain approach. *Computer Methods in Applied Mechanics and Engineering*, 401:115650, 2022.

[21] D. Boffi, F. Credali, L. Gastaldi, and S. Scacchi. A parallel solver for fluid structure interaction problems with Lagrange multiplier. *arXiv preprint arXiv:2212.13410*, 2022.

[22] D. Boffi, F. Credali, L. Gastaldi, and S. Scacchi. A parallel solver for FSI problems with fictitious domain approach. *Mathematical and Computational Applications*, 28(2), 2023.

[23] D. Boffi and L. Gastaldi. A finite element approach for the immersed boundary method. *Computers & structures*, 81(8-11):491–501, 2003.

[24] D. Boffi and L. Gastaldi. A fictitious domain approach with Lagrange multiplier for fluid-structure interactions. *Numerische Mathematik*, 135(3):711–732, 2017.

[25] D. Boffi and L. Gastaldi. On the existence and the uniqueness of the solution to a fluid-structure interaction problem. *Journal of Differential Equations*, 279:136–161, 2021.

[26] D. Boffi and L. Gastaldi. Existence, uniqueness, and approximation of a fictitious domain formulation for fluid-structure interactions. *Rendiconti Lincei*, 33(1):109–137, 2022.

[27] D. Boffi, L. Gastaldi, and L. Heltai. Numerical stability of the finite element immersed boundary method. *Mathematical Models and Methods in Applied Sciences*, 17(10):1479–1505, 2007.

[28] D. Boffi, L. Gastaldi, and L. Heltai. On the CFL condition for the finite element immersed boundary method. *Computers & Structures*, 85(11-14):775–783, 2007.

[29] D. Boffi, L. Gastaldi, L. Heltai, and C. S. Peskin. On the hyper-elastic formulation of the immersed boundary method. *Computer Methods in Applied Mechanics and Engineering*, 197(25-28):2210–2231, 2008.

[30] D. Boffi, L. Gastaldi, and M. Ruggeri. Mixed formulation for interface problems with distributed Lagrange multiplier. *Computers & Mathematics with Applications*, 68(12):2151–2166, 2014.

[31] D. Boffi, L. Gastaldi, and S. Wolf. Higher-order time-stepping schemes for fluid-structure interaction problems. *Discrete & Continuous Dynamical Systems-B*, 22(11):3807–3830, 2017.

[32] S. C. Brenner and L. R. Scott. *The mathematical theory of finite element methods*, volume 3. Springer, 2008.

[33] J. Brezina and P. Exner. Fast algorithms for intersection of non-matching grids using Plücker coordinates. *Computers & Mathematics with Applications*, 74(1):174–187, 2017.

[34] F. Brezzi. On the existence, uniqueness and approximation of saddle-point problems arising from Lagrangian multipliers. *Publications mathématiques et informatique de Rennes*, (S4):1–26, 1974.

[35] E. Burman and M. A. Fernández. An unfitted Nitsche method for incompressible fluid–structure interaction using overlapping meshes. *Computer Methods in Applied Mechanics and Engineering*, 279:497–514, 2014.

[36] P. Causin, J.-F. Gerbeau, and F. Nobile. Added-mass effect in the design of partitioned algorithms for fluid–structure problems. *Computer Methods in Applied Mechanics and Engineering*, 194(42):4506–4527, 2005.

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

[38] A. Chaudhuri, A. Hadjadj, and A. Chinnayya. On the use of immersed boundary methods for shock/obstacle interactions. *Journal of Computational Physics*, 230(5):1731–1748, 2011.

[39] W. Chen, M. Gunzburger, D. Sun, and X. Wang. Efficient and long-time accurate second-order methods for the Stokes–Darcy system. *SIAM Journal on Numerical Analysis*, 51(5):2563–2584, 2013.

[40] Z. Chen and C. S. Peskin. A Fourier spectral immersed boundary method with exact translation invariance, improved boundary resolution, and a divergence-free velocity field. *arXiv preprint arXiv:2302.08694*, 2023.

[41] P. G. Ciarlet. *The finite element method for elliptic problems*. SIAM, 2002.

[42] A. Cioncolini and D. Boffi. The MINI mixed finite element for the Stokes problem: An experimental investigation. *Computers & Mathematics with Applications*, 77(9):2432–2446, 2019.

[43] A. Cioncolini and D. Boffi. Superconvergence of the MINI mixed finite element discretization of the Stokes problem: An experimental study in 3D. *Finite Elements in Analysis and Design*, 201:103706, 2022.

[44] P. Crosetto, S. Deparis, G. Fourestey, and A. Quarteroni. Parallel algorithms for fluid-structure interaction problems in haemodynamics. *SIAM Journal on Scientific Computing*, 33(4):1598–1622, 2011.

[45] A. Delfino, N. Stergiopulos, J. Moore Jr, and J.-J. Meister. Residual strain effects on the stress field in a thick wall finite element model of the human carotid bifurcation. *Journal of biomechanics*, 30(8):777–786, 1997.

[46] S. Deparis, M. A. Fernández, and L. Formaggia. Acceleration of a fixed point algorithm for fluid-structure interaction using transpiration conditions. *ESAIM: Mathematical Modelling and Numerical Analysis*, 37(4):601–616, 2003.

[47] S. Deparis, D. Forti, G. Grandperrin, and A. Quarteroni. FaCSI: A block parallel preconditioner for fluid–structure interaction in hemodynamics. *Journal of Computational Physics*, 327:700–718, 2016.

[48] J. Donéa, P. Fasoli-Stella, and S. Giuliani. Lagrangian and Eulerian finite element techniques for transient fluid-structure interaction problems. 1977.

[49] J. Donea, A. Huerta, J.-P. Ponthot, and A. Rodríguez-Ferran. Arbitrary Lagrangian-Eulerian methods. *Encyclopedia of computational mechanics*, 2004.

[50] S. Dong. BDF-like methods for nonlinear dynamic analysis. *Journal of Computational physics*, 229(8):3019–3045, 2010.

[51] T. Dupont and R. Scott. Polynomial approximation of functions in Sobolev spaces. *Mathematics of Computation*, 34(150):441–463, 1980.

[52] R. Durán, L. Gastaldi, and A. Lombardi. Analysis of finite element approximations of Stokes equations with nonsmooth data. *SIAM Journal on Numerical Analysis*, 58(6):3309–3331, 2020.

[53] H. Eichel, L. Tobiska, and H. Xie. Supercloseness and superconvergence of stabilized low-order finite element discretizations of the Stokes problem. *Mathematics of computation*, 80(274):697–722, 2011.

[54] P. Farrell and J. Maddison. Conservative interpolation between volume meshes by local Galerkin projection. *Computer Methods in Applied Mechanics and Engineering*, 200(1-4):89–100, 2011.

[55] L. J. Fauci and C. S. Peskin. A computational model of aquatic animal locomotion. *Journal of Computational Physics*, 77(1):85–108, 1988.

[56] M. A. Fernández and M. Moubachir. A Newton method using exact Jacobians for solving fluid–structure coupling. *Computers & Structures*, 83(2-3):127–142, 2005.

[57] J. E. Fromm, N. Wunsch, R. Xiang, H. Zhao, K. Maute, J. A. Evans, and D. Kamensky. Interpolation-based immersed finite element and isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering*, 405:115890, 2023.

[58] C. Geuzaine and J.-F. Remacle. Gmsh: A 3-D finite element mesh generator with built-in pre-and post-processing facilities. *International journal for numerical methods in engineering*, 79(11):1309–1331, 2009.

[59] V. Girault and P.-A. Raviart. *Finite element approximation of the Navier-Stokes equations*, volume 749. Springer Berlin, 1979.

[60] R. Glowinski, T.-W. Pan, T. I. Hesla, D. D. Joseph, and J. Periaux. A fictitious domain approach to the direct numerical simulation of incompressible viscous flow past moving rigid bodies: application to particulate flow. *Journal of computational physics*, 169(2):363–426, 2001.

[61] R. Glowinski, T.-W. Pan, and J. Periaux. A Lagrange multiplier/fictitious domain method for the numerical simulation of incompressible viscous flow around moving rigid bodies:(i) case where the rigid body motions are known a priori. *Comptes Rendus de l'Académie des Sciences-Series I-Mathematics*, 324(3):361–369, 1997.

[62] S. Hartmann and P. Neff. Polyconvexity of generalized polynomial-type hyperelastic strain energy functions for near-incompressibility. *International journal of solids and structures*, 40(11):2767–2791, 2003.

[63] A. Heinlein, A. Klawonn, and O. Rheinbach. A parallel implementation of a two-level overlapping Schwarz method with energy-minimizing coarse space based on Trilinos. *SIAM Journal on Scientific Computing*, 38(6):C713–C747, 2016.

[64] L. Heltai. *The finite element immersed boundary method*. PhD thesis, 2006.

[65] L. Heltai. On the stability of the finite element immersed boundary method. *Computers & Structures*, 86(7-8):598–617, 2008.

[66] L. Heltai and F. Costanzo. Variational implementation of immersed finite element methods. *Computer Methods in Applied Mechanics and Engineering*, 229:110–127, 2012.

[67] C. W. Hirt, A. A. Amsden, and J. Cook. An arbitrary Lagrangian-Eulerian computing method for all flow speeds. *Journal of computational physics*, 14(3):227–253, 1974.

[68] G. A. Holzapfel, T. C. Gasser, and R. W. Ogden. A new constitutive framework for arterial wall mechanics and a comparative study of material models. *Journal of elasticity and the physical science of solids*, 61:1–48, 2000.

[69] G. Hou, J. Wang, and A. Layton. Numerical methods for fluid-structure interaction—a review. *Communications in Computational Physics*, 12(2):337–377, 2012.

[70] T. J. 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.

[71] D. Jodlbauer, U. Langer, and T. Wick. Parallel block–preconditioned monolithic solvers for fluid-structure interaction problems. *Internation Journal for Numerical Methods in Engineering*, 117:623–643, 2019.

[72] D. Kamensky, M.-C. Hsu, D. Schillinger, J. A. Evans, A. Aggarwal, Y. Bazilevs, M. S. Sacks, and T. J. Hughes. An immersogeometric variational framework for fluid–structure interaction: Application to bioprosthetic heart valves. *Computer methods in applied mechanics and engineering*, 284:1005–1053, 2015.

[73] Y. Kim and C. S. Peskin. Penalty immersed boundary method for an elastic boundary with mass. *Physics of Fluids*, 19(5), 2007.

[74] R. Krause and P. Zulian. A parallel approach to the variational transfer of discrete fields between arbitrarily distributed unstructured finite element meshes. *SIAM Journal on Scientific Computing*, 38(3):C307–C333, 2016.

[75] M.-C. Lai and C. S. Peskin. An immersed boundary method with formal second-order accuracy and reduced numerical viscosity. *Journal of Computational Physics*, 160(2):705–719, 2000.

[76] C. Lang, D. Makhija, A. Doostan, and K. Maute. A simple and efficient preconditioning scheme for heaviside enriched XFEM. *Computational Mechanics*, 54:1357–1374, 2014.

[77] C. Lehrenfeld and A. Reusken. Optimal preconditioners for Nitsche-XFEM discretizations of interface problems. *Numerische Mathematik*, 135:313–332, 2017.

[78] S. Lim and C. S. Peskin. Fluid-mechanical interaction of flexible bacterial flagella by the immersed boundary method. *Physical Review E*, 85(3):036307, 2012.

[79] J. L. Lions and E. Magenes. *Non-homogeneous boundary value problems and applications: Vol. 1*, volume 181. Springer Science & Business Media, 2012.

[80] Y. Maday, F. Rapetti, and B. I. Wohlmuth. The influence of quadrature formulas in 2D and 3D mortar element methods. In *Recent Developments in Domain Decomposition Methods*, pages 203–221. Springer, 2002.

[81] A. Massing, M. G. Larson, and A. Logg. Efficient implementation of finite element methods on nonmatching and overlapping meshes in three dimensions. *SIAM Journal on Scientific Computing*, 35(1):C23–C47, 2013.

[82] D. M. McQueen and C. S. Peskin. A three-dimensional computational method for blood flow in the heart. II. contractile fibers. *Journal of Computational Physics*, 82(2):289–297, 1989.

[83] R. W. Ogden. *Non-linear elastic deformations*. Courier Corporation, 1997.

[84] Y. Okamoto, K. Fujiwara, and Y. Ishihara. Effectiveness of higher order time integration in time-domain finite-element analysis. *IEEE transactions on magnetics*, 46(8):3321–3324, 2010.

[85] C. S. Peskin. Flow patterns around heart valves: a numerical method. *Journal of computational physics*, 10(2):252–271, 1972.

[86] C. S. Peskin. The immersed boundary method. *Acta numerica*, 11:479–517, 2002.

[87] C. S. Peskin and D. M. McQueen. A three-dimensional computational method for blood flow in the heart. I. immersed elastic fibers in a viscous incompressible fluid. *Journal of Computational Physics*, 81(2):372–405, 1989.

[88] A. Posa, A. Lippolis, R. Verzicco, and E. Balaras. Large-eddy simulations in mixed-flow pumps using an immersed-boundary method. *Computers & Fluids*, 47(1):33–43, 2011.

[89] A. M. Roma, C. S. Peskin, and M. J. Berger. An adaptive version of the immersed boundary method. *Journal of Computational Physics*, 153(2):509–534, 1999.

[90] S. Roy, L. Heltai, and F. Costanzo. Benchmarking the immersed finite element method for fluid–structure interaction problems. *Computers & Mathematics with Applications*, 69(10):1167–1188, 2015.

[91] S. Roy, L. Heltai, C. Drapaca, and F. Costanzo. An immersed finite element method approach for brain Biomechanics. In *Mechanics of Biological Systems and Materials, Volume 5: Proceedings of the 2012 Annual Conference on Experimental and Applied Mechanics*, pages 79–86. Springer, 2013.

[92] R. Stenberg. Analysis of mixed finite elements methods for the Stokes problem: a unified approach. *Mathematics of computation*, 42(165):9–23, 1984.

[93] R. Temam. *Navier-Stokes equations: theory and numerical analysis*, volume 343. American Mathematical Soc., 2001.

[94] M. Wichrowski, P. Krzyżanowski, L. Heltai, and S. Stupkiewicz. Exploiting high-contrast Stokes preconditioners to efficiently solve incompressible fluid-structure interaction problems. *arXiv preprint arXiv:2305.08986*, 2023.

[95] Y. Wu and X.-C. Cai. A fully implicit domain decomposition based ALE framework for three–dimensional fluid–structure interaction with application in blood flow computation. *Journal of Computational Physics*, 258:524–537, 2014.

[96] J. Xu and L. Zikatanov. Some observations on Babuška and Brezzi theories. *Numerische Mathematik*, 94(1):195–202, 2003.

[97] L. Zhu and C. S. Peskin. Simulation of a flapping flexible filament in a flowing soap film by the immersed boundary method. *Journal of Computational Physics*, 179(2):452–468, 2002.

Part II

Model order reduction in support of the Virtual Element Method

Introduction

Numerical analysis of partial differential equations is a key tool for the study and the approximation of mathematical models arising, for instance, in engineering, physics and biological studies. Classical examples are fluid-structure interaction problems, discussed in the first part of this thesis, fracture and contact problems, blood flow and biological networks formation.

Finite element methods are used to solve a wide range of PDEs since they provide accurate results in efficient way, with solid theoretical foundations, also in terms of error quantification in *a priori* and *a posteriori* settings. Complex computational geometries are treated by making use of spatial decompositions made up, in two dimensions, of triangles and quadrilaterals on which polynomial spaces are defined.

These aspects are at the base of the simplicity of finite elements, but in recent years several studies have been focused on the possibility of extending the method so that meshes consisting of more general polygonal elements can be considered too. The use of polytopal meshes provides more flexibility: for instance, complicated geometries can be represented with a minimal number of elements, reducing the computational cost, and hanging nodes are allowed, so that mesh adaptivity operations are relatively easy to carry out.

A first kind of generalization of finite elements are the so-called Composite Finite Elements, originally introduced in [88, 87]. In this case, polygonal elements are constructed by agglomeration of classical shaped elements. Conforming extensions such as Polygonal FEM [115, 92, 108] and Extended FEM [79] were obtained by enriching standard polynomial spaces with special shape functions able to capture jumps and singularities.

Hybrid High–Order methods [75, 76, 77, 61] are designed to support general polytopal meshes and arbitrary polynomial accuracy. This is achieved using two ingredients: in each cell, a potential reconstruction is carried out and then a sta-

bilization term on each face is constructed accordingly to the high order provided by the reconstruction itself. This procedure relies on additional unknowns, which are then eliminated by static condensation.

Discontinuous Galerkin Methods [54, 63, 74, 91, 113, 62] are based on a lack of continuity between elements and extremely general meshes can be employed. The use of general geometries comes naturally since the exchange of information between elements is managed with numerical fluxes. Moreover, this construction is also independent of the degree of accuracy under consideration, so that both $h-$refinement and $p-$refinement are easy to be carried out. Let us also observe that, in this setting, polynomials are defined by making use of less degrees of freedom than in classical finite elements on quadrilaterals (or hexahedra in 3D). In addition, this family of methods is directly constructed in the physical framework, without the need of resorting to reference elements and affine maps.

Finally, Virtual Element Methods [6, 12, 17] have been recently introduced as evolution of the mimetic finite difference method [20, 94]. The computational domain can be decomposed in polytopes of a very general shape and each function is described via a set of degrees of freedom. The local virtual element space is defined considering the polynomials of degree $\leq k$ plus additional contributions provided by (nonpolynomial) smooth functions which are themselves solutions of a PDE inside the element. These local problems are not explicitly solved and the quantities of interests, such as stiffness and mass matrices, are computed using suitable projections onto polynomials which are then completed by a stabilization term dealing with the nonpolynomial part. Thus, the discretization space is employed for designing the method and carrying out its analysis, but it is never fully evaluated at numerical level. Virtual element algorithms appear robust: the optimality has been proved under mesh regularity assumptions, but also observed in practice in case of badly shaped meshes. Despite that, the method has also some limitations caused by the virtual approach: solutions can only be accessed in *nonconforming* way by means of projection onto discontinuous polynomial spaces and standard stabilization terms produce polluted results is some particular situations.

In Chapter 4, we review the main features of the standard virtual element formulation for second order elliptic problems. Then, in Chapter 5, we present a reduced basis technique we designed as support tool for the lowest order VEM [64]. This technique can be exploited for different scopes and applied with different degrees of accuracy and computational costs. In few words, since with this approach

we are able to compute rough approximations of the virtual basis functions, we can design stabilization terms which are robust when standard choices show poor performance. This happens, for instance, when anisotopic problems are considered. Moreover, with this technique, we can reconstruct a virtual element solution in all (or some) elements of the mesh to allow tasks like visualization and pointwise evaluations of conforming solutions, which are not natural when the usual post-processing technique based on polynomial projections is employed.

Chapter 4

The Virtual Element Method

Virtual element methods were introduced a decade ago by Beirão da Veiga, Brezzi and collaborators in their pioneering works [6, 12, 1] as natural evolution of mimetic finite difference schemes. During the design process, the nodal values characterizing finite difference were replaced by degrees of freedom able to describe trial and test functions as usual in Galerkin methods. For this reason, VEMs are commonly considered as a generalization of the finite element method.

The virtual elements space, which can be constructed on polygonal/polyhedral meshes even with very general shapes, is defined on each element of the decomposition using polynomials of degree $\leq k$, which are then enriched by additional smooth functions so that unisolvence is ensured. These additional functions are themselves solutions of suitable local PDE problems: at the beginning, the common choice was a simple Laplace equation, but with the evolution of the methodology, more complicated choices have been done, usually connected with the problem under consideration. Anyway, these local problems are never required to be solved. Indeed, if the space is properly defined, the bilinear form describing the considered problem is exactly computed directly in terms of the degrees of freedom when at least one of the entries is a polynomial of degree $\leq k$. Thanks to this fact, suitable projections onto polynomials are used to decompose the virtual space into the direct sum of a polynomial space plus a residual space. In this way, the polynomial contribution is exactly computed using only the information provided by the degrees of freedom, while the nonpolynomial residual is dealt by means of *stabilization terms*, which are just required to mimic the behavior of the bilinear form. This stabilization term is basically arbitrarily chosen and influences stability and conditioning of the method, as observed in [30, 55].

The adjective *virtual* reflects the idea at the base of the discretization process: indeed, the discrete space provides tools for constructing the method and its theoretical foundations, but it is never explicitly constructed at numerical level. The method seeks for the solution in a conforming space, but using a non conforming Galerkin approach based on approximate bilinear forms.

Since the resulting algorithms are flexible and guarantee optimal convergence of the method, also when very badly shaped meshes are considered, virtual elements gained particular attention by the scientific community and are being studied under several aspects.

The development of mixed formulations is contextual to the birth of the method itself. We mention the first work by Brezzi, Falk and Marini regarding the basic principles of mixed virtual elements [52] and their application to general second order elliptic problems [13]. Dassi and Vacca discussed the construction of projectors and differential operators for high order mixed VEM [70], while the case of curved edges in 2D has been addressed in [65].

Stability and *a priori* estimates for classical formulations have been studied by Beirão da Veiga, Lovadina and Russo in [21] and by Chen and Huang [57] contextually with Brenner, Guan and Sung [48]. Beirão da Veiga, Mascotto and Meng recently focused on interpolation and stability properties for face and edge virtual element spaces [26, 24], while serendipity elements have been introduced by Beirão da Veiga, Brezzi, Marini and Russo in 2016 [14] and studied in subsequent works [16, 10, 124, 25].

The *a posteriori* error analysis has been conducted by Cangiani, Georgoulis, Pryer and Sutton in 2017 [55] and, two years later, by Beirão da Veiga, Manzini and Mascotto [23]. In case of mixed formulations, we have the work by Cangiani and Munar [56], whereas gradient recovery techniques are discussed in [86, 58, 122].

A first *hp* formulation on quasiuniform meshes has been presented by Beirão da Veiga, Chernov and Mascotto [18] and following papers discussed, for instance, the exponential convergence in presence of corner singularities [19], *a posteriori* estimates and adaptivity [23].

Ayuso de Dios, Lipnikov and Manzini introduced a first nonconforming formulation in [72], followed, for instance, by the work on nonconforming harmonic virtual elements by Mascotto, Perugia and Pichler [100]. We also mention the work by Di Pietro, Ern and collaborators about Hybrid High–Order methods [75, 76, 77], which can be actually interpreted as nonconforming virtual elements.

The work of Bertoluzza, Pennacchio and Prada has been focused on curved edges formulations [40], interior error estimates [42] as well as weakly imposed Dirichlet conditions [43] and image-based domain approximations [38].

Finally, stabilization free formulations have been recently introduced and discussed by Berrone and collaborators [31, 32, 33].

Over the years, this family of methods has gained more and more attention from engineers and mathematicians interested in applications, therefore implementation techniques and computational libraries are widely discussed in literature. We refer to the seminal *hitchhiker's guide* [12], which has been recalled by Mascotto [98] to describe how to construct the method with different monomial bases. Then, we mention the *50 lines* Matlab code by Sutton [116], the Matlab package `mVEM` [127] and the recent work [89] focused on high order two dimensional VEM. We finally cite the extensible object-oriented `C++` library `Veamy` [109] and the popular `DUNE-VEM` library by Dedner and Hodson [73].

Before listing some of the applications in which virtual elements have been employed, we mention the paper [104], where VEM is analyzed with engineering perspective, describing its interplay with the standard finite element methods.

We start citing the paper published in 2013 by Beirão da Veiga, Brezzi and Marini about linear elasticity [11] and the work by Mora, Rivera and Rodriguez about the Steklov eigenvalue problem [106], published in 2015. These two works, beyond the application, contain important results in virtual elements theory.

Starting from [11], the studies regarding elasticity applications evolved during the last decade thanks to the contribution of several groups: for instance, we have papers focused on three dimensional problems [80], estimates for spectral analysis [105], nonconforming formulations [128] and hybridization of virtual elements [67].

Regarding eigenvalue problems, Mora, Rivera, Rodriguez and collaborators continued the analysis about the Steklov problem [107, 93], while problems originating from general elliptic equations have been tackled in conforming setting by Gardini and Vacca [82], as well as by Boffi, Gardini and Gastaldi [46, 47]; nonconforming methods have been considered by Manzini, Gardini and Vacca [81].

Space-time formulations [84, 85], as well as virtual elements for the Helmholtz equation [101, 102, 103] are under investigation by Perugia and collaborators.

Several formulations have been presented in the case of the Stokes equation. For instance, Manzini and Mazzia adapted the popular Scott–Vogelius element in virtual framework [96], while Beirão da Veiga, Lovadina and Vacca studied a

divergence free method [22]. A stream formulation has been introduced in [3] by Antonietti, Beirão da Veiga,, Mora and Verani. A theoretical study on the Stokes complex have been conducted in [27].

Moving to more applied problems, we start mentioning the simulation of fluid flow through porous media. Gatica, Sequeira and collaborators focused on the Brinkman problem, in both linear and nonlinear framework [53, 83]. The work of Berrone and collaborators regards two-phase flow of immiscible fluids in porous media and fracture networks [34, 35], while Darcy flows have been considered, for instance, in [118, 95, 121, 129, 66].

Wriggers, Reddy and collaborators applied virtual elements to contact and deformations problems [126, 123, 125, 60], whereas geophysical applications and discrete fracture networks are studied by Berrone and his group [29, 36, 35]. Applications to magnetostatic problems are discussed in [8, 7, 9].

Of course, the use of virtual elements to solve applied problems poses the issue of the computational cost required to solve the linear system arising from the discretization, especially when three dimensional problems are considered. This aspect has led research towards the design of preconditioners. For instance, we cite the works by Scacchi and collaborators about preconditioning of saddle point problems [68, 69, 71, 45, 44] and Maxwell equations [5], as well as the results by Bertoluzza, Pennacchio and Prada focused on preconditioners for general elliptic problems [39, 111, 41].

At the end of this obviously non-exhaustive list, we mention the recent book [2], the review article by Beirão da Veiga, Brezzi, Marini and Russo [17] and the survey about stabilization terms by Mascotto [99].

In this chapter, we present the standard virtual elements formulation for second order elliptic problems and it is structured as follows. The model problem is described in Section 4.1, while in Section 4.2, we summarize the main geometrical assumptions one should consider to obtain a working method. The construction of discrete space, bilinear form and righ hand side is discussed in Section 4.3 and Section 4.4 respectively. Finally, some basics theoretical estimates are discussed in Section 4.5.

4.1 Model problem

Let us consider as model problem a second order diffusion equation with homogeneous Dirichlet boundary conditions

$$\begin{aligned} -\operatorname{div}\left(\mathbb{K}\,\nabla u\right) &= f \quad \text{in } \Omega \\ u &= 0 \quad \text{on } \partial\Omega, \end{aligned} \tag{4.1}$$

where $\Omega \subset \mathbb{R}^2$ is an open, bounded and connected polygonal domain and $f \in \mathrm{L}^2(\Omega)$ the given right hand side. The diffusivity tensor $\mathbb{K} \in \mathbf{L}^\infty(\Omega)$ is a positive definite matrix that we assume, for simplicity, to be constant.

The variational counterpart of (4.1) is given by the following formulation.

Problem 4.1.1. *Find $u \in V = \mathrm{H}^1_0(\Omega)$ such that*

$$\int_\Omega \mathbb{K}\,\nabla u \cdot \nabla v \,\mathrm{d}\mathbf{x} = \int_\Omega f\, v \,\mathrm{d}\mathbf{x} \quad \forall v \in V. \tag{4.2}$$

This problem admits a unique solution thanks to the Lax–Milgram theorem [51]: indeed, introducing the bilinear form

$$\begin{aligned} &a : V \times V \longrightarrow \mathbb{R} \\ &a(u,v) = \int_\Omega \mathbb{K}\,\nabla u \cdot \nabla v \,\mathrm{d}\mathbf{x} \end{aligned} \tag{4.3}$$

and the linear functional

$$\begin{aligned} &F : V \longrightarrow \mathbb{R} \\ &F(v) = \int_\Omega f\, v \,\mathrm{d}\mathbf{x}, \end{aligned} \tag{4.4}$$

it is not difficult to see that the following properties are satisfied [78]:

- a is continuous, i.e. there exists a constant $M_a = \|\mathbb{K}\|_{\infty,\Omega}$ such that

$$a(u,v) \le M_a \,|u|_{1,\Omega}\,|v|_{1,\Omega} \quad \forall u, v \in V; \tag{4.5}$$

- a is coercive because, since $\mathbb{K}$ is positive definite, there exists a constant α such that

$$a(v,v) \ge \alpha\,|v|^2_{1,\Omega} \quad \forall v \in V; \tag{4.6}$$

- F is continuous because $\mathrm{L}^2(\Omega) \subset \mathrm{H}^{-1}(\Omega)$, i.e there exists a constant M_F such that

$$F(v) \le M_F\,|v|_{1,\Omega} \quad \forall v \in V. \tag{4.7}$$

4.2 Domain discretization

In order to numerically solve Problem 4.1.1, we will consider a family of decompositions for the domain Ω. From classical finite elements theory, we know that assumptions on the regularity of the chosen triangulation are necessary to guarantee the convergence of the method [41]. The most common assumptions for classical finite elements are the *shape regularity condition* and the *minimum angle condition*.

1. **Shape regularity condition.** If $\{\mathfrak{T}_h\}_h$ is a sequence of triangulations, there exists a real number $\gamma \in (0,1)$, independent of h, such that for any triangle T, the longest edge h_{T} and its inradius r_{T} satisfy
$$r_{\mathrm{T}} \geq \gamma\, h_{\mathrm{T}}.$$

2. **Minimum angle condition.** If $\{\mathfrak{T}_h\}_h$ is a sequence of triangulations, there exists an angle $\theta_0 > 0$, independent of h, such that for any triangle T, its minimal angle θ_{T} satisfies
$$\theta_{\mathrm{T}} \geq \theta_0.$$

Even if virtual element methods allow the use of more general meshes, some shape regularity conditions are still necessary. Since during the years and in dependence of the application several conditions have been introduced [1, 19, 48, 50, 28, 37], we recall the assumptions presented in the first paper about the method [6] with just some comments on possible extensions.

We remark that in [4, 114], the authors studied the behavior of the method when the assumptions are stressed or violated.

Let us then consider a family of decompositions $\{\mathcal{T}_h\}_h$ for Ω made up of a finite number of non-overlapping elements E.

Assumption 4.2.1 (Simple polygons)**.** *For every h, the decomposition is made up of non overlapping simple polygons.*

This means that each $E \in \mathcal{T}_h$ is a simply connected open set without self-intersections in the boundary.

For each element E, h_E denotes its diameter, which is the maximum distance between two points of E, that is
$$h_E = \sup_{\mathbf{x},\mathbf{y}\in E} |\mathbf{x}-\mathbf{y}|$$

and $\mathbf{x}_E$ denotes the centroid. As usual, the mesh size h corresponds to $\max_E h_E$. The notation ∂E is reserved for the boundary of E, which is supposed to be formed by N straight edges, denoted by e; by abuse of notation $e \in \partial E$ signifies that e is an edge of E. The vertices are denoted by $\mathbf{v}_n$, for $n =, 1 \dots, N$. Moreover, $|E|$ and $|e|$ denote the area of the element E and the length of the edge e respectively.

A second assumption, related to the shape of the elements, plays a crucial role in the analysis of polygonal methods.

Assumption 4.2.2 (Star-shapedness)**.** *There exists a real number $\gamma \in (0, 1)$, independent of h, such that each element E is star-shaped with respect to a ball of radius $r_E \geq \gamma\, h_E$.*

This statement can be weakened assuming that each element E is the union of a finite number of sub-cells $E_1, \dots, E_L$ each satisfying Assumption 4.2.2, with the requirement that E_i and E_{i+1} share a common edge, for $i = 1, \dots, L$.

Finally, another common assumption regards the length of the elemental edges. In [6], the statement imposes a condition on the maximum point-to-point distance between vertices.

Assumption 4.2.3. *There exists a real number $\gamma \in (0, 1)$, independent of h, such that for each element E, the distance $|\mathbf{v}_i - \mathbf{v}_j|$ between any two vertices $\mathbf{v}_i, \mathbf{v}_j$ of E satisfies*

$$|\mathbf{v}_i - \mathbf{v}_j| \geq \gamma\, h_E \qquad i, j = 1, \dots, N.$$

Since this formulation was too restrictive, Assumption 4.2.4 has been introduced [1, 21, 48] and is commonly used in VEM literature.

Assumption 4.2.4. *There exists a real number $\gamma \in (0, 1)$, independent from h, such that for each element E, the length of every edge e satisfies*

$$|e| \geq \gamma\, h_E.$$

Moreover, in [21, 50], the authors showed that the following Assumption 4.2.5 is implied by Assumption 4.2.4, but not equivalent.

Assumption 4.2.5. *There exists a positive integer, independent from h, such that the number of edges of every polygon $E \in \mathcal{T}_h$ is uniformly bounded.*

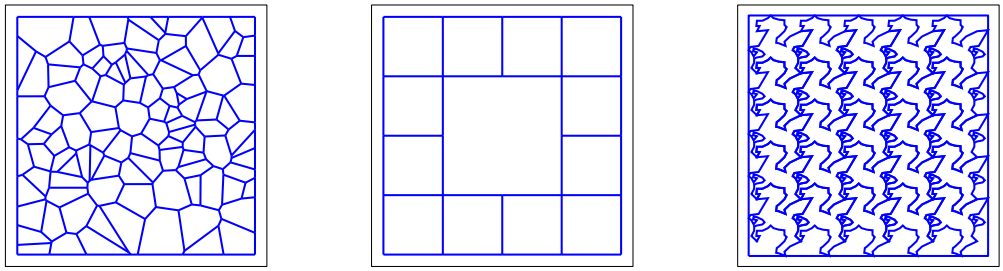


Figure 4.1: Three examples of admissible meshes for virtual element methods. From left to right: a Voronoi mesh of convex polygons, a quadrilateral mesh with hanging nodes (the square in the center is treated as an octagon), a mesh of non convex polygons [110].

It is important to notice that virtual element methods allow the presence of non convex polygons and hanging nodes in $\mathcal{T}_h$. Indeed, a hanging node is seen as a splitting of an edge into two edges at a π angle, increasing the number of edges of the considered element. This fact makes this method versatile for operations such as refinement and coarsening. Some examples of admissible meshes are collected in Figure 4.1.

4.3 A class of non-conforming discretizations

We now discuss how to discretize the problem under consideration. Given a polygonal decomposition $\mathcal{T}_h$ of Ω satisfying the geometrical assumptions, we rewrite the bilinear form a and the H^1 semi-norm as the sum of all the local contributions of $E \in \mathcal{T}_h$:

$$a(u,v) = \sum_{E\in\mathcal{T}_h} a^E(u,v) \quad \forall u,v \in V \quad \text{and} \quad |v|_{1,\Omega} = \sqrt{\sum_{E\in\mathcal{T}_h} |v|^2_{1,E}} \quad \forall v \in V. \tag{4.8}$$

Before presenting the discrete version of Problem 4.1.1 in virtual elements framework, we need to introduce some other ingredients. Let us consider a finite dimensional subspace V_h of V and introduce the discrete bilinear form $a_h : V_h \times V_h \longrightarrow \mathbb{R}$

$$a_h(u_h,v_h) = \sum_{E\in\mathcal{T}_h} a_h^E(u_h,v_h) \quad \forall u_h, v_h \in V_h \tag{4.9}$$

where each term a_h^E, approximating a^E, is itself a bilinear form defined locally on the element E. We denote by $V_h(E)$ the restriction of V_h to E. Letting f_h be an approximation of f, the discrete problem can be stated as follows.

Problem 4.3.1. *Given $f_h \in V_h'$, find $u_h \in V_h$ such that*

$$a_h(u_h, v_h) = (f_h, v_h) \qquad \forall v_h \in V_h. \tag{4.10}$$

In order to ensure the well-posedness of Problem 4.3.1, we introduce two hypotheses regarding the discrete bilinear form a_h.

Hypothesis 4.3.1 (k–consistency)**.** *There exists an integer $k \geq 1$ such that for all h and for all E in $\mathcal{T}_h$, we have that $\mathcal{P}_k(E) \subset V_h(E)$ and*

$$a_h^E(q, v_h) = a^E(q, v_h) \qquad \forall q \in \mathcal{P}_k(E),\, \forall v_h \in V_h(E). \tag{4.11}$$

Hypothesis 4.3.2 (Stability)**.** *There exist two constant $\alpha^\star, \alpha_\star > 0$, independent both from h and E, such that*

$$\alpha_\star\, a^E(v_h, v_h) \leq a_h^E(v_h, v_h) \leq \alpha^\star\, a^E(v_h, v_h) \qquad \forall v_h \in V_h(E). \tag{4.12}$$

We emphasize that Hypothesis 4.3.1 means that the discrete bilinear form is computed exactly when one of the entries is a polynomial of degree $\leq k$. Furthermore, if a_h^E is symmetric, from Hypothesis 4.3.2 and the definition of a^E, we can easily obtain the continuity

$$\begin{aligned} a_h^E(u_h, v_h) &\leq \big(a_h^E(u_h, u_h)\big)^{1/2} \big(a_h^E(v_h, v_h)\big)^{1/2} \\ &\leq \alpha^\star \big(a^E(u_h, u_h)\big)^{1/2} \big(a^E(v_h, v_h)\big)^{1/2} \\ &= \alpha^\star\, M_a\ |u_h|_{1,E}\, |v_h|_{1,E} \qquad \forall u_h, v_h \in V_h(E). \end{aligned} \tag{4.13}$$

We introduce the broken space

$$\mathrm{H}^1(\mathcal{T}_h) := \prod_{E \in \mathcal{T}_h} \mathrm{H}^1(E)$$

endowed with the semi-norm

$$|v|_{1,h,\Omega} := \left(\sum_{E \in \mathcal{T}_h} |\nabla v|_{0,E}^2 \right)^{1/2}.$$

Notice that, when dealing with discontinuous functions, this object is not a norm since, for instance, we have $|v_h|_{1,h,\Omega} = 0$ for every piecewise constant function v_h.

Theorem 4.3.1 ([6]). *Under Hypotheses 4.3.1 and 4.3.2, Problem 4.3.1 admits a unique solution u_h. Moreover, for every approximation $u_I \in V_h$ of u and for every approximation u_π piecewise in $\mathcal{P}_k$, the following estimate holds true*

$$|u-u_h|_{1,\Omega} \leq C\left(|u-u_I|_{1,\Omega} + |u-u_\pi|_{1,h,\Omega} + \sup_{v\in \mathrm{H}^1_0(\Omega)} \frac{|(f,v)-(f_h,v)|}{|v|_{1,\Omega}}\right) \tag{4.14}$$

where C is a constants depending only on $\mathbb{K}$, $\alpha_\star$ and $\alpha^\star$.

Proof. The existence and the uniqueness of the solution for Problem 4.3.1 are direct consequences of the Lax–Milgram Theorem.

Now, let us define $\delta_h = u_h - u_I$. Exploiting coercivity and consistency of the bilinear form, we can write

$$\alpha\,\alpha_\star |\delta_h|^2_{1,\Omega} \leq \alpha_\star a(\delta_h,\delta_h) \leq a_h(\delta_h,\delta_h)$$

so that, using linearity, Problem 4.3.1 and again consistency, we have

$$\begin{aligned}
\alpha\,\alpha_\star |\delta_h|^2_{1,\Omega} &\leq a_h(u_h,\delta_h) - a_h(u_I,\delta_h)\\
&= (f_h,\delta_h) - \sum_{E\in\mathcal{T}_h} a_h^E(u_I,\delta_h)\\
&= (f_h,\delta_h) - \sum_{E\in\mathcal{T}_h} \left(a_h^E(u_I-u_\pi,\delta_h) + a_h^E(u_\pi,\delta_h)\right)\\
&= (f_h,\delta_h) - \sum_{E\in\mathcal{T}_h} \left(a_h^E(u_I-u_\pi,\delta_h) + a^E(u_\pi,\delta_h)\right)\\
&= (f_h,\delta_h) - \sum_{E\in\mathcal{T}_h} \left(a_h^E(u_I-u_\pi,\delta_h) + a^E(u_\pi-u,\delta_h)\right) - a(u,\delta_h)\\
&= (f_h,\delta_h) - \sum_{E\in\mathcal{T}_h} \left(a_h^E(u_I-u_\pi,\delta_h) + a^E(u_\pi-u,\delta_h)\right) - (f,\delta_h)\\
&= (f_h,\delta_h) - (f,\delta_h) - \sum_{E\in\mathcal{T}_h} \left(a_h^E(u_I-u_\pi,\delta_h) + a^E(u_\pi-u,\delta_h)\right).
\end{aligned} \tag{4.15}$$

By continuity of a^E and a_h^E, we have the following local bounds

$$\begin{aligned}
a^E(u_\pi-u,\delta_h) &\leq M_a\ |u_\pi-u|_{1,E}\,|\delta_h|_{1,E}\,,\\
a_h^E(u_I-u_\pi,\delta_h) &\leq M_a\,\alpha^\star\ |u_I-u_\pi|_{1,E}\,|\delta_h|_{1,E}\,.
\end{aligned} \tag{4.16}$$

Putting everything together and making use of the broken norm, we get

$$|\delta_h|^2_{1,\Omega} \leq C\left(\sup_{v\in \mathrm{H}^1_0(\Omega)} \frac{|(f,v)-(f_h,v)|}{|v|_{1,\Omega}} + |u_I-u_\pi|_{1,\Omega,h} + |u-u_\pi|_{1,\Omega,h}\right)|\delta_h|_{1,\Omega}, \tag{4.17}$$

where the constant C depends only on $\mathbb{K}$, $\alpha_\star$ and $\alpha^\star$. Therefore, applying the triangle inequality, we obtain the final estimate (4.14).

□

4.4 The Virtual Element space

Let us consider a generic polygonal element E. For a fixed order of accuracy $k \geq 1$, we introduce the space of continuous functions over ∂E which are polynomials of degree less than or equal to k on each edge

$$\mathbb{B}_k(\partial E) = \{v \in C^0(\partial E) : v_{|e} \in \mathcal{P}_k(e) \quad \forall e \in \partial E\}. \tag{4.18}$$

The local VEM space is defined starting from $\mathbb{B}_k(\partial E)$ considering an additional property to describe functions in the interior of E

$$V_k(E) = \{v \in \mathrm{H}^1(E) : v_{|e} \in \mathbb{B}_k(\partial E),\, \Delta v_{|E} \in \mathcal{P}_{k-2}(E)\}, \tag{4.19}$$

in particular, we are using the convention that $\mathcal{P}_{-1} = \{0\}$.

It is not difficult to see that the dimension of the local VEM space defined in (4.19) is given by

$$N_k^E = Nk + \frac{k(k-2)}{2}. \tag{4.20}$$

Therefore, we define the degrees of freedom (DOFs) as follows.

Definition 4.4.1 (Local degrees of freedom)**.** *The degrees of freedom for $v_h \in V_k(E)$ are given by:*

- *the values of v_h at the vertices,*
- *for $k > 1$, the values of v_h at $k-1$ points on each edge $e \in \partial E$,*
- *for $k > 1$, the internal moments*

$$\frac{1}{|E|}\int_E m(\mathbf{x})\, v_h(\mathbf{x})\, \mathrm{d}\mathbf{x} \quad \forall m \in \mathfrak{M}_{k-2}(E). \tag{4.21}$$

For a generic $E \in \mathbb{R}^2$ and given a multi-index $\boldsymbol{\beta}$, we let $m_{\boldsymbol{\beta}}$ denote the scaled monomial of degree $|\boldsymbol{\beta}|$ defined as

$$m_{\boldsymbol{\beta}} = \left(\frac{\mathbf{x} - \mathbf{x}_E}{h_E}\right)^{\boldsymbol{\beta}}, \tag{4.22}$$

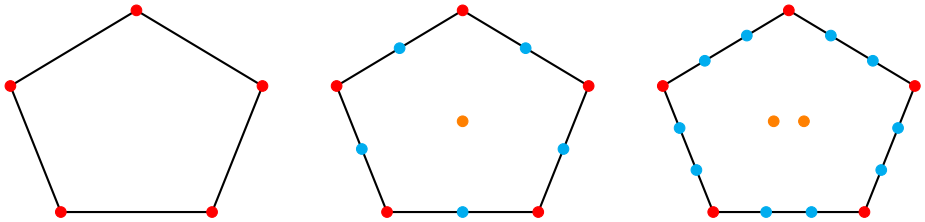


Figure 4.2: Schematic representation of the degrees of freedom of a pentagon for $k = 1, 2, 3$. The DOFs at the vertices are marked in red, the DOFs on the edges in cyan and the internal moments in orange.

and we denote by $\mathfrak{M}_k(E)$ the set of scaled monomials of degree less or equal than k

$$\mathfrak{M}_k(E) = \{m_{\boldsymbol{\beta}} : 0 \leq |\boldsymbol{\beta}| \leq k\}. \tag{4.23}$$

A schematic representation of the degrees of freedom is reported in Figure 4.2.

We have the following theorem.

Theorem 4.4.1 ([6])**.** *Let us consider a simple polygon E with N edges and the space $V_k(E)$ defined in* (4.19)*. The set of degrees of freedom presented in Definition 4.4.1 is unisolvent for $V_k(E)$.*

Remark 4.4.1. *Regarding the degrees of freedom on each edge, several choices are possible. For instance, they can be $k-1$ uniformly spaced points [6] or Gauss–Lobatto points [12]. Also the scaled moments up to degree $k-2$*

$$\frac{1}{|e|}\int_e m\, v_h \,\mathrm{da} \quad \forall m \in \mathfrak{M}_{k-2}(e)$$

are an admissible choice [1]. In general, we can consider any set of parameters that, together with the values at the vertices, uniquely identifies a polynomial of degree k on each edge.

Remark 4.4.2. *As observed by Mascotto [98], the definition of the space $V_k(E)$ is completely independent of the choice of basis for the polynomial space. Indeed, we introduced the scaled monomial just as a mathematical tool for defining the internal degrees of freedom. Consequently, any polynomial basis can be employed for this scope. Let us remark that, although the monomial basis is the most popular*

choice, it has bad effects on the condition number of the stiffness matrix for high polynomial degrees and in presence of extremely badly shaped polygons. In order to mitigate this negative behavior, the monomial basis can be replaced by an $\mathrm{L}^2(E)$ *orthonormal basis for* $\mathcal{P}_k(E)$.

From the mathematical point of view, a degree of freedom is defined as a linear functional [41]. We will use this notion at the end of this chapter.

Definition 4.4.2. *Given* E *in* $\mathcal{T}_h$*, we define the family of local operators*

$$DOF_i : V_k(E) \longrightarrow \mathbb{R} \quad \text{for } i = 1, \dots, N_k^E \tag{4.24}$$

such that

$$DOF_i(v_h) = \text{the } i^{th} \text{ degree of freedom of } v_h \tag{4.25}$$

for all $v_h \in V_k(E)$.

At this point, we are able to construct the global virtual element space: glueing together by continuity all the local spaces, we have

$$V_h^k = \{v \in V : v_{|\partial E} \in \mathbb{B}_k(\partial E) \text{ and } \Delta v_{|E} \in \mathcal{P}_{k-2}(E) \quad \forall E \in \mathcal{T}_h\}. \tag{4.26}$$

In this case, to determine the dimension of the space, we take into account that the degrees of freedom on $\partial\Omega$ do not play a role due to the Dirichlet boundary conditions, therefore we have

$$N_h = dim(V_h^k) = N_V + N_e(k-1) + N_{\mathcal{T}}\frac{k(k-1)}{2} \tag{4.27}$$

where N_V is the number of internal vertices, N_e the number of internal edges and $N_{\mathcal{T}}$ the number of elements of $\mathcal{T}_h$. Consequently, the choice of the degrees of freedom for $v_h \in V_h^k$ is naturally given by

- the values of v_h at the internal vertices,
- for $k > 1$, the values of v_h at $k-1$ points on each internal edge e,
- for $k > 1$, the moments

$$\frac{1}{|E|}\int_E m(\mathbf{x})v_h(\mathbf{x})\,\mathrm{d}\mathbf{x} \quad \forall m \in \mathfrak{M}_{k-2}(E) \quad \text{in each element } E; \tag{4.28}$$

which extends the unisolvency proved locally in Theorem 4.4.1.

Once the discrete space is defined, we can discuss how the discrete bilinear form a_h and the right hand side are constructed in practice: to this aim, we introduce the notion of *computability*. In virtual element methods, a quantity is said to be *computable* if it can be determined directly from information provided by the degrees of freedom.

4.4.1 The discrete bilinear form a_h

Exploiting the degrees of freedom, we can compute exactly $a^E(q, v)$ for any polynomial $q \in \mathcal{P}_k(E)$ and $v_h \in V_k(E)$, indeed, integrating by parts, we have

$$a^E(q, v_h) = \int_E \mathbb{K}\nabla q \cdot \nabla v_h \, \mathrm{d}\mathbf{x} = -\int_E \mathrm{div}(\mathbb{K}\nabla q)\, v_h \, \mathrm{d}\mathbf{x} + \int_{\partial E} (\mathbb{K}\nabla q \cdot \mathbf{n}) v_h \, \mathrm{da} \quad (4.29)$$

where $\mathbf{n}$ denotes the outer normal of ∂E. In particular, notice that $\mathrm{div}(\mathbb{K}\nabla q) \in \mathcal{P}_{k-2}(E)$, so that the last two integrals can be computed exactly without knowing the behavior of v_h inside the polygon. We emphasize that, if u_h, v_h are not polynomials, then $a^E(u_h, v_h)$ is not computable: we replace it with a computable approximate bilinear form.

An ingredient we need to successfully build a computable bilinear form is a projection operator from the local VEM space to the polynomials of degree up to k. The definition reads as follows.

Definition 4.4.3. *The projection operator $\Pi_k^\nabla : V_k(E) \to \mathcal{P}_k(E) \subset V_k(E)$ is solution of the problem*

$$\begin{aligned} \int_E \nabla \Pi_k^\nabla v_h \cdot \nabla q \, \mathrm{d}\mathbf{x} &= \int_E \nabla v_h \cdot \nabla q \, \mathrm{d}\mathbf{x} \quad \forall q \in \mathcal{P}_k(E) \\ \int_{\partial E} \Pi_k^\nabla v_h \, \mathrm{da} &= \int_{\partial E} v_h \, \mathrm{da} \end{aligned} \quad (4.30)$$

for all $v_h \in V_k(E)$.

A fundamental property of the projector Π_k^∇ is that it preserves polynomials, indeed

$$\Pi_k^\nabla q = q \quad \forall q \in \mathcal{P}_k(E). \quad (4.31)$$

Through Π_k^∇, we obtain a computable decomposition of $V_k(E)$ as

$$V_k(E) = \mathcal{P}_k(E) \oplus V_\perp(E) \quad (4.32)$$

where $V_\perp(E)$ denotes the kernel of Π_k^∇, which is orthogonal to the polynomial space $\mathcal{P}_k(E)$ with respect to the scalar product underlying (4.30). As a consequence, since we can write

$$v_h = \Pi_k^\nabla v_h + (v_h - \Pi_k^\nabla v_h) \quad \forall v_h \in V_k(E), \tag{4.33}$$

the continuous bilinear form is decomposed as follows

$$\begin{aligned} a^E(u_h, v_h) = a^E(\Pi_k^\nabla u_h, \Pi_k^\nabla v_h) + a^E(\Pi_k^\nabla u_h, (\mathrm{I} - \Pi_k^\nabla)v_h) \\ + a^E((\mathrm{I} - \Pi_k^\nabla)u_h, \Pi_k^\nabla v_h) + a^E((\mathrm{I} - \Pi_k^\nabla)u_h, (\mathrm{I} - \Pi_k^\nabla)v_h) \end{aligned} \tag{4.34}$$

We remark that the first three terms at the right hand side are directly computable by means of the degrees of freedom since Π_k^∇ is computable. On the other hand, the term $a^E((\mathrm{I} - \Pi_k^\nabla)u_h, (\mathrm{I} - \Pi_k^\nabla)v_h)$ is not computable.

If we did not consider the last term and chose

$$a_h^E(u_h, v_h) = a^E(\Pi_k^\nabla u_h, \Pi_k^\nabla v_h) + a^E(\Pi_k^\nabla u_h, (\mathrm{I} - \Pi_k^\nabla)v_h) + a^E((\mathrm{I} - \Pi_k^\nabla)u_h, \Pi_k^\nabla v_h),$$

we would obtain a k–consistent discrete bilinear form that is not stable in general. For this reason, we need to replace the fourth term in (4.34) with a computable term able to mimic its behavior and endowed with appropriate properties in such a way that the resulting bilinear form satisfies both Hypotheses 4.3.1 and 4.3.2. This new term is referred to as *stabilization*.

Let us consider any symmetric positive definite bilinear form S^E that scales like a^E in $V_\perp(E)$, i.e. satisfying

$$c_\star\, a^E(w_\perp, w_\perp) \le S^E(w_\perp, w_\perp) \le c^\star\, a^E(w_\perp, w_\perp) \quad \forall w_\perp \in V_\perp(E) \tag{4.35}$$

for some $c_\star, c^\star > 0$ independent of E and of its size h_E. The local discrete a_h^E can be defined as

$$\begin{aligned} a_h^E(u_h, v_h) = a^E(\Pi_k^\nabla u_h, \Pi_k^\nabla v_h) + a^E(\Pi_k^\nabla u_h, (\mathrm{I} - \Pi_k^\nabla)v_h) \\ + a^E((\mathrm{I} - \Pi_k^\nabla)u_h, \Pi_k^\nabla v_h) + S^E((\mathrm{I} - \Pi_k^\nabla)u_h, (\mathrm{I} - \Pi_k^\nabla)v_h) \end{aligned} \tag{4.36}$$

Theorem 4.4.2 ([6]). *The bilinear form defined in* (4.36) *is consistent and stable as required by Hypotheses 4.3.1 and 4.3.2.*

Remark 4.4.3. *In Section 4.3, we proved continuity of the generic a_h^E assuming its symmetry. On the other hand, considering the definition in* (4.36)*, continuity*

can be proved just assuming that a_h^E is symmetric in $V_\perp(E)$. Indeed, the first three terms at the right hand side of (4.36) *are continuous since a^E is continuous, whereas the virtual term $a^E((\mathrm{I}-\Pi_k^\nabla)u_h,(\mathrm{I}-\Pi_k^\nabla)v_h)$ is replaced by the stabilization S^E, which is symmetric by construction.*

Corollary 4.4.1. *If a_h^E is defined as in* (4.36)*, then Problem 4.3.1 admits a unique solution.*

The choice of stabilization term strongly depends both on the problem under consideration and on the definition of discrete space. We recall some popular choices which have been introduced for the Poisson equation. Their application can be then extended to more general problems, as done, for instance, in [12, 15, 32].

The first stabilization term we mention is usually called *dofi–dofi* [6] since is locally defined as

$$S^E_{dofi}(v_h,w_h)=\sum_{i=1}^{N_k^E} DOF_i(v_h)\,DOF_i(w_h)\quad\forall v_h,w_h\in V_\perp(E). \tag{4.37}$$

In [21], the authors presents H^1 error estimates studying the effects of this choice on the solution.

Another possible choice, more robust than *dofi–dofi*, is called *D–recipe* [98] and is defined as

$$S^E_{D-rec}(v_h,w_h)=\sum_{i=1}^{N_k^E}\omega_i\,DOF_i(v_h)\,DOF_i(w_h)\quad\forall u_h,v_h\in V_\perp(E), \tag{4.38}$$

where the weight ω_i is defined through the bilinear form as

$$\omega_i=\max\{1,a^E(\Pi_k^\nabla(\varphi_i),\Pi_k^\nabla(\varphi_i))\}.$$

In [21], the authors present a stabilization term which is given by summing two contributions

$$S^E_{\circ,\partial}(v_h,w_h)=S^E_\partial(v_h,w_h)+S^E_\circ(v_h,w_h)\quad\forall u_h,v_h\in V_\perp(E). \tag{4.39}$$

The first term S^E_∂ involves only the boundary degrees of freedom and can be defined in two different ways: the classical choice is the *dofi–dofi* restricted to the boundary, whereas the second option [126] is given by

$$S^E_\partial(v_h,w_h)=h_E\int_{\partial E}\partial_\tau v_h\,\partial_\tau w_h\,\mathrm{da}$$

where ∂_τ denotes the tangential derivative along ∂E. The internal contribution $S_\circ^E$ is usually set to be the *dofi–dofi* restricted to the internal degrees of freedom. Notice that, when both contributions are defined through the *dofi–dofi* stabilization, we obtain S_{dofi}^E.

Remark 4.4.4. *In general, the stiffness matrix K_h constructed from a_h^E is not close to the exact stiffness matrix K, which we would be able to assemble if we knew the basis functions of $V_k(E)$ [12]. Indeed, K_h is not an approximation of K in standard sense due to the stabilization term. Standard choices of S^E are only able to mimic the behavior of the nonpolynomial term $a^E((\mathrm{I}-\Pi_k^\nabla)u_h,(\mathrm{I}-\Pi_k^\nabla)v_h)$, without producing an actual approximation. A practical example showing this feature can be found in the next Chapter (see Section 5.5.1).*

4.4.2 The right hand side

Also the linear operator at the right hand side is not computable in general, therefore for its construction we introduce the local L^2 projection onto polynomials.

Definition 4.4.4. *The projection operator $\Pi_k^0 : V_k(E) \to \mathcal{P}_k(E) \subset V_k(E)$ is solution of the problem*

$$\int_E \Pi_k^0 v_h \, q \,\mathrm{d}\mathbf{x} = \int_E v_h \, q \,\mathrm{d}\mathbf{x} \quad \forall q \in \mathcal{P}_k(E) \tag{4.40}$$

for all $v_h \in V_k(E)$.

When dealing with $V_k(E)$ for $k \geq 2$, we project f onto the polynomial space $\mathcal{P}_{k-2}(E)$, whereas for $k = 1$, the projection is done onto the space $\mathcal{P}_0(E)$ of constants. Hence, for $k \geq 2$ we define f_h as

$$f_h = \Pi_{k-2}^0 f \tag{4.41}$$

on each element $E \in \mathcal{T}_h$. Notice that Π_{k-2}^0 is exactly computable through the internal moments.

For $k = 1$, f is approximated by piecewise constants and v_h is replaced by its average over ∂E, therefore

$$(f_h, v_h) = \sum_{E\in\mathcal{T}_h} \int_E \Pi_0^0 f \left(|\partial E|^{-1} \int_{\partial E} v_h \,\mathrm{da} \right) \mathrm{d}\mathbf{x} = \sum_{E\in\mathcal{T}_h} \Pi_0^0 f \frac{|E|}{|\partial E|} \int_{\partial E} v_h \,\mathrm{da}. \tag{4.42}$$

On the other hand, for $k \geq 2$, we have

$$(f_h, v_h) = \sum_{E \in \mathcal{T}_h} \int_E f_h \, v_h \, \mathrm{d}\mathbf{x} = \sum_{E \in \mathcal{T}_h} \int_E (\Pi^0_{k-2} f) \, v_h \, \mathrm{d}\mathbf{x} = \sum_{E \in \mathcal{T}_h} \int_E f \, (\Pi^0_{k-2} v_h) \, \mathrm{d}\mathbf{x} \tag{4.43}$$

which can be exactly computed as Π^0_{k-2} is itself computable.

Approximation estimates for the consistency term $|(f, v_h) - (f_h, v_h)|$, discussed for instance in [6] and [11], are recalled in the next section.

Remark 4.4.5. *In [1], the authors present an alternative way of defining the discrete bilinear form by replacing Π^∇_k with the L^2–projection Π^0_{k-1}, which is also computable. In this case, a^E_h is defined as*

$$\begin{aligned} a^E_h(u_h, v_h) = &\int_E \mathbb{K} \Pi^0_{k-1}(\nabla u_h) \cdot \Pi^0_{k-1}(\nabla v_h) \, \mathrm{d}\mathbf{x} \\ &+ \int_E \mathbb{K} \Pi^0_{k-1}(\nabla u_h) \cdot (\mathrm{I} - \Pi^0_{k-1})(\nabla v_h) \, \mathrm{d}\mathbf{x} \\ &+ \int_E \mathbb{K} (\mathrm{I} - \Pi^0_{k-1})(\nabla u_h) \cdot \Pi^0_{k-1}(\nabla v_h) \, \mathrm{d}\mathbf{x} \\ &+ S^E((\mathrm{I} - \Pi^\nabla_k) u_h, (\mathrm{I} - \Pi^\nabla_k) v_h) \end{aligned}$$

Notice that, for $k = 1$, the new definition is equivalent to (4.36), whereas this is not true for $k \geq 2$. This alternative definition is often preferred for $k \geq 3$ in presence of variable coefficients: in this situation, the Π^∇_k operator produces a loss of order of convergence [15].

4.5 Some estimates

In this section, we report local estimates for the projection error, the interpolation error and the approximation error for the right hand side.

Projection error

Under shape regularity assumptions, classical results for finite elements [32] can be extended to virtual elements. This is the case of the following estimate regarding local projection onto polynomials.

Proposition 4.5.1. *Let us consider a simple polygon E satisfying the geometrical Assumption 4.2.2. There exists a constant C, depending on γ and k, such that for every s, $1 \leq s \leq k+1$, and for every $w \in \mathrm{H}^s(E)$ there exists a $w_\pi \in \mathcal{P}_k(E)$ such that*

$$\|w - w_\pi\|_{0,E} + h_E|w - w_\pi|_{1,E} \leq Ch_E^s|w|_{s,E}. \tag{4.44}$$

Interpolation error

Recalling Definition 4.4.2, we can construct the canonical basis $\{\varphi_1, \ldots, \varphi_{N_k^E}\}$ of $V_k(E)$ satisfying

$$DOF_i(\varphi_j) = \delta_{ij} \quad i, j = 1, \ldots, N_k^E, \tag{4.45}$$

where δ_{ij} is the Kronecker delta. We can then express a discrete function $v_h \in V_k(E)$ using an interpolation identity in Lagrangian setting, that is

$$v_h = \sum_{i=1}^{N_k^E} DOF_i(v_h)\, \varphi_i. \tag{4.46}$$

For every smooth enough w there exists a unique element $w_I \in V_k(E)$ such that

$$DOF_i(w - w_I) = 0 \quad i = 1, \ldots, N_k^E. \tag{4.47}$$

The following interpolation error estimate has been stated in [6] and then formally proved in [57]. A version for less regular functions has been discussed in [106] by extending the Scott–Dupont theory.

Proposition 4.5.2. *Let us consider a mesh $\mathcal{T}_h$ satisfying the geometrical Assumption 4.2.2. There exists a constant C, depending on γ and k, such that for every s, $2 \leq s \leq k+1$, for every h, for all $E \in \mathcal{T}_h$ and for every $w \in \mathrm{H}^s(E)$ there exists a $w_I \in V_k(E)$ such that*

$$\|w - w_I\|_{0,E} + h_E|w - w_I|_{1,E} \leq Ch_E^s|w|_{s,E}. \tag{4.48}$$

Approximation estimates for the right-hand side

We now recall error estimates for the right hand side [6]. As we did in Section 4.4.2, we distinguish the case $k = 1$ from $k \geq 2$.

Proposition 4.5.3. *Given $f \in \mathrm{L}^2(\Omega)$, the following estimates hold true. For $k = 1$, there exists a positive constant C such that*

$$\sup_{v \in \mathrm{H}^1_0(\Omega)} \frac{|(f, v) - (f_h, v_h)|}{|v|_{1,\Omega}} \leq C\, h\, |f|_{1,\Omega} \tag{4.49}$$

while, for $k \geq 2$, there exists a positive constant C such that

$$\sup_{v \in \mathrm{H}^1_0(\Omega)} \frac{|(f, v) - (f_h, v_h)|}{|v|_{1,\Omega}} \leq C\, h^k\, |f|_{k-1,\Omega} \tag{4.50}$$

VEM approximation error

Combining all the estimates presented in this section with the abstract result of Theorem 4.3.1, we can easily deduce the approximation error estimate for the global virtual element space V_h^k [15].

Theorem 4.5.1. *Let $u \in \mathrm{H}^{k+1}(\Omega)$ be the solution of Problem 4.1.1 and let $u_h \in V_h^k$ the solution of Problem 4.3.1. Under mesh regularity assumptions, the following estimate holds true*

$$|u - u_h|_{1,\Omega} \leq C h^k\, |u|_{k+1,\Omega} \tag{4.51}$$

Proof. From the general Theorem 4.3.1, we have the following estimate

$$|u - u_h|_{1,\Omega} \leq C \left(|u - u_I|_{1,\Omega} + |u - u_\pi|_{1,h,\Omega} + \sup_{v \in \mathrm{H}^1_0(\Omega)} \frac{|(f, v) - (f_h, v_h)|}{|v|_{1,\Omega}} \right) \tag{4.52}$$

where u_I denotes any interpolant of u in V_h^k and u_π is the L^2 projection of u onto piecewise polynomials of degree up to k. In particular, thanks to the estimates summarized in this section, we have

$$|u - u_I| \leq C h^k\, |u|_{k+1,\Omega}$$
$$|u - u_\pi|_{1,h,\Omega} \leq C h^k\, |u|_{k+1,\Omega}$$

and

$$\sup_{v \in \mathrm{H}^1_0(\Omega)} \frac{|(f, v) - (f_h, v_h)|}{|v|_{1,\Omega}} \leq C\, h^k\, |f|_{cojns1,\Omega}$$

so that, putting everything together, the final estimate is easily derived. □

Chapter 5

Reduced Basis approach to the Virtual Element Method

In the previous chapter, we recalled the main features of the standard virtual element formulation for second order elliptic problems. In particular, we discussed the construction of the method recalling basic theoretical results.

Clearly, beyond all the appealing features, the method has also some limitations directly descending from the construction of the approach itself. Indeed, we saw that virtual element methods are constructed on conforming spaces which are actually treated in a nonconforming way: each local space is split into direct sum of a polynomial space plus a space consisting of smooth functions which are themselves solution of PDEs. Therefore, bilinear forms are approximated via polynomial projections and stabilization terms. This implies that, even if a numerical solution belongs to the conforming VEM space (which means that it is continuous), only a projection onto a discontinuous polynomial space is available, since the reconstruction of a conforming object would require the explicit knowledge of the virtual basis functions. Furthermore, it is well known that the stabilization terms, commonly used for dealing with the nonpolynomial component, are of isotropic nature. For this reason, the standard virtual elements formulation shows poor performance when applied to strongly anisotropic problems [32]: indeed, the stabilization terms currently available are not able to catch the anisotropy of the problem under consideration and the results appear polluted.

In order to overcome the limitations we have just mentioned, the explicit computation of the basis functions would be desiderable but is of course unfeasible

because it is prohibitive from the computational point of view. Therefore, it would be desirable to have a possibly cheap machinery for locally constructing more or less accurate approximations of the basis functions when needed. Resorting to model order reduction techniques could help us in this sense.

One of the most popular model order reduction techniques is the so-called Reduced Basis (RB) method [90, 112], which has been introduced during the last few decades. The aim of this method is to provide an efficient and robust tool for numerically solving parametrized PDEs for a large amount of different values of the involved parameters. The RB method consists in a computationally intensive *offline* phase, which is carried out once for all to solve the equation for a large enough sample of parameters. This solutions are commonly called *snapshots*. Then, a new discretization space (of reduced dimension) is spanned by a small number of linear combinations of the snapshots, so that an *online* phase leveraging such a space allows to cheaply solve the parametric problem for each new value of the parameter.

Our proposal is to use the RB method in its "geometric" version (see [90, Chap. 6, Sec. 2]) . Indeed, the elemental PDE associated to virtual basis functions on elements with different geometries can be reformulated as a parametric problem defined on a fixed reference element. In other words, the shape of polygons belonging to a virtual element mesh is treated as a parameter. Applying this approach, we are then in condition to efficiently reconstruct the nonpolynomial part of virtual element functions with different degrees of accuracy which allows to retrieve several other quantities of interests. We point out that, since each polygon is treated independently from the others, the entire procedure is highly parallelizable and applicable only where really needed.

We have in mind two possible ways for exploiting this new approach in the design and implementation of VEM. First, in a post-processing framework, it allows the reconstruction of an actual conforming solution: an object of this kind is useful for performing tasks such as visualization, pointwise evaluation and full evaluation of the error, overcoming the standard procedures based on projections onto polynomials. Second, since we are able to actually reconstruct the nonpolynomial part, we can design stabilization terms which are a true approximation of the residual contribution. In this case, the virtuality of the method is retained when a very small number of reduced basis functions is employed. For example, we anticipate here that the use of a single reduced basis function will be sufficient to improve the

performance of VEM for anisotropic PDEs. Let us also observe that, by means of the RB reconstruction, we can also define a fully conforming method based on the virtual element space, in the spirit of the approach proposed by Manzini, Russo and Sukumar [97]. In this case, the entire VEM machinery is used to manage the polynomial part, while the RB method focuses on the nonpolynomial component. This kind of "finite element approach" dealing the stabilization term is not completely new to the VEM community, since a similar idea has been used by Wriggers, Reddy and collaborators for a nonlinear virtual elements formulation for finite deformations [125].

This chapter consists of several sections. In Section 5.1, we describe how the elemental PDE associated to the basis functions of the lowest order VEM can be written as parametric problem. Then, after recalling in Section 5.2 the main features of the RB method, we describe in Section 5.3 how it is exploited to design our approach to VEM. The validation of this new method is described by means of several numerical tests in Section 5.4. The design of a new kind of stabilization term is discussed in Section 5.5, while in the last Section 5.6, we show how the reduced basis approach is used for post-processing VEM solutions.

5.1 Virtual nodal basis functions as solutions to parametric equations

The method we are going to present and discuss in this chapter is designed for the lowest order virtual element space; its extension to high order polynomial degrees will be subject of future studies. Therefore, given a generic polygon E with N vertices, we consider the discrete space

$$V_1(E) = \{v \in \mathrm{H}^1(E) : v_{|e} \in \mathcal{P}_1(e)\ \forall e \in \partial E,\ \Delta v = 0 \text{ inside } E\}. \tag{5.1}$$

The degrees of freedom of $v_h \in V_1(E)$ are given by the values of v_h at the N vertices of E.

Remark 5.1.1. *The space $V_1(E)$ coincides with the conforming polygonal finite element method based on harmonic barycentric coordinates introduced by Sukumar and Tabarraei [115]. However, the construction in the virtual elements setting is different [12].*

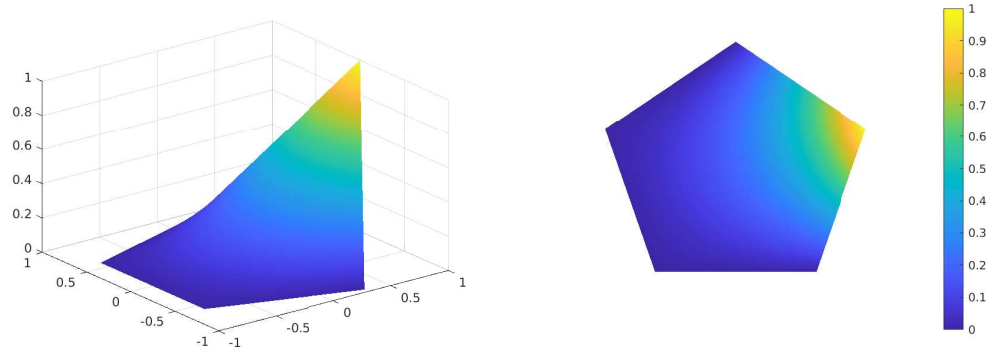


Figure 5.1: Profile of a nodal basis function for $V_1(E)$ on a regular pentagon.

The space $V_1(E)$ can be spanned by making use of N nodal basis functions $\{\varphi_1, \dots, \varphi_N\}$, where φ_j is the nodal basis function associated to the vertex $\mathbf{v}_j$. Consequently, any $w_h \in V_1(E)$ can be written as

$$w_h = \sum_{j=1}^{N} w_h(\mathbf{v}_j)\varphi_j. \tag{5.2}$$

The profile of such basis functions is visualized in Figure 5.1 in the case of a regular pentagon. From the definition of $V_1(E)$, we see that the basis functions $\varphi_1, \dots, \varphi_N$ are solutions of the following boundary value problem, where δ denotes the Kronecker delta.

Problem 5.1.1. *For $j = 1, \dots, N$, find φ_j such that*

$$\begin{aligned} -\Delta\varphi_j &= 0 \quad && \text{in } E \\ \varphi_j &= g_j \quad && \text{on } \partial E \end{aligned} \tag{5.3}$$

with $g_j \in \mathbb{B}_k(\partial E)$ such that $g_j(\mathbf{v}_i) = \delta_{ij}$ for $i = 1, \dots, N$.

It is well known that in classical finite elements several procedures are easily carried out since the basis functions, usually polynomials, are known in their analytical form: for instance, stiffness, mass and other type of matrices are assembled with explicit evaluation of the basis functions for quadrature purposes or, in post-processing framework, shape functions are used to evaluate quantities of interests for the final user, such as reconstruction in subdomains, pointwise evaluation and computation of the error when benchmarking the method. On the contrary, we

have already mentioned that in the case of virtual element methods, all the quantities of interest must be computed without explicitly knowing the basis functions: the method is then constructed in an approximate way: suitable choices of degrees of freedom and polynomial projectors allow the exact computation of the polynomial contribution, while stabilization terms take care of the nonpolynomial part.

In certain situations, this kind of construction may negatively affect the convergence for the method. This generally depends on the nature of the problem under consideration, but, for instance, this happens when dealing with anisotropic problems, since standard stabilization terms are inherently isotropic. Moreover, the final user might wish to access the continuous discrete solution, while, once the degrees of freedom of the solution are computed, only several kind of projections onto discontinuous piecewise polynomial are available.

The ability of reconstructing in a cheap and efficient way a possibly rough approximation of the virtual basis functions should instead allow the design of stabilization terms able to catch the anisotropy of the problem. Moreover, with the same approach, we can reconstruct the continuous solution, also enabling pointwise evaluations.

In order to efficiently carry out this kind of tasks, we resort to a model order reduction technique. In particular, we choose the reduced basis method, which has been designed for solving parametric PDEs. More precisely, for fixed N, all the different instances of Problem 5.1.1 corresponding to different N edges elements E can be treated as a collection of parametric equations where the parameter is the geometry of the element E. To this aim, we reformulate Problem 5.1.1 on a reference element $\widehat{E}$ by a change of variable.

Let us consider as reference domain the N vertices regular polygon $\widehat{E}$ centered in the origin. The vertices $\widehat{\mathbf{v}}_1, \dots, \widehat{\mathbf{v}}_N$ are ordered counterclockwise and we denote by $\widehat{\mathbf{x}} = (0, 0)$ the barycenter.

We introduce the following parameter space

$$\mathscr{P} = \{E : \ E \text{ polygon with } N \text{ vertices}, ker(E) \neq \emptyset, \, d_E = 1, \ \mathbf{x}_E = (0, 0)\}, \quad (5.4)$$

where we denote by d_E the diamater of the circumscribed circle to E, and by $\mathbf{x}_E$ the barycenter of the kernel of E. The kernel $ker(E)$ is the set of points with respect to which the polygon is star shaped; in particular, in case of convex polygons, the kernel coincides with the polygon itself. The restriction to the case of

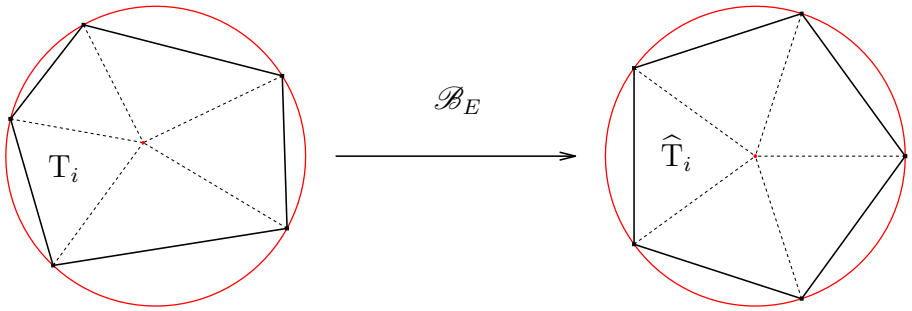


Figure 5.2: An example of affine mapping $\mathscr{B}_E$ between a random pentagon and its regular counterpart.

polygons with unit circumscribed diameter and centered in the origin is reasonable since the behavior of the problem under consideration with respect to translations and rescaling of the domain is well known. Let us observe that the considered parameter space can be further restricted in dependence of the features of the VEM tessellations: for instance, when the virtual element space is defined on Voronoi meshes, one may consider a parameter space consisting only of convex polygons.

Exploiting the fact that polygons considered in virtual elements framework are star shaped, we partition both E and the reference $\widehat{E}$ in N triangles. More precisely, if we assume that also the vertices $\mathbf{v}_1, \dots, \mathbf{v}_N$ of E are ordered counterclockwise, using the convention that $\mathbf{v}_{N+1} = \mathbf{v}_1$ and $\widehat{\mathbf{v}}_{N+1} = \widehat{\mathbf{v}}_1$, we denote by T_i the triangle with vertices $\mathbf{v}_i, \mathbf{v}_{i+1}$ and $\mathbf{x}_E$, while $\widehat{\mathrm{T}}_i$ is similarly constructed in $\widehat{E}$. Therefore, we have the following decompositions

$$E = \bigcup_{i=1}^{N} \mathrm{T}_i \quad \text{and} \quad \widehat{E} = \bigcup_{i=1}^{N} \widehat{\mathrm{T}}_i. \tag{5.5}$$

This procedure is sketched in Figure 5.2.

At this point, we introduce the following continuous piecewise affine transformation from the generic E to $\widehat{E}$

$$\mathscr{B}_E : E \longrightarrow \widehat{E}, \qquad \mathscr{B}_E(\mathbf{x}) = \mathsf{B}_{E,i}\,\mathbf{x} \quad \text{on } \mathrm{T}_i, \tag{5.6}$$

where $\mathsf{B}_{E,i}$ are invertible 2×2 matries defined in such a way that $\mathscr{B}_E$ maps vertices and centroid of E to the corresponding vertices and centroid of $\widehat{E}$, i.e. $\mathscr{B}_E(\mathbf{v}_i) = \widehat{\mathbf{v}}_i$ for all the vertices $\mathbf{v}_i$ of E, and $\mathscr{B}_E(\mathbf{x}_E) = \widehat{\mathbf{x}}$. By construction we then have that

$\mathscr{B}(\mathrm{T}_i) = \widehat{\mathrm{T}}_i$, for $i = 1, \dots, N$. It is easy to see that each matrix $\mathsf{B}_{E,i}$ can be expressed in terms of the coordinates of the vertices, indeed we have

$$\mathsf{B}_{E,i} = \begin{bmatrix} \mathbf{v}_{i,x} & \mathbf{v}_{i+1,x} \\ \mathbf{v}_{i,y} & \mathbf{v}_{i+1,y} \end{bmatrix} \begin{bmatrix} \widehat{\mathbf{v}}_{i,x} & \widehat{\mathbf{v}}_{i+1,x} \\ \widehat{\mathbf{v}}_{i,y} & \widehat{\mathbf{v}}_{i+1,y} \end{bmatrix}^{-1}, \tag{5.7}$$

where $\widehat{\mathbf{v}}_i = (\widehat{\mathbf{v}}_{i,x}, \widehat{\mathbf{v}}_{i,y})$ and $\mathbf{v}_i = (\mathbf{v}_{i,x}, \mathbf{v}_{i,y})$.

Before transferring Problem 5.1.1 to the reference domain, let us write it in variational form.

Problem 5.1.2. *For $j = 1, \dots, N$, find $\varphi_j \in \mathrm{H}^1(E)$ such that*

$$\begin{aligned} \int_E \nabla \varphi_i \cdot \nabla v \, \mathrm{d}\mathbf{x} = 0 \quad \forall v \in \mathrm{H}^1_0(E) \\ \varphi_j = g_j \quad \text{on } \partial E \end{aligned} \tag{5.8}$$

with g_j piecewise linear function on ∂E such that $g_j(\mathbf{v}_i) = \delta_{ij}$ for $i = 1, \dots, N$.

In order to move Problem 5.1.2 to a problem with solution in $\mathrm{H}^1(\widehat{E})$, we perform a change of variable in the integral of Equation (5.8)

$$\int_E \nabla u \cdot \nabla v \, \mathrm{d}\mathbf{x} = \sum_{i=1}^{N} \int_{\mathrm{T}_i} \nabla u \cdot \nabla v \, \mathrm{d}\mathbf{x} = \sum_{i=1}^{N} \int_{\widehat{\mathrm{T}}_i} |\det(\mathsf{B}_{E,i}^{-1})| \, \mathsf{B}_{E,i}^{\top} \mathsf{B}_{E,i} \, \nabla \widehat{u} \cdot \nabla \widehat{v} \, \mathrm{d}\widehat{\mathbf{x}}.$$

After this operation, we introduce the parameter dependent bilinear form A defined as

$$A(\widehat{u}, \widehat{v}; E) = \sum_{i=1}^{N} \int_{\widehat{\mathrm{T}}_i} |\det(\mathsf{B}_{E,i}^{-1})| \, \mathsf{B}_{E,i}^{\top} \mathsf{B}_{E,i} \, \nabla \widehat{u} \cdot \nabla \widehat{v} \, \mathrm{d}\widehat{\mathbf{x}}, \tag{5.9}$$

so that we can write an equivalent formulation of Problem 5.1.2 as parametric equation on the reference element.

Problem 5.1.3. *For $j = 1, \dots, N$, find $\widehat{\varphi}_j^E \in \mathrm{H}^1(\widehat{E})$ such that*

$$\begin{aligned} A(\widehat{\varphi}_j^E, \widehat{v}; E) = 0 \quad \forall \widehat{v} \in \mathrm{H}^1_0(\widehat{E}) \\ \widehat{\varphi}_j^E = \widehat{g}_j \quad \text{on } \partial \widehat{E} \end{aligned} \tag{5.10}$$

with $\widehat{g}_j$ piecewise linear function on $\partial \widehat{E}$ such that $\widehat{g}_j(\widehat{\mathbf{v}}_i) = \delta_{ij}$ for $i = 1, \dots, N$.

As we anticipated, we are going to tackle this problem by resorting to the reduced basis method, which have been introduced for efficiently solving parametric PDEs for large numbers of instances of the parameter.

5.2 The reduced basis method

In this section, we recall the main features of the reduced basis method basing our discussion on two monographs by Hesthaven, Rozza, Stamm [90] and Quarteroni, Manzoni, Negri. [112]. We describe the general method to tackle parametrized PDEs, introducing the idea of *solution manifold* and *affine decomposition*, which are the building blocks we need to design an accurate and efficient model order reduction approach.

5.2.1 General idea

Let us consider the following family of parameter dependent problems.

Problem 5.2.1. *Find* $u[\mu] \in \mathcal{V}$ *such that*

$$A(u[\mu], v; \mu) = F(v; \mu) \qquad \forall v \in \mathcal{V}. \tag{5.11}$$

We denote by μ a vector parameter belonging to the parameter set $\mathscr{P} \subset \mathbb{R}^L$. This set may represent physical coefficients related to the considered problem, several boundary conditions, right hand sides or the geometry of the computational domain. Then, $A(\cdot, \cdot; \mu) : \mathcal{V} \times \mathcal{V} \to \mathbb{R}$ is a continuous and parameter dependent bilinear form which we assume to be coercive for all values $\mu \in \mathscr{P}$. For the right hand side, we have $F(\cdot; \mu) : \mathcal{V} \to \mathbb{R}$ denoting a parameter dependent linear operator, continuous in $\mathcal{V}$ for all values of the parameter μ.

If we need to numerically solve this problem, for instance by a Galerkin method, we usually introduce a large finite dimensional space $\mathcal{V}_\delta$, included in $\mathcal{V}$, defined as

$$\mathcal{V}_\delta = span\{\psi_1, \dots, \psi_\mathcal{N}\} \tag{5.12}$$

and, for each value of μ, we seek for the approximation $u_\delta[\mu] \in \mathcal{V}_\delta$ to $u[\mu] \in \mathcal{V}$.

It is clear that solving Problem 5.2.1 in $\mathcal{V}_\delta$ for a large set of parameters is extremely costly. In order to mitigate the computational cost, one should use model order reduction techniques, such as the reduced basis method.

The reduced basis method is built on the assumption that the solution manifold, that is the set

$$\mathcal{M} = \{u_\delta[\mu] : \mu \in \mathscr{P}\} \subset \mathcal{V}_\delta$$

of all the approximate solutions $u_\delta[\mu]$ in variation of the parameter, can be approximated using a lower dimensional linear space

$$\mathcal{W} = span\{\xi_1, \dots, \xi_M\} \subset \mathcal{V}_\delta \tag{5.13}$$

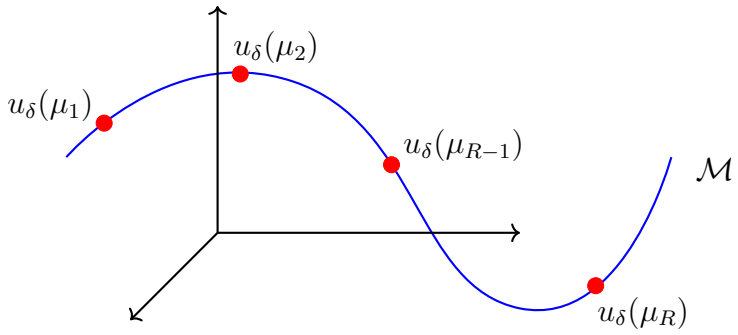


Figure 5.3: Intuitive representation of the solution manifold $\mathcal{M}$ with some chosen snapshots.

with $M \ll \mathcal{N}$, where the functions ξ_ℓ, for $\ell = 1, \dots, M$ are constructed as linear combinations of discrete snapshots $u_\delta[\mu_r]$ computed for suitably chosen parameters $\mu_r \in \mathscr{P}$.

Once the reduced space $\mathcal{W}$ is constructed, for each new value of μ, we seek for the corresponding approximate solution $u^{\mathrm{rb}}[\mu] = \sum_{\ell=1}^{M} x_\ell \, \xi_\ell$ of Problem 5.2.1 in $\mathcal{W}$: the cost for carrying out this operation is independent from the dimension $\mathcal{N}$ of the full space $\mathcal{V}_\delta$ and depends only on the dimension M of $\mathcal{W}$. Indeed, the discrete problem is equivalent to a small $M \times M$ linear system of the form

$$\mathsf{A}\,[\mu]\,\mathsf{x} = \mathsf{F}\,[\mu]\,, \tag{5.14}$$

where $\mathsf{x} = (x_\ell)_\ell$ denotes the coefficient vector of the solution in $\mathcal{W}$ with respect to $\xi_1, \dots, \xi_M$ and where matrix and right hand side are defined as

$$(\mathsf{A}\,[\mu])_{\ell,\ell'} = A(\xi_{\ell'}, \xi_\ell; \mu), \qquad (\mathsf{F}\,[\mu])_\ell = F(\xi_\ell; \mu). \tag{5.15}$$

Under the so-called *affine decomposition assumption*, the reduced system (5.14) is assembled with a very cheap procedure by making use of some quantities we can precompute at the time of the snapshots construction. Let us assume that both the bilinear form A and the right hand side F can be written in terms of parameter independent bilinear forms A^q, $q = 1, \dots, Q_A$ and linear functionals F^q, $q = 1, \dots, Q_F$ respectively, as

$$A(u, v; \mu) = \sum_{q=1}^{Q_A} \alpha_A^q[\mu] A^q(u, v), \qquad F(v; \mu) = \sum_{q=1}^{Q_F} \alpha_F^q[\mu] F^q(v), \tag{5.16}$$

where the dependence on the parameter is confined to the coefficients only, which are determined by given functions $\alpha_A^q : \mathscr{P} \to \mathbb{R}$ and $\alpha_F^q : \mathscr{P} \to \mathbb{R}$. Thanks to this construction, once the reduced basis is built, we can precompute and store the matrices A^q with $(\mathsf{A}^q)_{\ell,\ell'} = A^q(\xi_{\ell'}, \xi_\ell)$ and F^q with $(\mathsf{F}^q)_\ell = F^q(\xi_\ell)$, in such a way that $\mathsf{A}[\mu]$ and $\mathsf{F}[\mu]$ in (5.14) are easily obtained as

$$\mathsf{A}[\mu] = \sum_{q=1}^{Q_A} \alpha_A^q[\mu]\mathsf{A}^q, \qquad \mathsf{F}[\mu] = \sum_{q=1}^{Q_F} \alpha_F^q[\mu]\mathsf{F}^q.$$

Remark 5.2.1. *We observe that the matrix $\mathsf{A}[\mu]$ is generally full, while the one associated to the full space $\mathcal{V}_\delta$ is usually sparse. In general, this is not an issue for the method: System* (5.14) *is small, hence its solution is less computationally intensive than solving a system in $\mathcal{V}_\delta$.*

5.2.2 How to construct a reduced basis

We have mentioned above that the construction of the reduced basis is done by suitably combining snapshots $u_\delta[\mu]$ for a sample of values of μ. These snapshots are computed *offline* in the fine space $\mathcal{V}_\delta$ by solving $\mathcal{N} \times \mathcal{N}$ linear systems. To this aim, we first select a sample of parameters

$$S = \{\mu_1, \dots, \mu_R\} \subset \mathscr{P}, \tag{5.17}$$

which is usually randomly generated, accordingly to a certain probability distribution. If S is large enough, the set $\{u[\mu] : \mu \in S\}$ is hopefully a good representation of the solution manifold $\mathcal{M}$.

Once we have computed the snapshots $u_\delta[\mu_1], \dots, u_\delta[\mu_R]$, we select a suitable $\mathcal{W} \subset \mathcal{V}_\delta$ of dimension M spanned by linear combinations of elements of S. In order to carry out this task, several algorithms have been designed: we mention, for instance, the greedy selection and the proper orthogonal decomposition (POD). We focus on the POD algorithm, recalling its main features. The POD is called *explore-and-compress strategy* since it is based on a compression of the computed snapshots with the aim of retaining only the essential information given by the sample S. Each snapshots $u_\delta[\mu_r]$ can be represented as a column vector containing the values of the $\mathcal{N}$ degrees of freedom of the solution itself in $\mathcal{V}_\delta$. Therefore, we can collect all the snapshots in a matrix U of dimension $\mathcal{N} \times R$

$$\mathsf{U} = \left[\; u_\delta[\mu_1] \;\middle|\; \dots \;\middle|\; u_\delta[\mu_R] \; \right]. \tag{5.18}$$

At this point, we compute the correlation matrix $\mathsf{C} = R^{-1}\mathsf{U}^\top\mathsf{S}\mathsf{U}$, where S denotes the stiffness matrix for a scalar product of $\mathcal{V}_\delta$, and compute its eigenvalues and eigenvectors $(\lambda_\ell, \mathbf{z}_\ell)$ for $\ell = 1, \ldots, R$. We assume that the sequence of eigenvalues $\{\lambda_\ell\}$ is sorted in non increasing order, hence $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_R$. In order to obtain a new basis $\{\xi_1, \ldots, \xi_R\}$ for the span of the snapshots, we set

$$\xi_\ell = \frac{1}{\sqrt{R}} \sum_{r=1}^{R} z_\ell^r \, u_\delta[\mu_r] \qquad \text{for } \ell \leq R, \tag{5.19}$$

where z_ℓ^r denotes the r^{th} entry of the eigenvector $\mathbf{z}_\ell$, where $\mathbf{z}_\ell = (z_\ell^r)_{r=1}^R$. Then, the reduced basis is easily obtained by truncation of the new basis $\{\xi_1, \ldots, \xi_R\}$ to the first $M \leq R$ elements, so that we finally have

$$\mathcal{W} = span\{\xi_\ell, \ \ell = 1, \cdots, M\}.$$

Once the space is constructed, we can precompute and store (once for all) the building blocks for the affine decomposition

$$(\mathsf{A}^q)_{\ell,\ell'} = A^q(\xi_{\ell'}, \xi_\ell), \qquad (\mathsf{F}^q)_\ell = F^q(\xi_\ell), \tag{5.20}$$

so that, for each new value of the parameter μ, we can efficiently solve Problem 5.2.1 in the reduced space $\mathcal{W}$.

5.3 Computation of virtual basis functions with reduced basis method

In this section we describe how to apply the reduced basis method for solving the parameter dependent Problem 5.1.3, which represents the elemental elliptic equation solved by the virtual basis functions of $V_1(E)$, for a given element E with N vertices. By parametrization, the problem under consideration is defined on the reference polygon $\widehat{E}$ and E represents the parameter.

Let us fix the number of vertices N and introduce a fine triangulation $\mathcal{T}_\delta$ for $\widehat{E}$ with mesh size δ. We then denote by $\mathcal{V}_\delta$ the associated finite element space made up of piecewise linear polynomials

$$\mathcal{V}_\delta = \{\widehat{u}_\delta \in \mathrm{H}^1(\widehat{E}) : \ \widehat{u}_\delta|_\tau \in \mathcal{P}_1(\tau), \ \forall \tau \in \mathcal{T}_\delta\}.$$

We observe that Problem 5.1.3 is endowed with non homogeneous boundary conditions: it will be convenient to reformulate the mentioned equation as a problem with homogeneous Dirichlet boundary conditions. To this aim, let us introduce the discrete harmonic lifting $\widehat{\Lambda}_j \in \mathcal{V}_\delta$ of the boundary function $\widehat{g}_j$, defined as

$$\begin{aligned} \int_{\widehat{E}} \nabla \widehat{\Lambda}_j \cdot \nabla \widehat{v} \, \mathrm{d}\widehat{\mathbf{x}} = 0 \qquad & \forall \widehat{v} \in \mathcal{V}_\delta \cap \mathrm{H}_0^1(\widehat{E}) \\ \widehat{\Lambda}_j = \widehat{g}_j \qquad & \text{on } \partial\widehat{E} \end{aligned} \tag{5.21}$$

for $j = 1, \ldots, N$. Each virtual basis $\widehat{\varphi}_j^E$ of E constructed in the reference $\widehat{E}$ can be written as

$$\widehat{\varphi}_j^E = \widehat{\Lambda}_j + \widehat{\chi}_j[E],$$

where $\widehat{\chi}_j[E] \in \mathrm{H}_0^1(\widehat{E})$ is solution of the following parametrized problem.

Problem 5.3.1. *Find* $\widehat{\chi}_j[E] \in \mathrm{H}_0^1(\widehat{E})$ *such that*

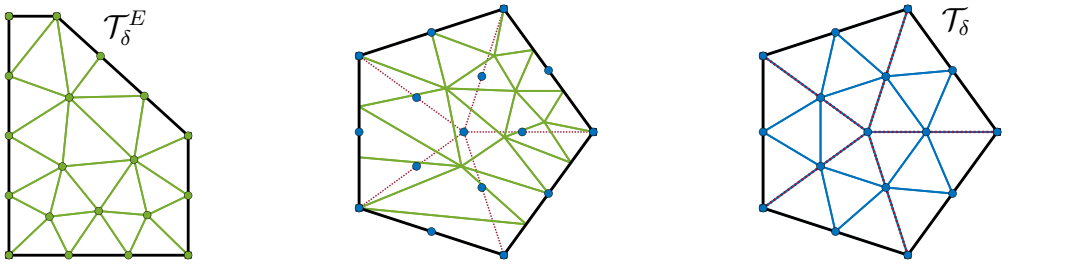
$$A(\widehat{\chi}_j[E], \widehat{v}; E) = -A(\widehat{\Lambda}_j, \widehat{v}; E) \qquad \forall \widehat{v} \in \mathrm{H}_0^1(\widehat{E}). \tag{5.22}$$

5.3.1 The offline phase: snapshots computation

We extract a random sample S of polygons in $\mathscr{P}$ and then for each $E \in S$ we compute the associated snapshots $\widehat{\chi}_j[E]$ by solving Problem 5.3.1 for $j = 1, \ldots, N$.

We observe that the sample S may contain badly shaped polygons, associated to a mapping $\mathscr{B}$ characterized by strong gradients. In this case, the bilinear form $A(\cdot, \cdot; E)$ on $\widehat{E}$ may be ill conditioned: then it is not convenient to solve such a problem by finite elements on the triangulation $\mathcal{T}_\delta$. In order to overcome this problem, we proceed as follows. We solve the equivalent Problem 5.1.2 on the physical polygon E via finite element method on a triangulation $\mathcal{T}_\delta^E$, with size δ^E, directly constructed in E. Notice that $\mathcal{T}_\delta^E$ is independent from $\mathcal{T}_\delta$. At this point, we have computed a set of N finite element functions $\varphi_{j,\delta}^E \in \mathrm{H}^1(E)$ which are the pull back of functions $\widetilde{\varphi}_j^E \in \mathrm{H}^1(\widehat{E})$. Given $\widetilde{\varphi}_j^E$, we know its values at the nodes obtained by applying a push forward to the nodes of the triangulation $\mathcal{T}_\delta^E$. In general, these mapped nodes do not coincide with the nodes of $\mathcal{T}_\delta$. Then, in order to compute the snapshots $\widehat{\chi}_{j,\delta}[E]$ on $\mathcal{T}_\delta$, we introduce the standard Lagrangian interpolation operator $I_\delta : C^0(\widehat{E}) \longrightarrow \mathcal{V}_\delta$, and we let

$$\widehat{\chi}_{j,\delta}[E] = I_\delta \, \widetilde{\varphi}_j^E - \widehat{\Lambda}_j \qquad j = 1, \ldots, N. \tag{5.23}$$



Step 1: compute $e_{j,\delta}^E$ **Step 2:** $\widetilde{e}_j^E = e_{j,\delta}^E \circ \mathscr{B}^{-1}$ **Step 3:** interpolate $I_\delta \widetilde{e}_j^E$ **Step 4:** $\widehat{\chi}_{j,\delta}[E] = I_\delta \widetilde{e}_j^E - \widehat{\Lambda}_j$

Figure 5.4: Sketch of snapshots computation: functions are computed on a mesh $\mathcal{T}_\delta^E$ for E and then interpolated on the fixed mesh $\mathcal{T}_\delta$ of $\widehat{E}$

This procedure is sketched in Figure 5.4.

Once we have computed the snapshots, we are able to construct the reduced basis functions. We believe that in order to design stabilization terms able to deal with anisotropic problems, it is important that we take into account the relation between different basis functions. For this reason, in our method, the POD algorithm is applied on the local basis taken as a whole and not on N individual basis functions taken one by one. Let us denote by E^r the r$^{\text{th}}$ polygon in S and set $\widehat{\chi}_{j,\delta}^r = \widehat{\chi}_{j,\delta}[E^r]$. Moreover, with abuse of notation, let us indicate with the same symbol both the functions in $\mathcal{V}_\delta$ and the corresponding vector of finite element degrees of freedom. We construct the r$^{\text{th}}$ column of the snapshots matrix U defined in (5.18) by stacking on top of each other the finite element coordinates $\widehat{\chi}_{1,\delta}^r, \ldots, \widehat{\chi}_{N,\delta}^r$ so that each column contains all the basis functions for the associated polygon. We set

$$\mathsf{U} = \begin{bmatrix} \widehat{\chi}_{1,\delta}^1 & \cdots & \widehat{\chi}_{1,\delta}^R \\ \vdots & & \vdots \\ \widehat{\chi}_{N,\delta}^1 & \cdots & \widehat{\chi}_{N,\delta}^R \end{bmatrix}$$

We then apply the POD to this matrix and we obtain an ordered sequence of N-tuples $(\widehat{\xi}_1^\ell, \cdots, \widehat{\xi}_N^\ell)^\top$ constructed as

$$\begin{pmatrix} \widehat{\xi}_1^\ell \\ \vdots \\ \widehat{\xi}_N^\ell \end{pmatrix} = \frac{1}{\sqrt{R}} \sum_{r=1}^{R} z_\ell^r \begin{pmatrix} \widehat{\chi}_{1,\delta}^r \\ \vdots \\ \widehat{\chi}_{N,\delta}^r \end{pmatrix} \qquad \text{for } \ell = 1, \ldots, R, \tag{5.24}$$

where $(\lambda_\ell, \mathbf{z}_\ell)$ for $\ell = 1, \ldots, R$, represents the set of eigenpairs of the correlation matrix $\mathsf{C} = R^{-1}\mathsf{U}^\top \mathsf{S} \mathsf{U}$.

Finally, the M dimensional reduced basis space in which we will look for an approximation of the virtual function corresponding to the j$^{\text{th}}$ node of a new polygon E is defined as

$$\mathcal{W}_{j,M} = span\{\widehat{\xi}_j^1, \dots, \widehat{\xi}_j^M\} \subset \mathcal{V}_\delta \cap \mathrm{H}_0^1(E), \qquad j = 1, \dots, N. \tag{5.25}$$

The entire offline phase is summarized in Algorithm 5.1.

Algorithm 5.1 The offline phase

Data:

S: sample of R polygons with N vertices
$\widehat{E}$: regular polygon with N vertices
$\widehat{g}_j$: boundary conditions on $\partial\widehat{E}$, $j = 1, \dots, N$
$\widehat{\Lambda}_j \in \mathcal{V}_\delta$: harmonic lifting of $\widehat{g}_j$

Snapshots computation:

for $E \in S$ **do**
  Compute $e_{j,\delta}^E$ solving Problem 5.1.1 with FEM on triangulation $\mathcal{T}_\delta^E$ of E
  Set $\widetilde{e}_j^E = e_{j,\delta}^E \circ \mathscr{B}^{-1}$ defined on $\mathscr{B}(\mathcal{T}_\delta^E)$ in $\widehat{E}$
  Compute $I_\delta \widetilde{e}_j^E$ interpolating $\widetilde{e}_j^E$ on $\mathcal{T}_\delta$
  Compute snapshots $\widehat{\chi}_{j,\delta}[E] = I_\delta \widetilde{e}_j^E - \widehat{\Lambda}_j \in \mathcal{V}_\delta$
end

Proper Orthogonal Decomposition:

Build snapshots matrix U and correlation matrix $\mathsf{C} = R^{-1}\mathsf{U}^\top \mathsf{S}\, \mathsf{U}$
Compute sequence $(\lambda_\ell, \mathbf{z}_\ell)$, $\ell = 1, \dots, R$, solving the eigenvalue problem $\mathsf{C}\mathbf{z} = \lambda\mathbf{z}$
Build $(\widehat{\xi}_1^\ell, \cdots, \widehat{\xi}_N^\ell)^\top$ according to (5.24)
Compute and store affine decomposition building blocks A_i^ν, F_i^ν (see Sec. 5.3.2)

5.3.2 The affine decomposition

We know that the efficiency of the reduced basis method relies on the affine decomposition assumption. Therefore, we need to provide an affine decomposition for Problem 5.3.1, in particular for the bilinear form $A(\cdot, \cdot; E)$ and the right hand side operator $A(\widehat{\Lambda}_j, \cdot; E)$. Let us observe that, using the notation introduced in

Section 5.1, we have

$$A(\widehat{u},\widehat{v};E)=\sum_{i=1}^{N}A(\widehat{u},\widehat{v};\mathrm{T}_i)\quad\text{where}\quad A(\widehat{u},\widehat{v};\mathrm{T}_i)=\int_{\widehat{\mathrm{T}}_i}|\det(\mathsf{B}_{E,i}^{-1})|\,\mathsf{B}_{E,i}^{\top}\mathsf{B}_{E,i}\nabla\widehat{u}\cdot\nabla\widehat{v}\,\mathrm{d}\widehat{\mathbf{x}}.$$

Since the matrix $|\det(\mathsf{B}_{E,i}^{-1})|\,\mathsf{B}_{E,i}^{\top}\mathsf{B}_{E,i}$ is symmetric, we introduce a basis for the space of 2×2 symmetric matrices

$$\mathcal{S}^1=\begin{bmatrix}1&0\\0&0\end{bmatrix},\qquad \mathcal{S}^2=\begin{bmatrix}0&0\\0&1\end{bmatrix},\qquad \mathcal{S}^3=\begin{bmatrix}0&1\\1&0\end{bmatrix},\tag{5.26}$$

so that we can write

$$|\det(\mathsf{B}_{E,i}^{-1})|\,\mathsf{B}_{E,i}^{\top}\mathsf{B}_{E,i}=\sum_{\nu=1}^{3}c_{\nu}^{i}[E]\mathcal{S}^{\nu}.\tag{5.27}$$

Consequently, we obtain the following affine decomposition

$$A(\widehat{u},\widehat{v};E)=\sum_{i}^{N}\sum_{\nu=1}^{3}c_{\nu}^{i}[E]A^{i,\nu}(\widehat{u},\widehat{v}),\quad\text{with}\quad A^{i,\nu}(\widehat{u},\widehat{v})=\int_{\widehat{\mathrm{T}}_i}\mathcal{S}^{\nu}\nabla\widehat{u}\cdot\nabla\widehat{v}\,\mathrm{d}\widehat{\mathbf{x}}.$$

which is applied in the same way also for the term at the right hand side, yielding

$$A(\widehat{\Lambda}_j,\widehat{v};E)=\sum_{i=1}^{N}\sum_{\nu=1}^{3}c_{\nu}^{i}[E]F_{j}^{i,\nu}(\widehat{v})\quad\text{with}\quad F_{j}^{i,\nu}(\widehat{v})=\int_{\widehat{\mathrm{T}}_i}\mathcal{S}^{\nu}\nabla\widehat{\Lambda}_j\cdot\nabla\widehat{v}\,\mathrm{d}\widehat{\mathbf{x}}.$$

At this point, we have selected a reduced basis and constructed the building blocks for the affine decomposition, therefore we can compute and store once for all some further bricks we need to implement an efficient online phase. In particular, since we may need to evaluate also non symmetric bilinear forms, we add an element to the basis of symmetric matrices, that is

$$\mathcal{S}^4=\begin{bmatrix}0&1\\-1&0\end{bmatrix},$$

and, given $\mathcal{W}_{j,M}$ for $j=1,\dots,N$, we compute and store the following quantities

$$\mathsf{A}_i^{\nu}(j,j',\ell,\ell')=\int_{\widehat{\mathrm{T}}_i}\mathcal{S}^{\nu}\nabla\widehat{\xi}_j^{\ell}\cdot\nabla\widehat{\xi}_{j'}^{\ell'},\qquad \mathsf{F}_i^{\nu}(j,j',\ell)=\int_{\widehat{\mathrm{T}}_i}\mathcal{S}^{\nu}\nabla\widehat{\xi}_j^{\ell}\cdot\nabla\widehat{\Lambda}_{j'}\tag{5.28}$$

for $\ell,\ell'=1,\cdots,M$, $j,j'=1,\cdots,N$, and for $\nu=1,\cdots,4$.

5.3.3 The online phase: reconstruction of basis functions

We are now able to compute approximations of the virtual basis functions for the local space $V_1(E)$. Given a new polygon E with N vertices, we look for $\widehat{\chi}_{j,\delta}[E]$ solving Problem 5.3.1 in the form

$$\widehat{\chi}_{j,\delta}[E] = \sum_{\ell=1}^{M} \mathsf{x}_\ell^{E,j}\, \widehat{\xi}_j^\ell. \tag{5.29}$$

Exploiting the affine decomposition, the linear system we need to solve for getting the coefficients $\mathsf{x}_\ell^{E,j}$, $\ell = 1, \dots, M$ is assembled with a very cheap procedure, as described in Section 5.2. Moreover, if the number of reduced basis M is small, then the system is also cheaply solved. Then, the virtual basis functions φ_j^E is obtained via the pull back $\widehat{\varphi}_{M,j}^{\mathrm{rb}}[E] \circ \mathscr{B}_E$ of the function

$$\widehat{\varphi}_{M,j}^{\mathrm{rb}}[E] = \widehat{\Lambda}_j + \sum_{\ell=1}^{M} \mathsf{x}_\ell^{E,j}\, \widehat{\xi}_j^\ell \in \mathcal{V}_\delta. \tag{5.30}$$

All these steps are summarized in Algorithm 5.2.

Let us point out that the computation of a very rough approximation of the virtual bases may be enough to reach our goals, such as the design of a stabilization term for anisotropic problems (see Section 5.5). Moreover, we observe that the polynomial part of the virtual functions can be exactly computed making use of standard tools, therefore the reduced basis technique can be used just to handle the nonpolynomial part, reducing the impact of the error we commit with the proposed approach. Consequently, the use of few reduced basis (small values of M) should be enough.

Remark 5.3.1. *We observe that, in general, the reduced basis method described in Section 5.2 would consist in looking for a coefficient vector* x^E *so that*

$$\widehat{\chi}_{j,\delta}[E] = \sum_{\ell=1}^{M} \mathsf{x}_\ell^{E}\, \widehat{\xi}_j^\ell \qquad j = 1, \dots, N,$$

where each x_ℓ^E *is independent of the index* j*. On the other hand, in our construction, we build each* $\widehat{\chi}_{j,\delta}[E]$ *independently of the others: a priori, this may imply a better approximation of the exact virtual basis functions.*

Algorithm 5.2 The online phase

Data:

$\widehat{g}_j$: boundary condition on $\partial\widehat{E}$, $j = 1, \dots, N$
$\widehat{\Lambda}_j \in \mathcal{V}_\delta$: harmonic lifting of $\widehat{g}_j$
E: polygon with N vertices

Set $M \leq R$ and consider reduced basis $(\widehat{\xi}_1^\ell, \cdots, \widehat{\xi}_N^\ell)^\top$, $\ell = 1, \cdots, M$

Approximating basis functions for $V_1(E)$:

Assemble reduced system $\mathsf{A}[E]\,\mathsf{x}^E = \mathsf{F}[E]$, with affine decomposition bricks A_i^ν, F_i^ν
Solve for w and construct $\widehat{e}_{M,j}^{\mathrm{rb}}[E] = \widehat{\Lambda}_j + \sum_{\ell=1}^M \mathsf{x}_\ell^{E,j}\,\widehat{\xi}_j^\ell \in \mathcal{V}_\delta$

5.4 Numerical validation

In this section we discuss accuracy and efficiency of the RB technique we introduced in this chapter. Starting from now on, we restrict our discussion to the case of convex polygons. Despite that, we describe two algorithms to generate datasets of convex and general star shaped polygons respectively. The procedure to generate star shaped polygons is reported for completeness and it will be exploited of future studies. We then build a reduced basis for each value of N between 4 and 14 and we study the accuracy of the method comparing the reduced basis reconstructions with the standard projection onto polynomials. Moreover, the efficiency of the method is assessed by measuring the average CPU time required to carry out the online phase on the considered datasets. We point out that the reduced basis we construct in this section will be used later for performing the tests we present in Sections 5.5 and 5.6.

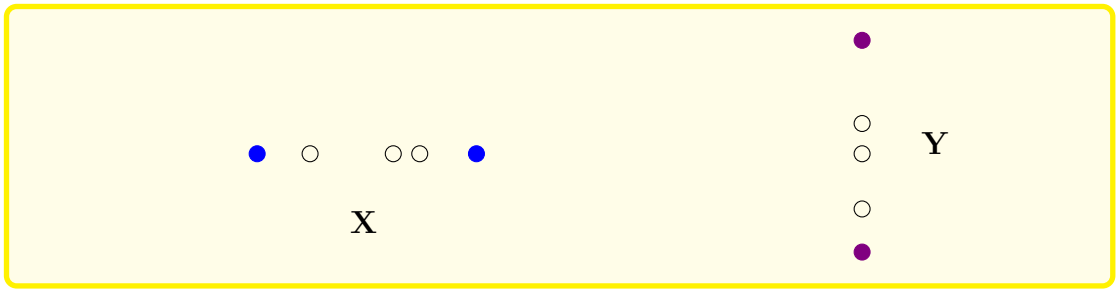
5.4.1 Dataset generation

Convex polygons

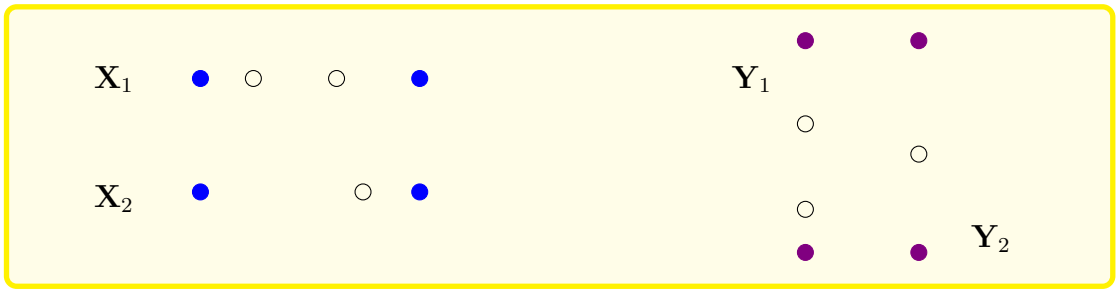
If we need to generate datasets made up of convex polygons, we adopt a different algorithm. There are many options and we decide to work with a sampling algorithm [120] developed by exploiting the theory presented in [119]. For fixed N, it generates random polygons in the unit square which are then translated and scaled to satisfy the defining properties of the parameter space $\mathscr{P}$.

We now describe the procedure one has to carry out in order to obtain a random convex polygons with N edges. The text is accompanied by illustrations providing a simple example of how the procedure works, while the pseudocode can be found in Algorithm 5.3. We report in Figure 5.5 some examples of convex polygons generated with the described algorithm.

We first generate two random vectors $\mathbf{X}$ and $\mathbf{Y}$ with uniform distribution in $(0,1)^N$. Then, we sort their elements in non decreasing order.

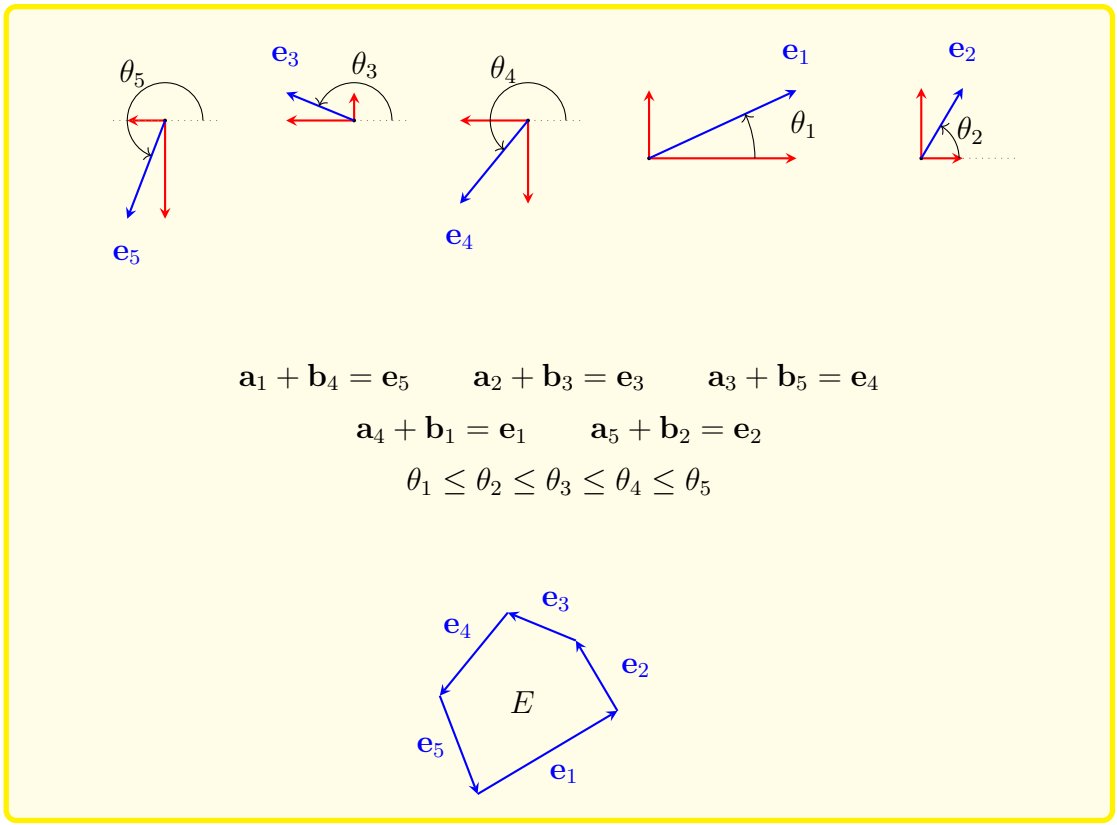


Once this is done, we let $E_{\mathbf{X}}$ and $E_{\mathbf{Y}}$ denote the set of exterior points of $\mathbf{X}$ and $\mathbf{Y}$ respectively, and we randomly divide the interior points of each set into two disjoint subsets $I^1_{\mathbf{X}}, I^2_{\mathbf{X}}$ and $I^1_{\mathbf{Y}}, I^2_{\mathbf{Y}}$. Then, we set $\mathbf{X}_j = E_{\mathbf{X}} \cup I^j_{\mathbf{X}}$ and $\mathbf{Y}_j = E_{\mathbf{Y}} \cup I^j_{\mathbf{Y}}$ for $j = 1, 2$.



From $\mathbf{X}_1$ and $\mathbf{X}_2$ we can identify a set of vectors $\{\mathbf{a}_i\}_{i=1}^N$ such that $\sum_{i=1}^N \mathbf{a}_i = 0$. Similarly, from $\mathbf{Y}_1$ and $\mathbf{Y}_2$, we can create the set of vectors $\{\mathbf{b}_i\}_{i=1}^N$ such that $\sum_{i=1}^N \mathbf{b}_i = 0$.

We *randomly* pair up the obtained vectors to obtain a set of N pairs $\{(\mathbf{a}_i, \mathbf{b}_i)\}_{i=1}^N$ so that we can construct $\{\mathbf{e}_i\}_{i=1}^N$ where $\mathbf{e}_i = \mathbf{a}_i + \mathbf{b}_i$. We sort $\{\mathbf{e}_i\}_{i=1}^N$ with respect to the angle they form when centered in the origin and we finally lay them end-to-end so that the convex polygon is constructed.



Proposition 5.4.1 ([119]). *For fixed N, the described algorithm provides a convex polygon with N vertices.*

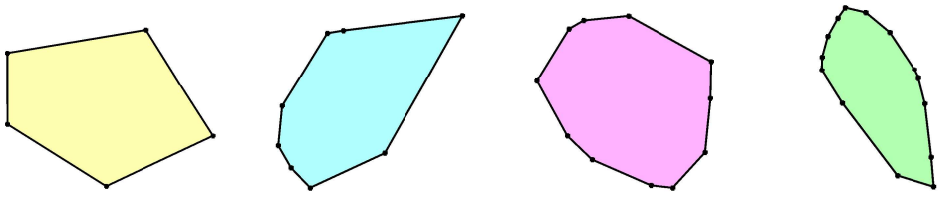


Figure 5.5: Four examples of random convex polygons generated by means of Algorithm 5.3.

Algorithm 5.3 Generating random convex polygons in the unit square

Generate $\mathbf{X}$ and $\mathbf{Y}$ uniformly in $(0,1)^N$
Sort the elements of $\mathbf{X}$ and $\mathbf{Y}$ with non decreasing order

Isolate the extreme points of $\mathbf{X}$ and $\mathbf{Y}$ in two sets $E_{\mathbf{X}}, E_{\mathbf{Y}}$
Randomly divide the interior points of $\mathbf{X}$ into two disjoint subsets $I^1_{\mathbf{X}}, I^2_{\mathbf{X}}$
Repeat the same operation on $\mathbf{Y}$ to get $I^1_{\mathbf{Y}}, I^2_{\mathbf{Y}}$
Set $\mathbf{X}_j = E_{\mathbf{X}} \cup I^j_{\mathbf{X}}$ and $\mathbf{Y}_j = E_{\mathbf{Y}} \cup I^j_{\mathbf{Y}}$ for $j = 1, 2$

Extract vectors $\{\mathbf{a}_i\}_{i=1}^N$ from $\mathbf{X}_1, \mathbf{X}_2$ such that $\sum_{i=1}^N \mathbf{a}_i = 0$
Extract vectors $\{\mathbf{b}_i\}_{i=1}^N$ from $\mathbf{Y}_1, \mathbf{Y}_2$ such that $\sum_{i=1}^N \mathbf{b}_i = 0$
Randomly pair up elements from $\{\mathbf{a}_i\}_{i=1}^N$, $\{\mathbf{b}_i\}_{i=1}^N$ to get a set of pairs $\{(\mathbf{a}_i, \mathbf{b}_i)\}_{i=1}^N$
Compute $\{\mathbf{e}_i\}_{i=1}^N$ as $\mathbf{e}_i = \mathbf{a}_i + \mathbf{b}_i$

Sort $\{\mathbf{e}_i\}_{i=1}^N$ with respect to the angle they form when centered in the origin
Lay end-to-end the ordered vectors to obtain a convex polygon

General star-shaped polygons

In this section, we describe the algorithm we designed for generating random star-shaped polygons centered at the origin and with unit diameter d_E. Given the N vertices regular polygon $\widehat{E}$ centered at the origin, the algorithm consists of applying random deformations to $\widehat{E}$ so that a generic polygon E is obtained.

We represent a polygon $E \in \mathscr{P}$ by the vectors $\boldsymbol{\rho} = (\rho_i)_{i=1}^N$ and $\boldsymbol{\theta} = (\theta_i)_{i=1}^N$, where (ρ_i, θ_i) are the polar coordinates of the vertex $\mathbf{v}_i$. We construct the random instances $\boldsymbol{\rho}$ and $\boldsymbol{\theta}$ of the parameter E by applying a sequence of deformations to the reference polygon, which is represented by the vectors

$$\widehat{\boldsymbol{\rho}} = (1, \dots, 1) \quad \text{and} \quad \widehat{\boldsymbol{\theta}} = \left(0, \frac{2\pi}{N}, \dots, \frac{2\pi(N-1)}{N}\right).$$

We introduce three parameters $\mathtt{t}_\rho, \mathtt{t}_\theta, \mathtt{s} \in [0,1]$, which will allow to control the shape regularity of the polygons in $\mathscr{P}$. The vector $\boldsymbol{\rho} \in \mathbb{R}^N$ for a random E is generated with the following transformation

$$\boldsymbol{\rho} = \widehat{\boldsymbol{\rho}} + \mathtt{t}_\rho \mathbf{X} \tag{5.31}$$

where $\mathbf{X}$ is a random vector uniformly distributed in $(-1,1)^N$.

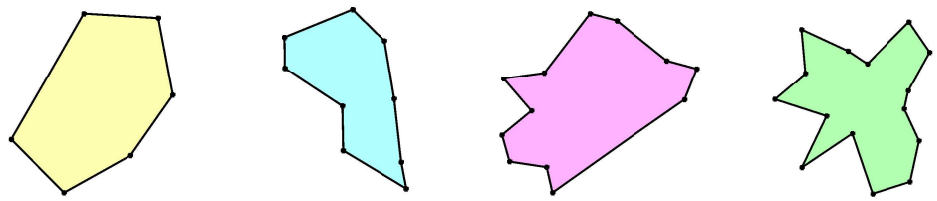


Figure 5.6: Four examples of random star shaped polygons generated by means of Algorithm 5.4.

The vector $\boldsymbol{\theta}$ of angular coordinates is given by the sum of two contributions

$$\boldsymbol{\theta} = \boldsymbol{\gamma} + 2\pi Z. \tag{5.32}$$

The variable Z is simply uniformly distributed in $(0, 1)$, while $\boldsymbol{\gamma} = (\gamma_i)_{i=1}^N$ is generated in several steps. At the beginning, we set $\boldsymbol{\gamma} = \widehat{\boldsymbol{\theta}}$. For fixed index i, we then apply a random perturbation: we generate a uniform random variable $Y \in (-1, 1)$, so that we can write

$$\gamma_i = \gamma_i + \pi \mathtt{t}_\theta Y.$$

We next perform two safety checks to avoid that $\gamma_i \leq \gamma_{i-1}$ and that the new value γ_i is too close (or coincides) with the previous and the subsequent angles. How much γ_i can be close to the the previous and the subsequent angles is controlled by the parameter $\mathtt{s}$. More precisely, we introduce the following two limiting angles

$$\gamma_{\star,i} = \gamma_{i-1} + \frac{2\pi}{N}\mathtt{s} \quad \text{and} \quad \gamma^{\star,i} = 2\pi\frac{N - \mathtt{s}(N - i + 1)}{N}$$

and then we finally choose

$$\gamma_i = \min\{\max\{\gamma_i, \gamma_{\star,i}\}, \gamma^{\star,i}\}.$$

We report in Algorithm 5.4 the entire procedure, while in Figure 5.6 we depict some examples of star shaped polygons generated the presented algorithm.

5.4.2 Construction of the reduced basis

In order to construct our reduced basis spaces, we generate a dataset consisting of 5000 random polygons with N vertices ($N = 4, \ldots, 14$), by making use of

Algorithm 5.4 Generating random star shaped polygons in $\mathscr{P}$

Data:

$\widehat{\boldsymbol{\rho}} = (1, \dots, 1)$

$\widehat{\boldsymbol{\theta}} = \left(0, \frac{2\pi}{N}, \dots, \frac{2\pi(N-1)}{N}\right)$

$\mathtt{t}_\rho, \mathtt{t}_\theta, \mathtt{s} \in [0, 1]$

Output:

$\boldsymbol{\rho}$, $\boldsymbol{\theta}$

Initialize $\boldsymbol{\gamma} = \widehat{\boldsymbol{\theta}}$, $\boldsymbol{\gamma} = (\gamma_i)_{i=1}^N$

Generate $\mathbf{X}$ uniformly in $(-1, 1)^N$

Compute $\boldsymbol{\rho} = \widehat{\boldsymbol{\rho}} + \mathtt{t}_\rho \mathbf{X}$

for $i = 2, \dots, N$ **do**

  Set $\gamma_{\star,i} = \gamma_{i-1} + \frac{2\pi}{N}\mathtt{s}$

  Set $\gamma^{\star,i} = 2\pi\frac{N - \mathtt{s}(N-i+1)}{N}$

  Generate Y uniformly in $(0, 1)$

  $\gamma_i = \gamma_i + \pi \mathtt{t}_\theta Y$

  $\gamma_i = \max\{\gamma_i, \gamma_{\star,i}\}$

  $\gamma_i = \min\{\gamma_i, \gamma^{\star,i}\}$

end

Generate Z uniformly in $(0, 1)$

Compute $\boldsymbol{\theta} = \boldsymbol{\gamma} + 2\pi Z$

---

the algorithm described in Section 5.4.1 (the case $N = 3$ will not be considered since the basis functions of lowest order VEM on triangles coincide with the finite element shape functions, which are known in their analytical form).

Once the required datasets are generated, we are able to construct the reduced basis for each value of N. The offline phase described in Section 5.3.1 is performed on a sample of $R = 300$ polygons $\{E^\ell\}_{\ell=1}^{300}$ randomly selected out of the database. We apply the POD algorithm so that we can construct the ordered sequence of N-tuples $(\widehat{\xi}_1^\ell, \dots, \widehat{\xi}_N^\ell)^\top$ for $\ell = 1, \dots, 60$. Then, the reduced bases with $M \leq 60$ elements are easily obtained by truncation of these tuples.

The snapshots computation is performed on triangulations $\mathcal{T}_\delta^E$ and $\mathcal{T}_\delta$ for E and $\widehat{E}$ respectively with mesh size set to $\delta = \delta^E = 0.01$. In order to map the snap-

shots from $\mathcal{T}_\delta^E$ onto $\mathcal{T}_\delta$ we implement an interpolation rule based on barycentric coordinates. To this aim, we map $\mathcal{T}_\delta^E$ onto $\widehat{E}$ by simply applying the affine map $\mathscr{B}$. Then, given a node $\mathbf{p} \in \mathcal{T}_\delta$, it is easy to identify the triangle $\mathrm{T} \in \mathscr{B}(\mathcal{T}_\delta^E)$ containing $\mathbf{p}$. Denoting by $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$ the vertices of T, we interpolate $\widetilde{\varphi}_j^E$ at $\mathbf{p}$ making use of the following formula

$$I_\delta \widetilde{\varphi}_j^E(\mathbf{p}) \approx \sum_{i=1}^{3} \eta_i \, \widetilde{\varphi}_j^E(\mathbf{v}_i) \qquad j = 1, \dots, N,$$

where η_1, η_2, η_3 are the barycentric coordinates of $\mathbf{p}$ with respect to T.

Finally, for each value of N, we test the reduced basis of dimension M over a set of 500 polygons randomly selected out of our database. In particular, we consider $M = 1, 2, 3, 4, 5, 10, 15, 20, 25, 30, 40, 50, 60$.

5.4.3 Accuracy

In this section, we study the accuracy of the RB method when used to reconstruct a virtual function $u_h \in V_1(E)$. For our analysis, we consider two different situations in which the nonpolynomial part of $V_1(E)$ has two different magnitudes. We have

a) **polynomial DOFs:** the function $u_h \in V_1(E)$ is obtained by interpolation of the polynomial $p(x, y) = x^5 + y^5$. This means that the reconstruction is done imposing as degrees of freedom for $V_1(E)$ the evaluation of p at the vertices of E;

b) **random DOFs:** we reconstruct a function $u_h \in V_1(E)$ whose degrees of freedom are randomly generated in the unit interval with normal distribution; in this case, the nonpolynomial contribution dominates.

Since the polynomial projection $\Pi_1^\nabla u_h$ is always exactly computed, we apply the reduced basis method just to reconstruct the nonpolynomial residual $u_h - \Pi_1^\nabla u_h$. Hence, for each test polygon under consideration, the validation process is carried out with the following four steps:

i) we compute the projection onto polynomials $\Pi_1^\nabla u_h$;

ii) we approximate the nonpolynomial residual $\sigma_M^{\mathrm{rb}} \approx u_h - \Pi_1^\nabla u_h$ by applying the reduced basis method with precision M. Then, we construct $u_M^{\mathrm{rb}} \approx u_h$, where $u_M^{\mathrm{rb}} = \Pi_1^\nabla u_h + \sigma_M^{\mathrm{rb}}$;

iii) we compute the "exact" reconstruction u_h^{fe} by solving Problem 5.1.1 by means of the finite element method on a triangulation $\mathcal{T}_\delta^E$;

iv) we compute the error committed by approximating u_h respectively with the polynomial and RB reconstructions for several values of M. More precisely, we consider $\left\|\Pi_1^\nabla u_h - u_h^{\mathrm{fe}}\right\|_{1,E} / \left\|u_h^{\mathrm{fe}}\right\|_{1,E}$ and $\left\|u_M^{\mathrm{rb}} - u_h^{\mathrm{fe}}\right\|_{1,E} / \left\|u_h^{\mathrm{fe}}\right\|_{1,E}$.

The data obtained at the end of the described procedure is then processed to produce statistical plots of the errors. These plots are collected in Figure 5.7 for the test case a) of polynomial DOFs and in Figure 5.8 for the test case b) of random generated DOFs. For each value of M, we mark the maximum and minimum value by a circle and a diamond respectively. Then, the 95^{th} and 5^{th} percentiles are connected by a straight line on which the mean value is marked by a square.

From both figures, it is clear that in the 95% of cases the reduced basis reconstruction performs better than the projection onto polynomials. In some cases there are instances in which $\Pi_1^\nabla u_h$ produces a better approximation than u_M^{rb}: we consider this results acceptable, since errors have the same order of magnitude. Conversely, if we particularly look at test case b), the use of just two reduced basis functions ($M = 2$) produces already a slight improvement also for the worst case scenario, which makes sense since the nonpolynomial part is dominating. For the best case scenario, the use of a single reduced basis ($M = 1$) produces a gain of one order of magnitude. For the largest degree of precision $M = 60$, the improvement is significant: from Figure 5.7, we gain one order of magnitude with respect to $\Pi_1^\nabla u_h$ even for the worst case scenario, while the order of magnitudes are almost two if we look at the the best case scenario. The mean value behaves accordingly. The same behavior is similarly observed in Figure 5.8 for the test case b).

In Figure 5.9, we visualize some polygons realizing the maximum values of error accordingly to the statistical plots. A common feature of these polygons is the presence of very small edges, which are associated to badly shaped triangles T_i: this may be the cause that weakens the accuracy of the reduced basis method. It would be interesting to understand *a priori* which polygons need a special treatment and further studies will be focused on this aspect.

5.4.4 Computational efficiency

Once we established that the reconstruction of virtual functions via reduced basis method produces accurate results, we study the efficiency of the method. To this

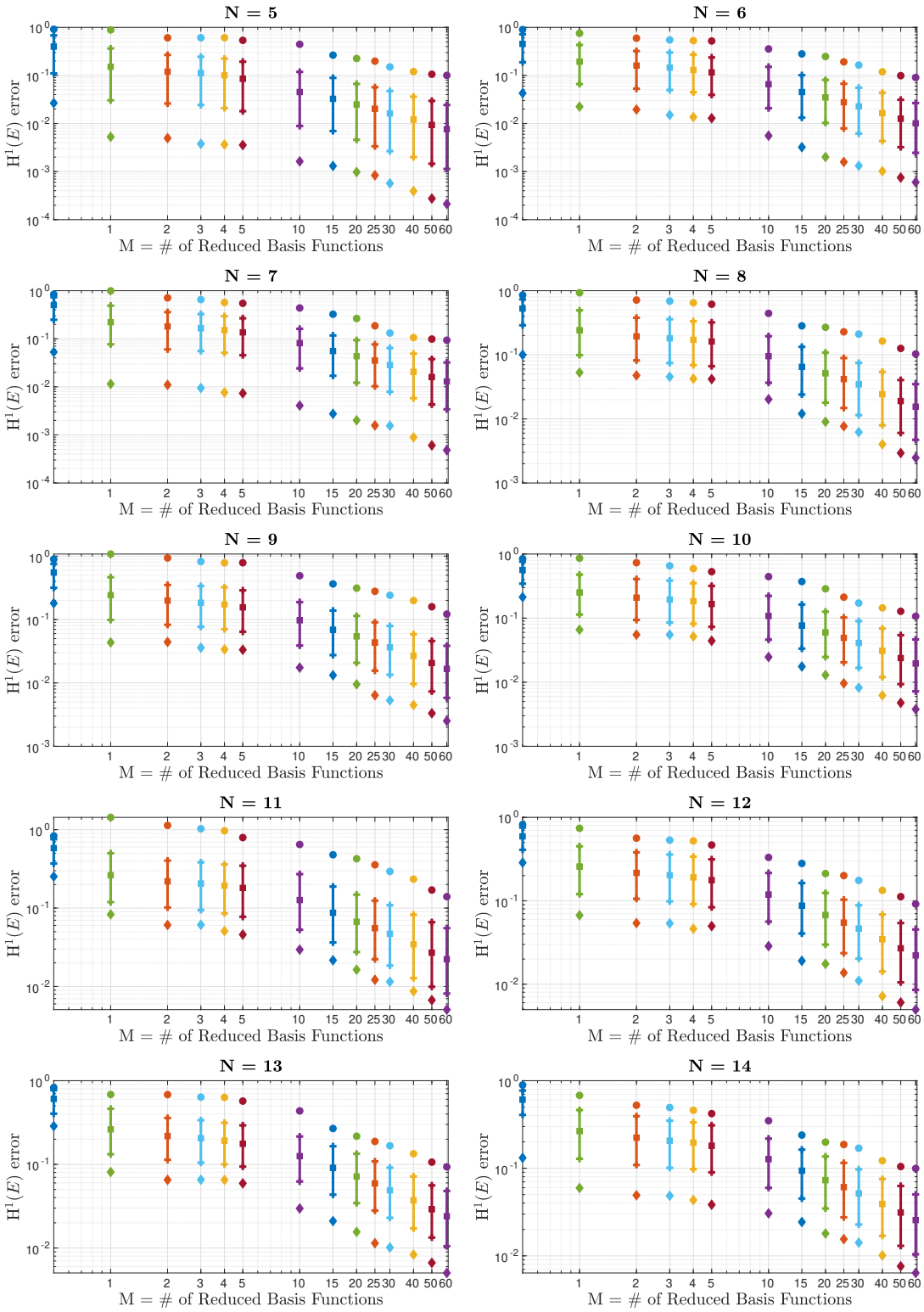


Figure 5.7: Statistical plots of the errors committed by the RB approach for $N = 5, \dots, 14$, increasing the precision M. a) The degrees of freedom are imposed evaluating $p(x, y) = x^5 + y^5$ at the vertices of each E. The first blue data, related to $M = 0$, refers the error between u_h and $\Pi_1^\nabla u_h$. The maximum values are represented with circles, while the minimum values are marked with diamonds. Squares represent mean values. Vertical lines connects the 95^{th} and 5^{th} percentiles

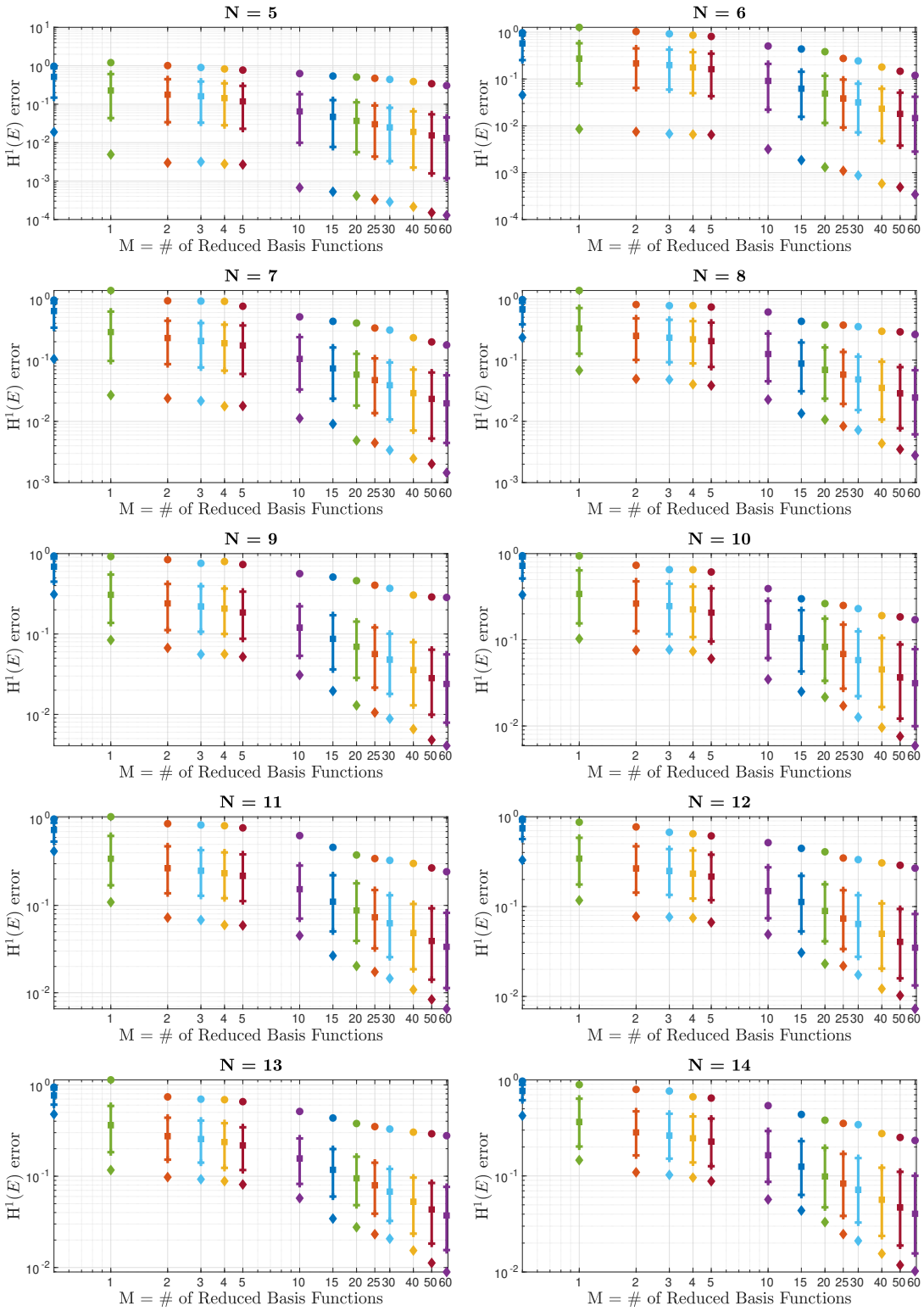
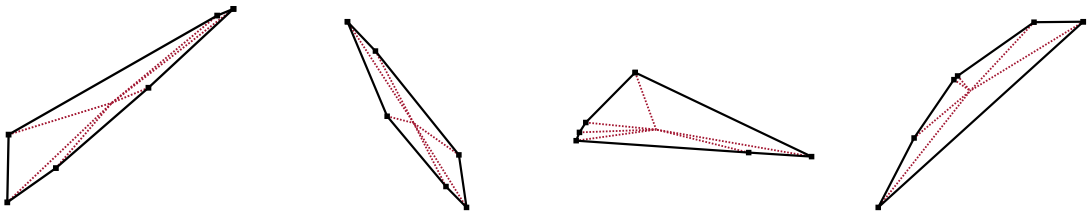
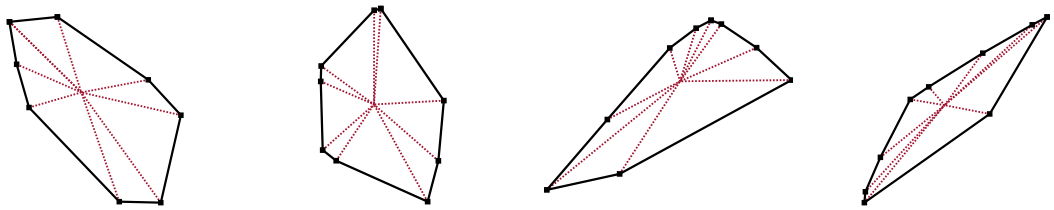


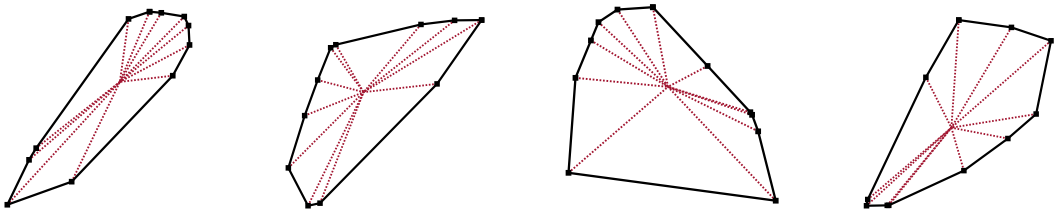
Figure 5.8: Statistical plots of the errors committed by the RB approach for $N = 5, \ldots, 14$, increasing the precision M. b) The degrees of freedom are randomly generated in $(0, 1)$. Same format as Table 5.7



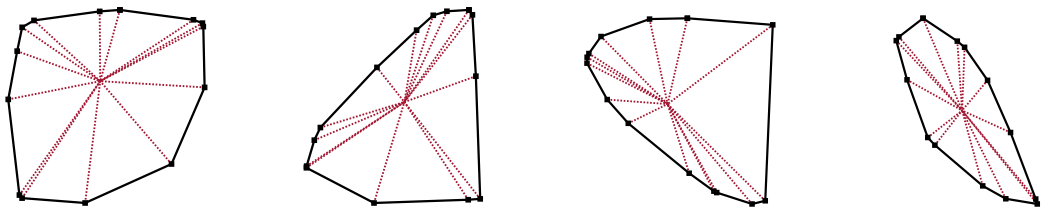
(a) $N = 6$



(b) $N = 9$



(c) $N = 11$



(d) $N = 14$

Figure 5.9: Some polygons realizing the maximum values in Figures 5.7 and 5.8 for $N = 6, 9, 11, 14$. Since some edges are very short, the associated triangles appears to be very badly shaped. Due to this feature, the reduced basis method may underperform.

aim we compare the computational cost deriving from the construction and evaluation of u_M^{rb} for different values of M with both the cost of evaluation of $\Pi_1^\nabla u_h$ and the "exact" reconstruction u_h^{fe}. Of course, the "exact" reconstruction cannot be carried out in practical situations since it requires the solution of N different PDEs per each E: in this case, this technique is taken into account just for academic purpose.

The average times for each phase and each value of N from 4 to 14 are reported in Table 5.1. For the construction of $\Pi_1^\nabla u_h$ we measure the time T_{build} we need to build the stiffness matrix involved in the projection [12] and the time T_{apply} spent to evaluate the projection at the nodes of the triangulation. Also for the computation of the finite element function u_h^{fe} we measure the time of two different phases: we are interested in the time T_{asm} we need to assemble the finite element stiffness matrix and in T_{sol}, which is the time needed to solve the full linear system. In terms of reduced basis approximation, we focus on $M = 1, 5, 30, 60$ and we measure the time T_{asm} and T_{sol} we spend in constructing and solving the reduced linear system during the online phase.

From Table 5.1, we see that in general the computation of u_M^{rb} presents run times which are comparable with the evaluation of $\Pi_1^\nabla u_h$ also when we consider the most expensive case with $M = 60$. Moreover, if, for the sake of comparison, we look at the total time T_{tot} required by the evaluation of u_h^{fe} and we compare it with the total time of the reduced basis procedure, we see that the latter is faster of at least two orders of magnitude. Let us, for instance, focus on the case $N = 14$: if $M = 1$, the time required to evaluate u_M^{rb} is $3.96 \times 10^{-3}\, s$, which is almost half of T_{tot} for computing $\Pi_1^\nabla u_h$, which requires $6.27 \times 10^{-3}\, s$. Moreover, for $M = 60$, a total time of $1.41 \times 10^{-2}\, s$ is spent to build u_M^{rb} which is a very good approximation of u_h^{fe}, whereas the explicit computation of u_h^{fe} requires $2.81\, s$.

At the end, it is evident that we designed an accurate and efficient method to reconstruct virtual functions living in $V_1(E)$ with the same cost of projecting onto polynomials through Π_1^∇.

Remark 5.4.1. *Let us observe that, in practical situations, the online phase of the reduced basis approach is highly parallelizable, since each polygon E can be treated independently from the others. Moreover, thanks to the affine decomposition and the precomputed objects, the online phase does not require to generate or refer to any kind of triangulation in physical or reference elements.*

Comparison in time of post–processing techniques

	$\Pi_1^\nabla u_h$			u_h^{fe}			u_M^{rb}, $M = 60$		
N	$T_{build}(s)$	$T_{apply}(s)$	$T_{tot}(s)$	$T_{asm}(s)$	$T_{sol}(s)$	$T_{tot}(s)$	$T_{asm}(s)$	$T_{sol}(s)$	$T_{tot}(s)$
4	1.50e-3	1.76e-3	3.26e-3	3.61e-1	4.04e-1	7.65e-1	1.53e-3	7.57e-4	2.29e-3
5	1.27e-3	2.31e-3	3.58e-3	4.54e-1	5.50e-1	1.00	2.15e-3	8.35e-4	2.99e-3
6	1.25e-3	2.90e-3	4.14e-3	4.42e-1	6.68e-1	1.21	2.76e-3	8.58e-4	3.63e-3
7	1.26e-3	3.39e-3	4.65e-3	6.99e-1	8.73e-1	1.57	3.96e-3	1.05e-3	5.01e-3
8	1.19e-3	3.48e-3	4.67e-3	7.20e-1	8.95e-1	1.61	4.50e-3	1.05e-3	5.56e-3
9	1.40e-3	3.96e-3	5.36e-3	8.44e-1	1.13	1.97	5.71e-3	1.16e-3	6.87e-3
10	1.37e-3	4.19e-3	5.56e-3	9.34e-1	1.26	2.20	7.00e-3	1.33e-3	8.33e-3
11	1.34e-3	4.15e-3	5.50e-3	9.32e-1	1.34	2.27	8.15e-3	1.34e-3	9.49e-3
12	1.57e-3	4.67e-3	6.26e-3	1.06	1.54	2.60	9.88e-3	1.49e-3	1.14e-2
13	1.47e-3	4.66e-3	6.13e-3	1.06	1.55	2.60	1.08e-2	1.52e-3	1.23e-2
14	1.49e-3	4.78e-3	6.27e-3	1.11	1.70	2.81	1.25e-2	1.63e-3	1.41e-2
	u_M^{rb}, $M = 1$			u_M^{rb}, $M = 5$			u_M^{rb}, $M = 30$		
N	$T_{asm}(s)$	$T_{sol}(s)$	$T_{tot}(s)$	$T_{asm}(s)$	$T_{sol}(s)$	$T_{tot}(s)$	$T_{asm}(s)$	$T_{sol}(s)$	$T_{tot}(s)$
4	6.69e-4	6.58e-5	7.35e-4	7.19e-4	2.22e-4	9.41e-4	9.94e-4	4.11e-4	1.41e-3
5	8.74e-4	5.85e-5	9.33e-4	9.64e-4	2.32e-4	1.20e-3	1.41e-3	4.48e-4	1.86e-3
6	1.05e-3	5.40e-5	1.10e-3	1.18e-3	2.28e-4	1.41e-3	1.76e-3	4.65e-4	2.23e-3
7	1.39e-3	6.19e-5	1.45e-3	1.60e-3	2.60e-4	1.86e-3	2.44e-3	5.37e-4	2.97e-3
8	1.55e-3	5.52e-5	1.60e-3	1.82e-3	2.55e-4	2.07e-3	2.80e-3	5.56e-4	3.36e-3
9	2.04e-3	6.21e-5	2.10e-3	2.32e-3	2.82e-4	2.60e-3	3.52e-3	5.81e-4	4.10e-3
10	2.36e-3	7.01e-5	2.43e-3	2.75e-3	3.03e-4	3.06e-3	4.41e-3	6.69e-4	5.08e-3
11	2.57e-3	6.16e-5	2.64e-3	2.98e-3	2.96e-4	3.27e-3	4.89e-3	6.66e-4	5.56e-3
12	3.13e-3	7.40e-5	3.20e-3	3.63e-3	3.24e-4	3.96e-3	5.94e-3	7.21e-4	6.66e-3
13	3.37e-3	6.85e-5	3.44e-3	3.89e-3	3.11e-4	4.21e-3	6.42e-3	7.18e-4	7.13e-3
14	3.89e-3	7.08e-5	3.96e-3	4.50e-3	3.30e-4	4.83e-3	7.39e-3	7.71e-4	8.16e-3

Table 5.1: Average CPU times required to evaluate $\Pi_1^\nabla u_h$, u_h^{fe} and u_M^{rb} for $M = 1, 5, 30, 60$ on 500 test polygons for each value of N. Each polygon is considered twice since we take into account both polynomial and random DOFs. T_{build} = CPU time to build the projection matrix associated to Π_1^∇; T_{apply} = CPU time to evaluate $\Pi_1^\nabla u_h$ on mesh nodes. For the exact reconstruction u_h^{fe}: T_{asm} = CPU time to assemble the FEM linear system; T_{sol} = CPU time to solve it. For the reduced basis approximations: T_{asm} = CPU time to assemble the linear system of the online phase; T_{sol} = CPU time to solve it. The linear systems are solved with the *backslash* command provided by Matlab. The serial code was ran on a Intel Xeon Gold 6230R core running at 2.10GHz.

5.5 Design of a new VEM stabilization

We previously mentioned that we would like to exploit the reduced basis reconstruction of the virtual functions for the design of stabilization terms able to deal with anisotropic problems. In this section, we describe how to explicitly construct such terms, also discussing the relation between this RB–VEM formulation and conforming polygonal Galerkin methods. We then present some numerical tests to compare the convergence properties of the RB stabilization with respect to the standard *dofi–dofi* and *D–recipe*.

Given a polygon E, we denote by $\widehat{a}^E$ the bilinear form on the reference element $\widehat{E}$ obtained from a^E by change of variable:

$$\widehat{a}^E(v,w) = \sum_{i=1}^{N} \int_{\widehat{\mathrm{T}}_i} \left|\det \mathsf{B}_{E,i}^{-1}\right| \mathsf{B}_{E,i}^{\top} \mathbb{K} \mathsf{B}_{E,i} \nabla \widehat{u} \cdot \widehat{v} \, \mathrm{d}\widehat{\mathbf{x}}.$$

The idea at the basis of our method, is to define S^E as

$$S^E(\varphi_i, \varphi_j) = \widehat{a}^E(\widehat{\varphi}^{\mathrm{rb}}_{M,i}[E], \widehat{\varphi}^{\mathrm{rb}}_{M,j}[E]). \tag{5.33}$$

In (5.30), we saw that the RB approximation of the $V_1(E)$ basis reads

$$\widehat{\varphi}^{\mathrm{rb}}_{M,j}[E] = \widehat{\Lambda}_j + \sum_{\ell=1}^{M} \mathsf{x}_\ell^{E,j} \, \widehat{\xi}_j^\ell$$

so that, using this expansion for S^E, we easily have

$$\begin{aligned} S^E(\varphi_i, \varphi_j) =& \widehat{a}^E(\widehat{\Lambda}_i, \widehat{\Lambda}_j) + \sum_{\ell=1}^{M} \mathsf{x}_\ell^{E,j} \, \widehat{a}^E(\widehat{\xi}_i^\ell, \widehat{\Lambda}_j) \\ &+ \sum_{\ell=1}^{M} \mathsf{x}_\ell^{E,j} \, \widehat{a}^E(\widehat{\Lambda}_i, \widehat{\xi}_j^\ell) + \sum_{\ell,\ell'=1}^{M} \mathsf{x}_\ell^{E,j} \, \widehat{a}^E(\widehat{\xi}_i^{\ell'}, \widehat{\xi}_j^\ell). \end{aligned} \tag{5.34}$$

For efficiently constructing this object, we can exploit again the affine decomposition. We can express the matrix $\left|\det \mathsf{B}_{E,i}^{-1}\right| \mathsf{B}_{E,i}^{\top} \mathbb{K} \mathsf{B}_{E,i}$ as linear combination of $\mathcal{S}^\nu$ for $\nu = 1, \ldots, 4$ defined in Section 5.3.2:

$$\left|\det \mathsf{B}_{E,i}^{-1}\right| \mathsf{B}_{E,i}^{\top} \mathbb{K} \mathsf{B}_{E,i} = \sum_{\nu=1}^{4} \gamma_\nu^i[E] \mathcal{S}^\nu.$$

Then, we obtain the following expression

$$\widehat{a}^E(\widehat{\xi}_j^{\ell'}, \widehat{\xi}_{j'}^{\ell}) = \sum_{i=1}^{N} \sum_{\nu=1}^{4} \gamma_\nu^i[E] \mathsf{A}_i^\nu(j, j', \ell, \ell'), \tag{5.35}$$

which also allow us to efficiently compute the right hand side of (5.34) thanks to the precomputed objects A_i^ν for $i = 1, \dots, N$ and $\nu = 1, \dots, 4$. Therefore, the local bilinear form is constructed as

$$a_h^E(\varphi_i, \varphi_j) = a^E(\Pi_1^\nabla \varphi_i, \Pi_1^\nabla \varphi_j) + S_{\mathrm{rb}}^E(\varphi_i, \varphi_j), \qquad i, j = 1, \dots, N$$

where

$$S_{\mathrm{rb}}^E(\varphi_i, \varphi_j) = \widehat{a}^E(\widehat{\varphi}_{M,i}^{\mathrm{rb}}[E] - \Pi_1^\nabla \varphi_i, \widehat{\varphi}_{M,j}^{\mathrm{rb}}[E] - \Pi_1^\nabla \varphi_j). \tag{5.36}$$

Algorithm 5.5 describes how to construct the RB stabilization term when assembling the virtual elements stiffness matrix.

Algorithm 5.5 Reduced basis stabilization in $V_1(E)$

Data:
$\mathcal{T}_h$: VEM mesh for Ω

for $E \in \mathcal{T}_h$ *with* N *vertices* **do**
Compute $a^E(\Pi_1^\nabla \varphi_i, \Pi_1^\nabla \varphi_j)$ for $i, j = 1, \dots, N$
Go to **Online phase** and compute $\widehat{\varphi}_{M,i}^{\mathrm{rb}}[E]$
Build stabilization $S_{\mathrm{rb}}^E(\varphi_i, \varphi_j) = \widehat{a}^E(\widehat{\varphi}_{M,i}^{\mathrm{rb}}[E] - \Pi_1^\nabla \varphi_i, \widehat{\varphi}_{M,j}^{\mathrm{rb}}[E] - \Pi_1^\nabla \varphi_j)$ on $\widehat{E}$
Set $a_h^E(\varphi_i, \varphi_j) = a^E(\Pi_1^\nabla \varphi_i, \Pi_1^\nabla \varphi_j) + S_{\mathrm{rb}}^E(\varphi_i, \varphi_j)$
end

5.5.1 The RB–stabilized VEM as a fully conforming method

Before analyzing some numerical tests, we remark that we can interpret the method we obtain by applying the RB stabilization with M reduced basis functions as a fully conforming method in a non standard discrete space V_h^{rb}. Given E, let us first define the following space on the boundary

$$\mathbb{B}^\perp(\partial E) = \left\{ v \in \mathrm{H}^{1/2}(\partial E) : \int_{\partial E} v \, \nabla q \cdot \mathbf{n} \, \mathrm{da} = 0, \, \forall q \in \mathcal{P}_1(E), \, \int_{\partial E} v \, \mathrm{da} = 0 \right\}. \tag{5.37}$$

In particular, $\Pi_1^\nabla v = 0$ if and only if $v \in \mathbb{B}^\perp(\partial E)$. Now, considering the RB space $W^{\mathrm{rb}}(E) \subset \mathrm{H}^1(E)$ defined as

$$W^{\mathrm{rb}}(E) = \left\{ v \in span\{\varphi_1^{\mathrm{rb}}, \dots, \varphi_N^{\mathrm{rb}}\} : v_{|\partial E} \in \mathbb{B}^\perp(\partial E) \right\}, \tag{5.38}$$

we can construct $V_h^{\mathrm{rb}}(E)$ as

$$V_h^{\mathrm{rb}}(E) = \mathcal{P}_1(E) \oplus W^{\mathrm{rb}}(E). \tag{5.39}$$

By construction, we have that the polynomials are included in $V_h^{\mathrm{rb}}(E)$, i.e. $\mathcal{P}_1(E) \subset V_h^{\mathrm{rb}}(E)$. Moreover, the restriction of $V_h^{\mathrm{rb}}(E)$ on the boundary ∂E coincides with the boundary space $\mathbb{B}_1(E)$. Finally, notice that discretizing Problem 4.3.1 in

$$V_h^{\mathrm{rb}} = \{ v \in \mathrm{H}_0^1(\Omega) : v_{|E} \in V_h^{\mathrm{rb}}(E) \quad \forall E \in \mathcal{T}_h \}, \tag{5.40}$$

by a Galerkin method, we get the same linear system as for VEM with reduces basis stabilization.

We can also exploit the RB virtual basis reconstruction to design an efficient implementation of the polygonal finite element method in the VEM space. Indeed, by choosing M large enough, we can obtain good approximations of the exact stiffness matrix.

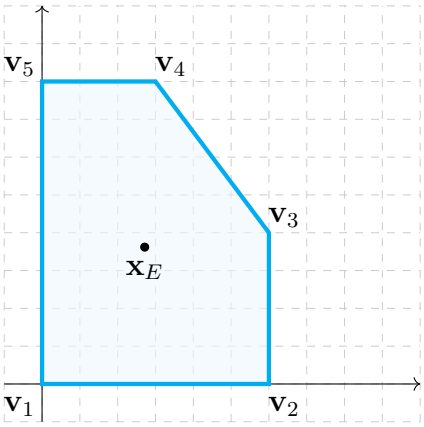
We show here an example: we consider the pentagon depicted in Figure 5.10 and we build its stiffness matrix in four different ways: $\mathsf{K}^{\mathrm{dofi}}$ is the standard matrix in $V_1(E)$ built with *dofi–dofi* stabilization, $\mathsf{K}_1^{\mathrm{rb}}$ and $\mathsf{K}_{60}^{\mathrm{rb}}$ are built applying the procedure in Algorithm 5.5 choosing $M = 1, 60$ respectively and K^{fe} is the stiffness matrix computed with an "exact" computation of the virtual functions.

- Stiffness matrix constructed with *dofi–dofi* stabilization

$$\mathsf{K}^{\mathrm{dofi}} = \begin{bmatrix} +0.7422 & -0.1966 & -0.3412 & -0.2578 & +0.0534 \\ -0.1966 & +0.7422 & -0.3412 & -0.1354 & -0.0690 \\ -0.3412 & -0.3412 & +0.9896 & +0.0364 & -0.3437 \\ -0.2578 & -0.1354 & +0.0364 & +0.8646 & -0.5078 \\ +0.0534 & -0.0690 & -0.3437 & -0.5078 & +0.8672 \end{bmatrix}$$

- Stiffness matrix constructed with RB stabilization, $M = 1$

$$\mathsf{K}_1^{\mathrm{rb}} = \begin{bmatrix} +0.7151 & -0.1463 & -0.3876 & -0.2654 & +0.0843 \\ -0.1463 & +0.6586 & -0.2746 & -0.1013 & -0.1364 \\ -0.3876 & -0.2746 & +0.9495 & -0.0164 & -0.2708 \\ -0.2654 & -0.1013 & -0.0164 & +0.9022 & -0.5190 \\ +0.0843 & -0.1364 & -0.2708 & -0.5190 & +0.8419 \end{bmatrix}$$



$\mathbf{v}_1 = (0,0)$
$\mathbf{v}_2 = (3,0)$
$\mathbf{v}_3 = (3,2)$
$\mathbf{v}_4 = (3/2,4)$
$\mathbf{v}_5 = (0,4)$

$\mathbf{x}_E = (19/14, 38/21)$
$h_E = 5$

Figure 5.10: The pentagon we use to compare several constructions of the stiffness matrix

- Stiffness matrix constructed with RB stabilization, $M = 60$

$$\mathsf{K}_{60}^{\mathrm{rb}} = \begin{bmatrix} +0.7111 & -0.1410 & -0.3902 & -0.2711 & +0.0911 \\ -0.1410 & +0.6488 & -0.2656 & -0.0997 & -0.1425 \\ -0.3902 & -0.2656 & +0.9365 & -0.0085 & -0.2723 \\ -0.2711 & -0.0997 & -0.0085 & +0.8830 & -0.5038 \\ +0.0911 & -0.1425 & -0.2723 & -0.5038 & +0.8274 \end{bmatrix}$$

- Exact stiffness matrix

$$\mathsf{K}^{\mathrm{fe}} = \begin{bmatrix} +0.7110 & -0.1409 & -0.3902 & -0.2711 & +0.0912 \\ -0.1409 & +0.6487 & -0.2654 & -0.0997 & -0.1425 \\ -0.3902 & -0.2654 & +0.9363 & -0.0083 & -0.2723 \\ -0.2711 & -0.0997 & -0.0083 & +0.8828 & -0.5036 \\ +0.0912 & -0.1425 & -0.2723 & -0.5036 & +0.8273 \end{bmatrix}$$

As expected, $\mathsf{K}_{60}^{\mathrm{rb}}$ is a very good approximation of K^{fe}. Of course, as observed in [12] and in Remark 4.4.4, $\mathsf{K}^{\mathrm{dofi}}$ is not close to the exact stiffness K^{fe}, while the cheaply constructed $\mathsf{K}_1^{\mathrm{rb}}$ is something "halfway" between $\mathsf{K}^{\mathrm{dofi}}$ and K^{fe}.

5.5.2 Numerical tests

It is well known that standard formulations of lowest order VEM show poor convergence properties when applied for solving strongly anisotropic problems [32].

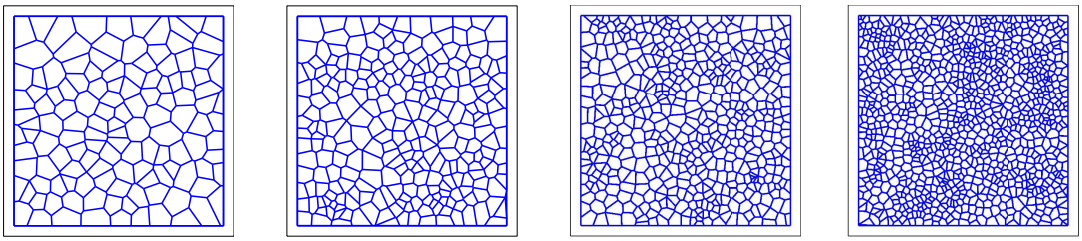


Figure 5.11: Examples of Voronoi meshes on the unit square generated by means of Polymesher.

In this section, we compare the performance of our RB stabilization with classical *dofi–dofi* and *D–recipe*.

We present three numerical tests in which Problem 4.1.1 is solved on the unit square discretized by a sequence of Voronoi meshes generated by means of Polymesher [117]. See Figure 5.11 for some examples. In particular, the first two tests have been used in [32] to compare classical VEM with the stabilization free formulation.

The analysis we conduct in this and in the next section is based on three relative discrete errors. For $\spadesuit = u_h, \Pi_1^\nabla u_h, u_M^{\mathrm{rb}}$ and $\clubsuit = u_h, u_M^{\mathrm{rb}}$, we set

$$\begin{aligned}
\mathtt{err}^\star(\spadesuit) &= \frac{\left(\sum_{E\in\mathcal{T}_h} \|u-\spadesuit\|^2_{\star,E}\right)^{1/2}}{\|u\|_{\star,\Omega}} \qquad \text{for} \quad \star = 0,1 \\
\mathtt{err}^{\mathbb{K}}(\spadesuit) &= \frac{\left(\sum_{E\in\mathcal{T}_h} \left\|\sqrt{\mathbb{K}}\,\nabla\left(u-\spadesuit\right)\right\|^2_{0,E}\right)^{1/2}}{\left\|\sqrt{\mathbb{K}}\,\nabla\, u\right\|_{0,\Omega}} \\
\mathtt{err}^{\infty}(\clubsuit) &= \frac{\max_{\mathbf{x}\in\Omega}|u-\clubsuit|}{\max_{\mathbf{x}\in\Omega}|u|}
\end{aligned} \tag{5.41}$$

In particular, in this section, the discussion is based on the evaluation of $\mathtt{err}^1(\Pi_1^\nabla u_h)$ and $\mathtt{err}^{\mathbb{K}}(\Pi_1^\nabla u_h)$ as usual in virtual element methods.

Test 1

For the first test, we consider a problem in which the diffusivity tensor $\mathbb{K}$ is strongly anisotropic and the solution is characterized by a boundary layer in x-direction next to the right edge of the domain. More precisely, we compute the right hand

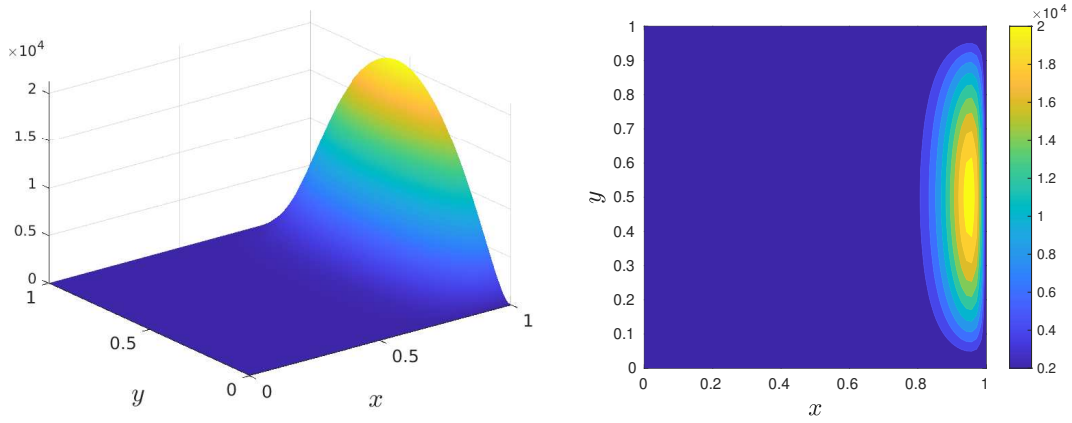


Figure 5.12: Surface and contour plots for the exact solution u for Test 1. We can see a strong boundary layer next to the right edge of $\Omega = [0,1]^2$.

side f accordingly to the following choices:

$$u(x,y) = 10^{-2}xy(1-x)(1-y)(e^{20x}-1), \qquad \mathbb{K} = \begin{bmatrix} 8\times 10^{-3} & 0 \\ 0 & 1 \end{bmatrix}. \tag{5.42}$$

A graphical representation of the solution u is reported in Figure 5.12. The plots depicting the convergence history of the three VEM formulations under consideration are reported in Figure 5.13, while the data related to *dofi–dofi* and RB stabilization ($M = 1$) are collected in Table 5.2. Looking at $\mathtt{err}^1(\Pi_1^\nabla u_h)$, we notice that the method endowed with *dofi–dofi* and *D–recipe* presents the same convergence curves, while using the reduced basis stabilization, in this case with $M = 1$ and $M = 10$, we obtain an improvement in terms of value of the error, whereas the convergence rate is the same as for standard stabilization terms. The same behavior is registered for $\mathtt{err}^{\mathbb{K}}(\Pi_1^\nabla u_h)$ and confirms our idea that a stabilization built using a rough approximation of the virtual shape functions is able to catch the anisotropy of the problem, unlike the classical stabilization terms, which are of isotropic type.

Test 2

For this second test, we increase the anisotropy of the solution. Indeed, we choose a function u highly oscillating in y-direction with the law described in Figure 5.14. More precisely, we choose the right hand side of Problem 4.1 in such a way that

$$u(x,y) = \sin(2\pi x)\sin(\mathtt{z}\pi y), \qquad \mathtt{z} = 80, \tag{5.43}$$

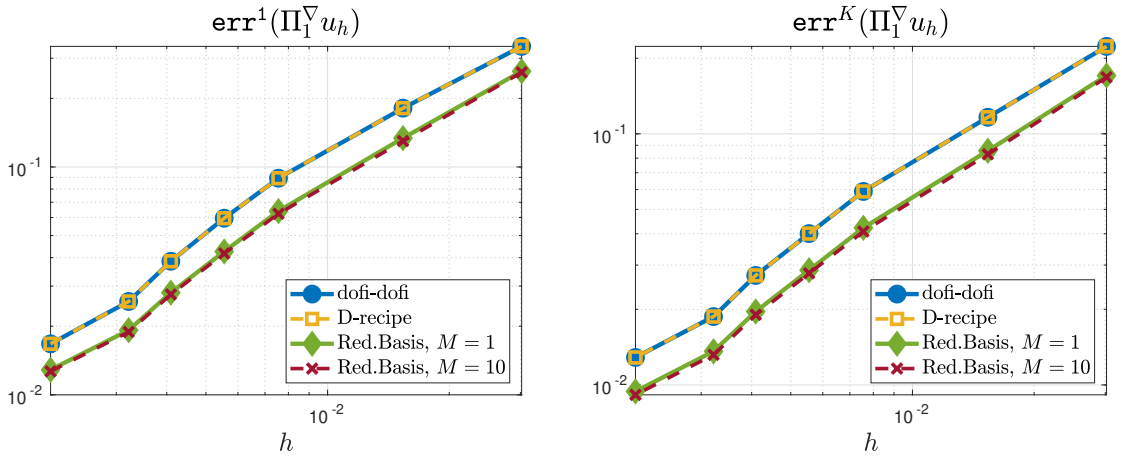


Figure 5.13: Comparison of stabilization terms: convergence plots for Test 1. We represent in blue and yellow the convergence history of *dofi–dofi* and *D–recipe* stabilizations respectively. The green curve is related to the RB stabilization with $M = 1$, while the red curve is for $M = 10$. The RB stabilization catches the anisotropy of the problem slightly improving the method.

Test 1 - Convergence history

	dofi–dofi				Reduced Basis, $M = 1$			
h	$\mathtt{err}^1(\Pi_1^\nabla u_h)$	Rate	$\mathtt{err}^{\mathbb{K}}(\Pi_1^\nabla u_h)$	Rate	$\mathtt{err}^1(\Pi_1^\nabla u_h)$	Rate	$\mathtt{err}^{\mathbb{K}}(\Pi_1^\nabla u_h)$	Rate
3.021e-2	3.382e-1	–	2.229e-1	–	2.623e-1	–	1.707e-1	–
1.536e-2	1.812e-1	0.92	1.164e-1	0.96	1.229e-1	0.99	8.561e-2	1.02
7.569e-3	8.927e-2	0.99	5.895e-2	0.96	6.401e-2	1.04	4.223e-2	0.99
5.548e-3	5.958e-2	1.30	4.006e-2	1.25	4.256e-2	1.32	2.863e-2	1.26
4.094e-3	3.856e-2	1.43	2.729e-2	1.26	2.809e-2	1.37	1.960e-2	1.25
3.219e-3	2.569e-2	1.69	1.871e-2	1.56	1.921e-2	1.58	1.360e-2	1.52
2.063e-3	1.675e-2	0.96	1.286e-2	0.84	1.288e-2	0.90	9.421e-3	0.83

Table 5.2: Convergence history for Test 1: we compare *dofi–dofi* and RB stabilization built with $M = 1$.

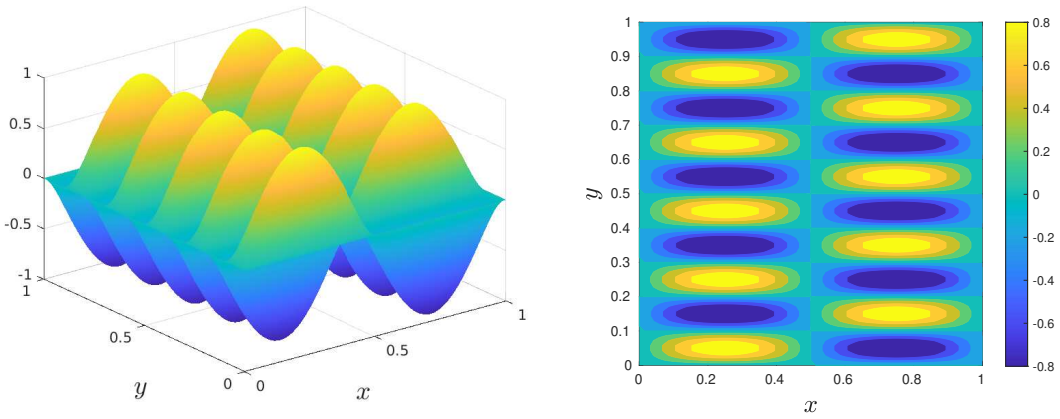


Figure 5.14: Surface and contour plots for the exact solution u for Test 2. The parameter `z` is responsible of the oscillations in y-direction. In this case, we set `z` $= 10$.

is the solution, with diffusivity tensor defined as

$$\mathbb{K} = \begin{bmatrix} 1 & 0 \\ 0 & 6.25 \times 10^{-4} \end{bmatrix}. \tag{5.44}$$

Convergence results are collected in Figure 5.15 and Table 5.3, confirming the general behavior already observed in the previous test. Indeed, looking at both $\texttt{err}^1(\Pi_1^\nabla u_h)$ and $\texttt{err}^{\mathbb{K}}(\Pi_1^\nabla u_h)$, the results of standard stabilization terms are improved by the reduced basis approach. The improvement is evident already with $M = 1$, in this case also in terms of convergence rate. Due to the strong anisotropy, we also check the behavior of the method when $M = 10$ and $M = 60$ are considered. However, the increase in precision is not reflected in the results, which do not show a significant gain with respect to $M = 1$. This confirms that it is possible to obtain good results with little additional computational effort.

Test 3

In this last test, we consider a less regular solution. We choose the right hand side in such a way that the following piecewise continuous u with discontinuous

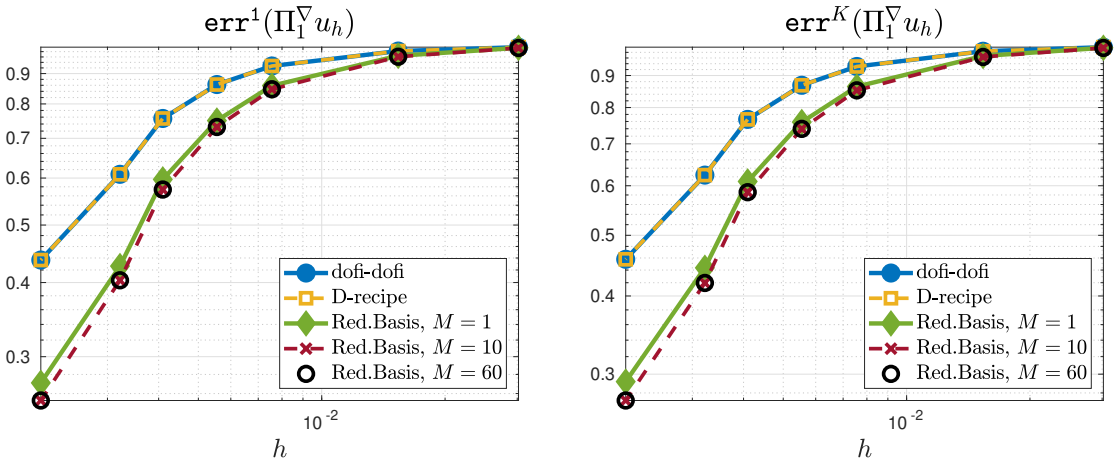


Figure 5.15: Comparison of stabilization terms: convergence plots for Test 2. Same color code used in Figure 5.13. In addition, black circles denote the convergence history of the RB stabilization built with $M = 60$. Also in this case the RB stabilization is able to catch the strong anisotropy of the problem improving the performance obtained by standard stabilization terms. We observe almost coincident results for $M = 1, 10, 60$, therefore the method is effective already in the cheapest and rough case.

Test 2 - Convergence history

	dofi–dofi				Reduced Basis, $M = 1$			
h	$\mathtt{err}^1(\Pi_1^\nabla u_h)$	Rate	$\mathtt{err}^{\mathbb{K}}(\Pi_1^\nabla u_h)$	Rate	$\mathtt{err}^1(\Pi_1^\nabla u_h)$	Rate	$\mathtt{err}^{\mathbb{K}}(\Pi_1^\nabla u_h)$	Rate
3.021e-2	9.971e-1	–	9.989e-1	–	9.945e-1	–	9.966e-1	–
1.536e-2	9.822e-1	0.02	9.841e-1	0.02	9.643e-1	0.05	9.671e-1	0.04
7.569e-3	9.269e-1	0.08	9.304e-1	0.08	8.586e-1	0.16	8.641e-1	0.16
5.548e-3	8.625e-1	0.23	8.683e-1	0.22	7.498e-1	0.44	7.583e-1	0.42
4.094e-3	7.561e-1	0.43	7.659e-1	0.41	5.968e-1	0.75	6.095e-1	0.72
3.219e-3	6.088e-1	0.90	6.239e-1	0.85	4.264e-1	1.40	4.436e-1	1.32
2.063e-3	4.365e-1	0.74	4.578e-1	0.70	2.709e-1	1.02	2.919e-1	0.94

Table 5.3: Convergence history for Test 2: we compare *dofi–dofi* and RB stabilization built with $M = 1$.

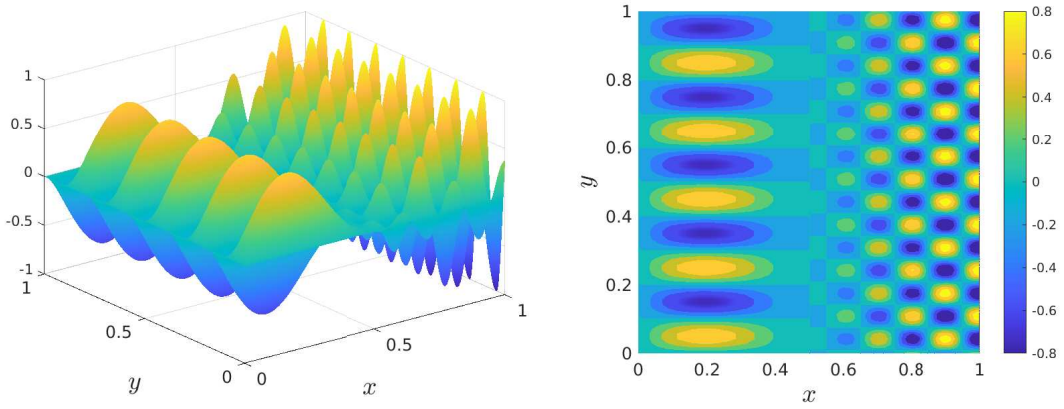


Figure 5.16: Surface and contour plots for the exact solution u for Test 3. The parameter $\mathtt{z}_1$ is responsible of the oscillations in y-direction, while $\mathtt{z}_2$ is responsible of the oscillations in x-direction. In this case, we set $\mathtt{z}_1 = 10$ and $\mathtt{z}_2 = 15$.

gradient along the line $x = 1/2$ is the solution:

$$u(x,y) = \begin{cases} \sin(2\pi x)\sin(\mathtt{z}_1\pi y)\cos(\pi x) & x \leq \frac{1}{2} \\ \cos(\mathtt{z}_1\pi x)\cos(\pi x)\sin(\mathtt{z}_2(\pi - y)\pi) & x > \frac{1}{2} \end{cases}, \qquad \mathtt{z}_1 = 80,\ \mathtt{z}_2 = 30. \tag{5.45}$$

The function u presents strong oscillations in y-direction for $x \leq 1/2$, as well as oscillations in both x and y direction when $x > 1/2$. An example of this functions is depicted in Figure 5.16 for $\mathtt{z}_1 = 10$ and $\mathtt{z}_2 = 15$. Moreover, we consider a non symmetric diffusivity tensor:

$$\mathbb{K} = \begin{bmatrix} 1 & 10^{-2} \\ 5 \times 10^{-3} & 10^{-4} \end{bmatrix}.$$

The behavior of the stabilization terms under consideration is compared in Figure 5.16 and Table 5.4. The convergence history is basically consistent with the results of the previous Test 2. If we look at $\mathtt{err}^1(\Pi_1^\nabla u_h)$, the RB stabilization with $M = 1$ improves the results of *dofi–dofi* and *D–recipe* (they coincide also in this case), both in terms of value of the error and convergence rate. On the contrary, we obtain slightly different results for the energy error $\mathtt{err}^{\mathbb{K}}(\Pi_1^\nabla u_h)$: indeed, all the methods under consideration perform equivalently until the two finest case, where the RB approach presents a slight improvement.

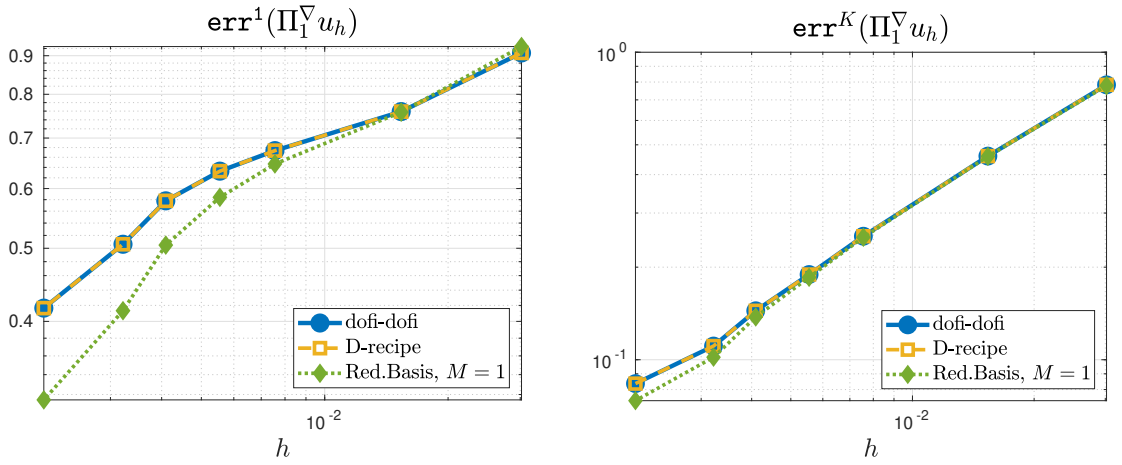


Figure 5.17: Comparison of stabilization terms: convergence plots for Test 3. Same color code used in Figure 5.13 and 5.15. In this case the diffusivity tensor is not symmetric. The RB stabilization clearly improves the results of *dofi–dofi* and *D–recipe* if we look at $\mathtt{err}^1(\Pi_1^\nabla u_h)$. Conversely, if we look at the error in energy norm, all the approaches behave in similar way, with a slight improvement of the RB stabilization in the two finest cases.

Test 3 - Convergence history

	dofi–dofi				Reduced Basis, $M = 1$			
h	$\mathtt{err}^1(\Pi_1^\nabla u_h)$	Rate	$\mathtt{err}^{\mathbb{K}}(\Pi_1^\nabla u_h)$	Rate	$\mathtt{err}^1(\Pi_1^\nabla u_h)$	Rate	$\mathtt{err}^{\mathbb{K}}(\Pi_1^\nabla u_h)$	Rate
3.021e-2	9.077e-1	–	7.833e-1	–	9.255e-1	–	7.787e-1	–
1.536e-2	7.589e-1	0.26	4.595e-1	0.79	7.576e-1	0.30	4.584e-1	0.78
7.569e-3	6.736e-1	0.17	2.522e-1	0.85	6.468e-1	0.22	2.501e-1	0.85
5.548e-3	6.327e-1	0.20	1.893e-1	0.93	5.842e-1	0.33	1.853e-1	0.97
4.094e-3	5.779e-1	0.30	1.443e-1	0.89	5.049e-1	0.48	1.377e-1	0.98
3.219e-3	5.062e-1	0.55	1.108e-1	1.09	4.133e-1	0.83	1.019e-1	1.25
2.063e-3	4.165e-1	0.44	8.364e-2	0.63	3.148e-1	0.61	7.360e-2	0.73

Table 5.4: Convergence history for Test 3: we compare *dofi–dofi* and RB stabilization built with $M = 1$.

5.6 Post-processing of VEM with RB method

The reduced basis machinery we successfully validated and applied for the design of a new stabilization term can be also used to post-process virtual elements solutions. Indeed, by reconstructing the basis functions of $V_1(E)$, we are able to build the actual $u_h \in V_1(E)$, so that a conforming solution is obtained from the degrees of freedom. This can be useful for several purposed: visualization, pointwise evaluation or to compute the actual error in $\mathrm{H}^1(\Omega)$ with respect to a known solution when benchmarking the method in academic research. In Algorithm 5.6, we sketch how to post-process a VEM solution by means of the reduced basis method.

Algorithm 5.6 Reduced basis virtual functions reconstruction

Data:
$\mathcal{T}_h$: VEM mesh for Ω
$\{u_h(\mathbf{v}_j)\}_{j=1,\dots,N}$: DOFs of numerical solution in $E \in \mathcal{T}_h$ (values at vertices)

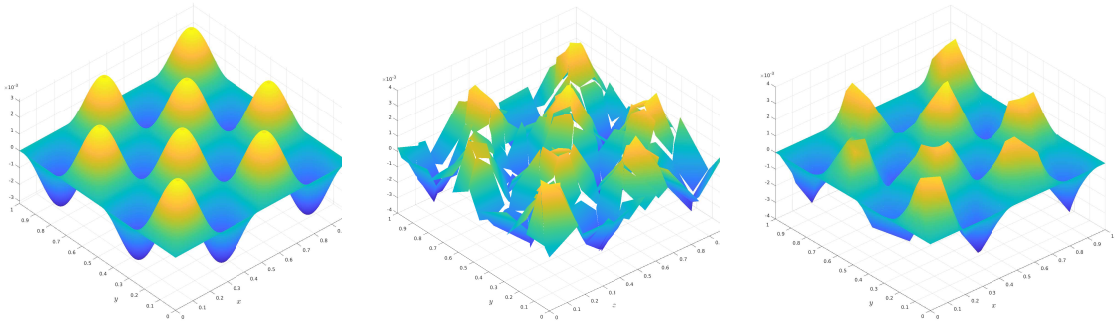
for $E \in \mathcal{T}_h$ **do**
Go to **Online phase** and compute $\widehat{\varphi}^{\mathrm{rb}}_{M,i}[E]$ on $\mathcal{T}_\delta$
Pull back $\varphi^E_j = \widehat{\varphi}^{\mathrm{rb}}_{M,j}[E] \circ \mathscr{B}_E$ on $\mathcal{T}^E_\delta = \mathscr{B}^{-1}(\mathcal{T}_\delta)$ in E
Build $\Pi^\nabla_1 u_h$ evaluating on $\mathcal{T}^E_\delta$
Compute $u^{\mathrm{rb}}_M = \Pi^\nabla_1 u_h + \sum_{j=1}^N (u_h(\mathbf{v}_j) - \Pi^\nabla_1 u_h(\mathbf{v}_j))\, \varphi^E_j$ in E
end

In this section, we show three possible applications: in the first test, we apply the reduced basis method to reconstruct and visualize the conforming solution. In the second test, we carry out the reconstruction along a line, while in the last test we perform a convergence analysis comparing u^{rb}_M, $\Pi^\nabla_1 u_h$ and the "exact" u^{fe}_h.

For all the considered examples, we solve Problem 4.3.1 with $\mathbb{K} = \mathbb{I}$ by means of the lowest order VEM defined on the unit square discretized by Voronoi meshes. In this section, the chosen stabilization is *dofi–dofi*.

5.6.1 Visualization

In this test, $\Omega = [0,1]^2$ is discretized by a relatively coarse Voronoi mesh $\mathcal{T}_h$ consisting of 100 polygons, which is depicted in Figure 5.19a. Problem 4.3.1 is



(a) u (b) $\Pi_1^\nabla u_h$ (c) u_M^{rb}, $M = 1$

Figure 5.18: Visualization of post-processed solution on the Voronoi mesh depicted in Figure 5.19a. (a) The exact solution u. (b) $\Pi_1^\nabla u_h$ computed projecting onto polynomials in each element; the obtained function is discontinuous across the elements. (c) The reconstructed solution u_M^{rb}, computed with a single reduced basis in each element; in this case the solution is conforming in the global virtual element space.

solved considering the right hand side f so that an approximation of the following exact solution is obtained

$$u(x, y) = \frac{1}{32\pi^2} \sin(4\pi x) \sin(4\pi y). \tag{5.46}$$

Once the degrees of freedom are computed, in the standard VEM approach, the solution can be reconstructed by evaluating a projection onto polynomials. This reconstruction is depicted in Figure 5.18b, that shows the evident discontinuities. If for any reason, the final user requires a continuous solutions, we can instead reconstruct the solution u_M^{rb} using even a single reduced basis, i.e. $M = 1$. The post-processed solution thus obtained is represented in Figure 5.18c.

5.6.2 Local reconstruction

We now show how the reduced basis method can be used to locally reconstruct a VEM solution in a subdomain. We solve again the Poisson problem on the Voronoi mesh $\mathcal{T}_h$ drawn in Figure 5.19a and we then reconstruct the solution on a one dimensional grid of the diagonal $y = x$ of the unit square. To this aim we only need to post-process the solution in the polygons E intersecting the considered line.

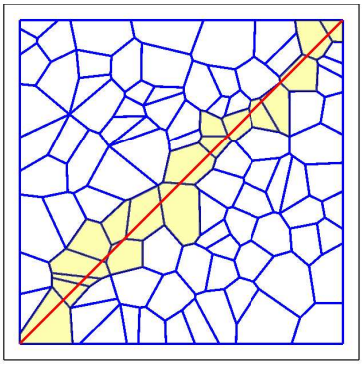
In particular, for our test, we consider

$$\begin{aligned} u(x,y) = & x^3 - xy^2 + yx^2 + x^2 - xy \\ & - x + y - 1 + \sin(5x)\sin(7y) + \log(1 + x^2 + y^4). \end{aligned} \tag{5.47}$$

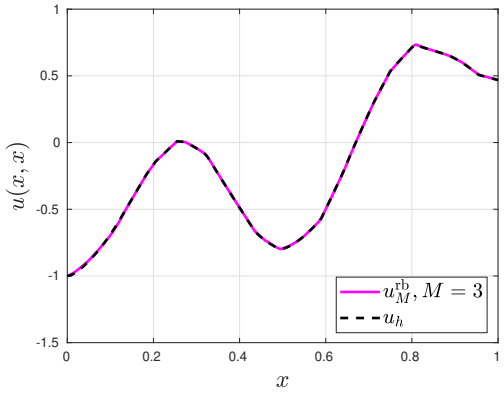
The reduced basis reconstruction is performed in each involved element E using both $M = 1$ and $M = 3$. For the sake of comparison, we also perform a full finite element reconstruction on local triangulations $\mathcal{T}_\delta^E$ with size $\delta^E = h_E/100$. The results are reported in Figure 5.19. In particular, in Figure 5.19b, we plot in magenta the RB reconstruction u_M^{rb} with $M = 3$ and in black u_h^{fe}: notice that, although we use few reduced basis functions, the two reconstructions coincide. In Figure 5.19c, we plot again u_h^{fe}, and compare it with the projection $\Pi_1^\nabla u_h$ (denoted by blue segments): as already observed in the previous visualization test, also $\Pi_1^\nabla u_h$ is a good approximation, but characterized by discontinuity across the elements. Finally, in Figure 5.19d, we zoom in the interval $[0.4, 0.6]$ to compare all the post-processed solutions, taking also into account u_M^{rb} with $M = 1$, which is denoted by a green line.

5.6.3 Convergence test

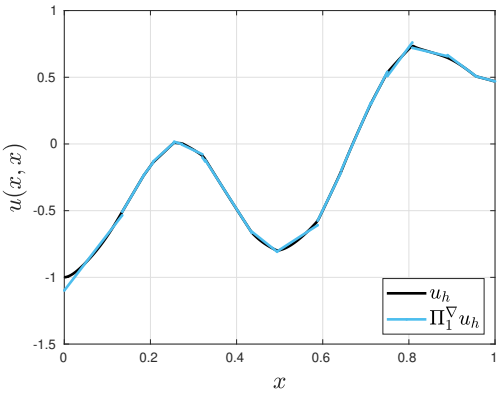
As mentioned before, in this test we carry out a convergence analysis. We solve again the Poisson problem with exact solution u defined in (5.46) for the visualization test. This time, the domain is discretized with a sequence $\{\mathcal{T}_h\}_h$ of seven Voronoi meshes with decreasing size and the post-processing is performed element by element in the entire Ω, with a procedure similar to the one adopted in the previous reconstruction test. In particular, we compare the convergence history of u_h^{fe}, $\Pi_1^\nabla u_h$ and u_M^{rb}, $M = 1$ in terms of $\mathtt{err}^0(\diamond), \mathtt{err}^1(\diamond), \mathtt{err}^\infty(\diamond)$ and the results are collected in Figure 5.20. The reduced basis reconstruction, represented with an orange line, behaves as the "exact" u_h^{fe} (black line) in terms of all the considered error indicators. These conforming reconstructions slightly improve the convergence properties of $\Pi_1^\nabla u_h$. Moreover, if we look at $\mathtt{err}^\infty(\diamond)$, the results are in line with the previous test since the reduced basis reconstruction is optimal also from a pointwise perspective.



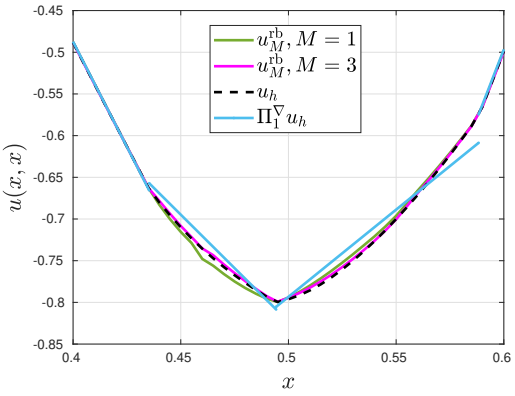
(a) Geometry



(b) u_M^{rb} vs u_h



(c) $\Pi_1^\nabla u_h$ vs u_h



(d) Zoom

Figure 5.19: Reconstruction in subdomain. (a) Geometric configuration for visualization and reconstruction tests. The reconstruction in subdomain is performed on the red line computing the virtual functions in the yellow elements. (b) Comparison between the "exact" reconstruction u_h^{fe} (black line) and the reduced basis approximation $u_M^{\mathrm{rb}}, M = 3$ (magenta line). (c) Comparison between u_h^{fe} and the projection $\Pi_1^\nabla u_h$ (blue line). (d) Zoom on the interval $[0.4, 0.6]$ comparing u_h^{fe}, $\Pi_1^\nabla u_h$, $u_M^{\mathrm{rb}}, M = 3$ and in addition $u_M^{\mathrm{rb}}, M = 1$ (green line). The RB approach produces very good approximations, basically coincident with u_h^{fe} even if small values of M are considered.

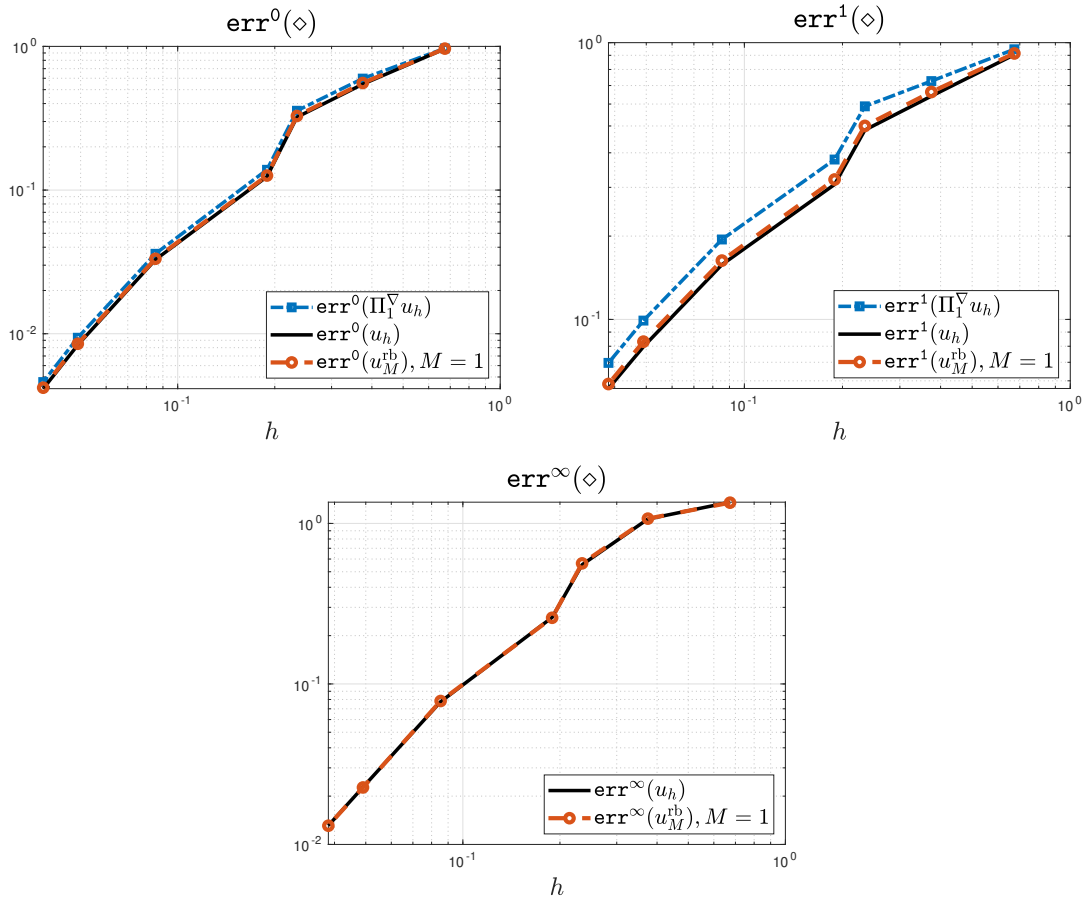


Figure 5.20: Convergence test. Convergence analysis comparing the reduced basis approximation u_M^{rb} computed with $M = 1$ (orange line) with the standard polynomial projection $\Pi_1^\nabla u_h$ (blue line) and the "exact" reconstruction u_h^{fe} (black line). We observe that u_M^{rb} and u_h^{fe} behave equivalently.

Bibliography

[1] B. Ahmad, A. Alsaedi, F. Brezzi, L. D. Marini, and A. Russo. Equivalent projectors for virtual element methods. *Computers & Mathematics with Applications*, 66(3):376–391, 2013.

[2] P. F. Antonietti, L. Beirão da Veiga, and G. Manzini. *The virtual element method and its applications*, volume 31. Springer Nature, 2022.

[3] P. F. Antonietti, L. Beirão da Veiga, D. Mora, and M. Verani. A stream virtual element formulation of the Stokes problem on polygonal meshes. *SIAM Journal on Numerical Analysis*, 52(1):386–404, 2014.

[4] M. Attene, S. Biasotti, S. Bertoluzza, D. Cabiddu, M. Livesu, G. Patanè, M. Pennacchio, D. Prada, and M. Spagnuolo. Benchmarking the geometrical robustness of a virtual element Poisson solver. *Mathematics and Computers in Simulation*, 190:1392–1414, 2021.

[5] N. A. Barnafi, F. Dassi, and S. Scacchi. Parallel block preconditioners for virtual element discretizations of the time-dependent Maxwell equations. *Journal of Computational Physics*, 478:111970, 2023.

[6] L. Beirão da Veiga, F. Brezzi, A. Cangiani, G. Manzini, L. D. Marini, and A. Russo. Basic principles of virtual element methods. *Mathematical Models and Methods in Applied Sciences*, 23(01):199–214, 2013.

[7] L. Beirão da Veiga, F. Brezzi, F. Dassi, L. Marini, and A. Russo. Lowest order virtual element approximation of magnetostatic problems. *Computer Methods in Applied Mechanics and Engineering*, 332:343–362, 2018.

[8] L. Beirão da Veiga, F. Brezzi, F. Dassi, L. D. Marini, and A. Russo. Virtual element approximation of 2D magnetostatic problems. *Computer Methods in Applied Mechanics and Engineering*, 327:173–195, 2017.

[9] L. Beirão da Veiga, F. Brezzi, F. Dassi, L. D. Marini, and A. Russo. A family of three-dimensional virtual elements with applications to magnetostatics. *SIAM Journal on Numerical Analysis*, 56(5):2940–2962, 2018.

[10] L. Beirão Da Veiga, F. Brezzi, F. Dassi, L. D. Marini, and A. Russo. Serendipity virtual elements for general elliptic equations in three dimensions. *Chinese Annals of Mathematics, Series B*, 39(2):315–334, 2018.

[11] L. Beirão da Veiga, F. Brezzi, and L. D. Marini. Virtual elements for linear elasticity problems. *SIAM Journal on Numerical Analysis*, 51(2):794–812, 2013.

[12] L. Beirão da Veiga, F. Brezzi, L. D. Marini, and A. Russo. The hitchhiker's guide to the virtual element method. *Mathematical models and methods in applied sciences*, 24(08):1541–1573, 2014.

[13] L. Beirão da Veiga, F. Brezzi, L. D. Marini, and A. Russo. Mixed virtual element methods for general second order elliptic problems on polygonal meshes. *ESAIM: Mathematical Modelling and Numerical Analysis-Modélisation Mathématique et Analyse Numérique*, 50(3):727–747, 2016.

[14] L. Beirão da Veiga, F. Brezzi, L. D. Marini, and A. Russo. Serendipity nodal VEM spaces. *Computers & Fluids*, 141:2–12, 2016.

[15] L. Beirão da Veiga, F. Brezzi, L. D. Marini, and A. Russo. Virtual element method for general second-order elliptic problems on polygonal meshes. *Mathematical Models and Methods in Applied Sciences*, 26(04):729–750, 2016.

[16] L. Beirão da Veiga, F. Brezzi, L. D. Marini, and A. Russo. Serendipity face and edge VEM spaces. *Rendiconti Lincei*, 28(1):143–180, 2017.

[17] L. Beirão da Veiga, F. Brezzi, L. D. Marini, and A. Russo. The virtual element method. *Acta Numerica*, 32:123–202, 2023.

[18] L. Beirão da Veiga, A. Chernov, L. Mascotto, and A. Russo. Basic principles of hp virtual elements on quasiuniform meshes. *Mathematical Models and Methods in Applied Sciences*, 26(08):1567–1598, 2016.

[19] L. Beirão da Veiga, A. Chernov, L. Mascotto, and A. Russo. Exponential convergence of the hp virtual element method in presence of corner singularities. *Numerische Mathematik*, 138(3):581–613, 2018.

[20] L. Beirão da Veiga, V. Gyrya, K. Lipnikov, and G. Manzini. Mimetic finite difference method for the Stokes problem on polygonal meshes. *Journal of computational physics*, 228(19):7215–7232, 2009.

[21] L. Beirão da Veiga, C. Lovadina, and A. Russo. Stability analysis for the virtual element method. *Mathematical Models and Methods in Applied Sciences*, 27(13):2557–2594, 2017.

[22] L. Beirão da Veiga, C. Lovadina, and G. Vacca. Divergence free virtual elements for the Stokes problem on polygonal meshes. *ESAIM: Mathematical Modelling and Numerical Analysis*, 51(2):509–535, 2017.

[23] L. Beirão da Veiga, G. Manzini, and L. Mascotto. A posteriori error estimation and adaptivity in hp virtual elements. *Numerische Mathematik*, 143(1):139–175, 2019.

[24] L. Beirão da Veiga and L. Mascotto. Interpolation and stability properties of low-order face and edge virtual element spaces. *IMA Journal of Numerical Analysis*, 43(2):828–851, 2023.

[25] L. Beirão da Veiga and L. Mascotto. Stability and interpolation properties of serendipity nodal virtual elements. *Applied Mathematics Letters*, 142:108639, 2023.

[26] L. Beirão da Veiga, L. Mascotto, and J. Meng. Interpolation and stability estimates for edge and face virtual elements of general order. *Mathematical Models and Methods in Applied Sciences*, 32(08):1589–1631, 2022.

[27] L. Beirão da Veiga, D. Mora, and G. Vacca. The Stokes complex for virtual elements with application to Navier–Stokes flows. *Journal of Scientific Computing*, 81:990–1018, 2019.

[28] L. Beirão da Veiga and G. Vacca. Sharper error estimates for virtual elements and a bubble-enriched version. *SIAM Journal on Numerical Analysis*, 60(4):1853–1878, 2022.

[29] M. F. Benedetto, S. Berrone, A. Borio, S. Pieraccini, and S. Scialo. A hybrid mortar virtual element method for discrete fracture network simulations. *Journal of Computational Physics*, 306:148–166, 2016.

[30] S. Berrone and A. Borio. A residual a posteriori error estimate for the virtual element method. *Mathematical Models and Methods in Applied Sciences*, 27(08):1423–1458, 2017.

[31] S. Berrone, A. Borio, and F. Marcon. Lowest order stabilization free virtual element method for the Poisson equation. *arXiv preprint arXiv:2103.16896*, 2021.

[32] S. Berrone, A. Borio, and F. Marcon. Comparison of standard and stabilization free virtual elements on anisotropic elliptic problems. *Applied Mathematics Letters*, 129:107971, 2022.

[33] S. Berrone, A. Borio, F. Marcon, and G. Teora. A first-order stabilization-free virtual element method. *Applied Mathematics Letters*, 142:108641, 2023.

[34] S. Berrone and M. Busetto. A virtual element method for the two-phase flow of immiscible fluids in porous media. *Computational Geosciences*, 26(1):195–216, 2022.

[35] S. Berrone, M. Busetto, and F. Vicini. Virtual element simulation of two-phase flow of immiscible fluids in discrete fracture networks. *Journal of Computational Physics*, 473:111735, 2023.

[36] S. Berrone and A. Raeli. Efficient partitioning of conforming virtual element discretizations for large scale discrete fracture network flow parallel solvers. *Engineering Geology*, 306:106747, 2022.

[37] S. Bertoluzza, G. Manzini, M. Pennacchio, and D. Prada. Stabilization of the nonconforming virtual element method. *Computers & Mathematics with Applications*, 116:25–47, 2022.

[38] S. Bertoluzza, M. Montardini, M. Pennacchio, and D. Prada. The virtual element method on image-based domain approximations. *arXiv preprint arXiv:2206.03449*, 2022.

[39] S. Bertoluzza, M. Pennacchio, and D. Prada. BDDC and FETI-DP for the virtual element method. *Calcolo*, 54:1565–1593, 2017.

[40] S. Bertoluzza, M. Pennacchio, and D. Prada. High order VEM on curved domains. *Rendiconti Lincei*, 30(2):391–412, 2019.

[41] S. Bertoluzza, M. Pennacchio, and D. Prada. FETI-DP for the three dimensional virtual element method. *SIAM Journal on Numerical Analysis*, 58(3):1556–1591, 2020.

[42] S. Bertoluzza, M. Pennacchio, and D. Prada. Interior estimates for the virtual element method. *arXiv preprint arXiv:2204.09955*, 2022.

[43] S. Bertoluzza, M. Pennacchio, and D. Prada. Weakly imposed Dirichlet boundary conditions for 2D and 3D virtual elements. *Computer Methods in Applied Mechanics and Engineering*, 400:115454, 2022.

[44] T. Bevilacqua, F. Dassi, S. Zampini, and S. Scacchi. BDDC preconditioners for virtual element approximations of the three-dimensional Stokes equations. *arXiv preprint arXiv:2304.09770*, 2023.

[45] T. Bevilacqua and S. Scacchi. BDDC preconditioners for divergence free virtual element discretizations of the Stokes equations. *Journal of Scientific Computing*, 92(2):63, 2022.

[46] D. Boffi, F. Gardini, and L. Gastaldi. Approximation of PDE eigenvalue problems involving parameter dependent matrices. *Calcolo*, 57(4):41, 2020.

[47] D. Boffi, F. Gardini, and L. Gastaldi. Virtual element approximation of eigenvalue problems. In *The Virtual Element Method and its Applications*, pages 275–320. Springer, 2022.

[48] S. C. Brenner, Q. Guan, and L.-Y. Sung. Some estimates for virtual element methods. *Computational Methods in Applied Mathematics*, 17(4):553–574, 2017.

[49] S. C. Brenner and L. R. Scott. *The mathematical theory of finite element methods*, volume 3. Springer, 2008.

[50] S. C. Brenner and L.-Y. Sung. Virtual element methods on meshes with small edges or faces. *Mathematical Models and Methods in Applied Sciences*, 28(07):1291–1336, 2018.

[51] H. Brézis. *Functional analysis, Sobolev spaces and partial differential equations*, volume 2. Springer, 2011.

[52] F. Brezzi, R. S. Falk, and L. D. Marini. Basic principles of mixed virtual element methods. *ESAIM: Mathematical Modelling and Numerical Analysis*, 48(4):1227–1240, 2014.

[53] E. Cáceres, G. N. Gatica, and F. A. Sequeira. A mixed virtual element method for the Brinkman problem. *Mathematical Models and Methods in Applied Sciences*, 27(04):707–743, 2017.

[54] A. Cangiani, E. H. Georgoulis, and P. Houston. hp-version discontinuous Galerkin methods on polygonal and polyhedral meshes. *Mathematical Models and Methods in Applied Sciences*, 24(10):2009–2041, 2014.

[55] A. Cangiani, E. H. Georgoulis, T. Pryer, and O. J. Sutton. A posteriori error estimates for the virtual element method. *Numerische mathematik*, 137:857–893, 2017.

[56] A. Cangiani and M. Munar. A posteriori error estimates for mixed virtual element methods. *arXiv preprint arXiv:1904.10054*, 2019.

[57] L. Chen and J. Huang. Some error analysis on virtual element methods. *Calcolo*, 55:1–23, 2018.

[58] H. Chi, L. Beirão da Veiga, and G. H. Paulino. A simple and effective gradient recovery scheme and a posteriori error estimator for the virtual element method (VEM). *Computer Methods in Applied Mechanics and Engineering*, 347:21–58, 2019.

[59] P. G. Ciarlet. *The finite element method for elliptic problems*. SIAM, 2002.

[60] M. Cihan, B. Hudobivnik, J. Korelc, and P. Wriggers. A virtual element method for 3D contact problems with non-conforming meshes. *Computer Methods in Applied Mechanics and Engineering*, 402:115385, 2022.

[61] B. Cockburn, D. A. Di Pietro, and A. Ern. Bridging the hybrid high-order and hybridizable discontinuous Galerkin methods. *ESAIM: Mathematical Modelling and Numerical Analysis*, 50(3):635–650, 2016.

[62] B. Cockburn, J. Gopalakrishnan, and R. Lazarov. Unified hybridization of discontinuous Galerkin, mixed, and continuous Galerkin methods for second order elliptic problems. *SIAM Journal on Numerical Analysis*, 47(2):1319–1365, 2009.

[63] B. Cockburn, G. E. Karniadakis, and C.-W. Shu. The development of discontinuous Galerkin methods. In *Discontinuous Galerkin methods: theory, computation and applications*, pages 3–50. Springer, 2000.

[64] F. Credali, S. Bertoluzza, and D. Prada. Reduced basis stabilization and post-processing for the virtual element method. *arXiv preprint arXiv:2310.00625*, 2023.

[65] F. Dassi, A. Fumagalli, D. Losapio, S. Scialò, A. Scotti, and G. Vacca. The mixed virtual element method on curved edges in two dimensions. *Computer Methods in Applied Mechanics and Engineering*, 386:114098, 2021.

[66] F. Dassi, A. Fumagalli, A. Scotti, and G. Vacca. Bend 3D mixed virtual element method for Darcy problems. *Computers & Mathematics with Applications*, 119:1–12, 2022.

[67] F. Dassi, C. Lovadina, and M. Visinoni. Hybridization of the virtual element method for linear elasticity problems. *Mathematical Models and Methods in Applied Sciences*, 31(14):2979–3008, 2021.

[68] F. Dassi and S. Scacchi. Parallel block preconditioners for three-dimensional virtual element discretizations of saddle-point problems. *Computer Methods in Applied Mechanics and Engineering*, 372:113424, 2020.

[69] F. Dassi and S. Scacchi. Parallel solvers for virtual element discretizations of elliptic equations in mixed form. *Computers & Mathematics with Applications*, 79(7):1972–1989, 2020.

[70] F. Dassi and G. Vacca. Bricks for the mixed high-order virtual element method: projectors and differential operators. *Applied Numerical Mathematics*, 155:140–159, 2020.

[71] F. Dassi, S. Zampini, and S. Scacchi. Robust and scalable adaptive BDDC preconditioners for virtual element discretizations of elliptic partial differential equations in mixed form. *Computer Methods in Applied Mechanics and Engineering*, 391:114620, 2022.

[72] B. A. de Dios, K. Lipnikov, and G. Manzini. The nonconforming virtual element method. *ESAIM: Mathematical Modelling and Numerical Analysis*, 50(3):879–904, 2016.

[73] A. Dedner and A. Hodson. A framework for implementing general virtual element spaces. *arXiv preprint arXiv:2208.08978*, 2022.

[74] D. A. Di Pietro and A. Ern. *Mathematical aspects of discontinuous Galerkin methods*, volume 69. Springer Science & Business Media, 2011.

[75] D. A. Di Pietro and A. Ern. A hybrid high-order locking-free method for linear elasticity on general meshes. *Computer Methods in Applied Mechanics and Engineering*, 283:1–21, 2015.

[76] D. A. Di Pietro and A. Ern. Hybrid high-order methods for variable-diffusion problems on general meshes. *Comptes Rendus Mathématique*, 353(1):31–34, 2015.

[77] D. A. Di Pietro, A. Ern, and S. Lemaire. An arbitrary-order and compact-stencil discretization of diffusion on general meshes based on local reconstruction operators. *Computational Methods in Applied Mathematics*, 14(4):461–472, 2014.

[78] L. C. Evans. *Partial differential equations*, volume 19. American Mathematical Society, 2022.

[79] T.-P. Fries and T. Belytschko. The extended/generalized finite element method: an overview of the method and its applications. *International journal for numerical methods in engineering*, 84(3):253–304, 2010.

[80] A. L. Gain, C. Talischi, and G. H. Paulino. On the virtual element method for three-dimensional linear elasticity problems on arbitrary polyhedral meshes. *Computer Methods in Applied Mechanics and Engineering*, 282:132–160, 2014.

[81] F. Gardini, G. Manzini, and G. Vacca. The nonconforming virtual element method for eigenvalue problems. *ESAIM: Mathematical Modelling and Numerical Analysis*, 53(3):749–774, 2019.

[82] F. Gardini and G. Vacca. Virtual element method for second-order elliptic eigenvalue problems. *IMA Journal of Numerical Analysis*, 38(4):2026–2054, 2018.

[83] G. N. Gatica, M. Munar, and F. A. Sequeira. A mixed virtual element method for a nonlinear Brinkman model of porous media flow. *Calcolo*, 55:1–36, 2018.

[84] S. Gómez, L. Mascotto, A. Moiola, and I. Perugia. Space-time virtual elements for the heat equation. *arXiv preprint arXiv:2212.05343*, 2022.

[85] S. Gómez, L. Mascotto, and I. Perugia. Design and performance of a space–time virtual element method for the heat equation on prismatic meshes. *Comput. Methods Appl. Mech. Engrg.*, 418:Paper No. 116491, 2024.

[86] H. Guo, C. Xie, and R. Zhao. Superconvergent gradient recovery for virtual element methods. *Mathematical Models and Methods in Applied Sciences*, 29(11):2007–2031, 2019.

[87] W. Hackbusch and S. A. Sauter. Composite finite elements for problems containing small geometric details: Part II: Implementation and numerical results. *Computing and Visualization in Science*, 1(1):15–25, 1997.

[88] W. Hackbusch and S. A. Sauter. Composite finite elements for the approximation of pdes on domains with complicated micro-structures. *Numerische Mathematik*, 75:447–472, 1997.

[89] C. Herrera, R. Corrales-Barquero, J. Arroyo-Esquivel, and J. G. Calvo. A numerical implementation for the high-order 2D virtual element method in MATLAB. *Numerical Algorithms*, 92(3):1707–1721, 2023.

[90] J. S. Hesthaven, G. Rozza, and B. Stamm. *Certified reduced basis methods for parametrized partial differential equations*, volume 590. Springer, 2016.

[91] J. S. Hesthaven and T. Warburton. *Nodal discontinuous Galerkin methods: algorithms, analysis, and applications*. Springer Science & Business Media, 2007.

[92] Y. Kuznetsov and S. Repin. New mixed finite element method on polygonal and polyhedral meshes. 2003.

[93] F. Lepe, D. Mora, G. Rivera, and I. Velásquez. A virtual element method for the Steklov eigenvalue problem allowing small edges. *Journal of Scientific Computing*, 88:1–21, 2021.

[94] K. Lipnikov, G. Manzini, and M. Shashkov. Mimetic finite difference method. *Journal of Computational Physics*, 257:1163–1227, 2014.

[95] X. Liu, R. Li, and Z. Chen. A virtual element method for the coupled Stokes–Darcy problem with the Beaver–Joseph–Saffman interface condition. *Calcolo*, 56(4):48, 2019.

[96] G. Manzini and A. Mazzia. Conforming virtual element approximations of the two-dimensional Stokes problem. *Applied Numerical Mathematics*, 181:176–203, 2022.

[97] G. Manzini, A. Russo, and N. Sukumar. New perspectives on polygonal and polyhedral finite element methods. *Mathematical Models and Methods in Applied Sciences*, 24(08):1665–1699, 2014.

[98] L. Mascotto. Ill-conditioning in the virtual element method: Stabilizations and bases. *Numerical Methods for Partial Differential Equations*, 34(4):1258–1281, 2018.

[99] L. Mascotto. The role of stabilization in the virtual element method: A survey. *Comput. Math. Appl.*, 151:244–251, 2023.

[100] L. Mascotto, I. Perugia, and A. Pichler. Non-conforming harmonic virtual element method: h-and p-versions. *Journal of Scientific Computing*, 77(3):1874–1908, 2018.

[101] L. Mascotto, I. Perugia, and A. Pichler. A nonconforming Trefftz virtual element method for the Helmholtz problem. *Mathematical Models and Methods in Applied Sciences*, 29(09):1619–1656, 2019.

[102] L. Mascotto, I. Perugia, and A. Pichler. A nonconforming Trefftz virtual element method for the Helmholtz problem: numerical aspects. *Computer Methods in Applied Mechanics and Engineering*, 347:445–476, 2019.

[103] L. Mascotto, I. Perugia, and A. Pichler. The nonconforming Trefftz virtual element method: general setting, applications, and dispersion analysis for the Helmholtz equation. In *The Virtual Element Method and its Applications*, pages 363–410. Springer, 2022.

[104] M. Mengolini, M. F. Benedetto, and A. M. Aragón. An engineering perspective to the virtual element method and its interplay with the standard finite element method. *Computer Methods in Applied Mechanics and Engineering*, 350:995–1023, 2019.

[105] D. Mora and G. Rivera. A priori and a posteriori error estimates for a virtual element spectral analysis for the elasticity equations. *IMA Journal of Numerical Analysis*, 40(1):322–357, 2020.

[106] D. Mora, G. Rivera, and R. Rodríguez. A virtual element method for the steklov eigenvalue problem. *Mathematical Models and Methods in Applied Sciences*, 25(08):1421–1445, 2015.

[107] D. Mora, G. Rivera, and R. Rodríguez. A posteriori error estimates for a virtual element method for the Steklov eigenvalue problem. *Computers & Mathematics with Applications*, 74(9):2172–2190, 2017.

[108] L. Mu, J. Wang, and X. Ye. Weak Galerkin finite element methods on polytopal meshes. *International Journal of Numerical Analysis & Modeling*, 12(1), 2015.

[109] A. Ortiz-Bernardin, C. Alvarez, N. Hitschfeld-Kahler, A. Russo, R. Silva-Valenzuela, and E. Olate-Sanzana. Veamy: an extensible object-oriented C++ library for the virtual element method. *Numerical Algorithms*, 82:1189–1220, 2019.

[110] G. H. Paulino and A. L. Gain. Bridging art and engineering using Escher-based virtual elements. *Structural and Multidisciplinary Optimization*, 51:867–883, 2015.

[111] D. Prada, S. Bertoluzza, M. Pennacchio, and M. Livesu. FETI-DP preconditioners for the virtual element method on general 2D meshes. In *Numerical Mathematics and Advanced Applications ENUMATH 2017*, pages 157–164. Springer, 2019.

[112] A. Quarteroni, G. Rozza, and A. Manzoni. Certified reduced basis approximation for parametrized partial differential equations and applications. *Journal of Mathematics in Industry*, 1(1):1–49, 2011.

[113] B. Rivière. *Discontinuous Galerkin methods for solving elliptic and parabolic equations: theory and implementation*. SIAM, 2008.

[114] T. Sorgente, D. Prada, D. Cabiddu, S. Biasotti, G. Patanè, M. Pennacchio, S. Bertoluzza, G. Manzini, and M. Spagnuolo. VEM and the Mesh. In *The Virtual Element Method and its Applications*, pages 1–57. Springer, 2022.

[115] N. Sukumar and A. Tabarraei. Conforming polygonal finite elements. *International Journal for Numerical Methods in Engineering*, 61(12):2045–2066, 2004.

[116] O. J. Sutton. The virtual element method in 50 lines of MATLAB. *Numerical Algorithms*, 75(4):1141–1159, 2017.

[117] C. Talischi, G. H. Paulino, A. Pereira, and I. Menezes. PolyMesher: a general-purpose mesh generator for polygonal elements written in Matlab. *Structural and Multidisciplinary Optimization*, 45:309–328, 2012.

[118] G. Vacca. An H^1-conforming virtual element for Darcy and Brinkman equations. *Mathematical Models and Methods in Applied Sciences*, 28(01):159–194, 2018.

[119] P. Valtr. Planar point sets with bounded ratios of distances. *Dissertation, Freie Univ.*, 1994.

[120] S. Vendoschot. Generating random convex polygons, 2017.

[121] G. Wang, F. Wang, L. Chen, and Y. He. A divergence free weak virtual element method for the Stokes–Darcy problem on general meshes. *Computer Methods in Applied Mechanics and Engineering*, 344:998–1020, 2019.

[122] H. Wei, Y. Deng, and F. Wang. Gradient recovery type a posteriori error estimates of virtual element method for an elliptic variational inequality of the second kind. *Nonlinear Analysis: Real World Applications*, 73:103903, 2023.

[123] P. Wriggers and B. Hudobivnik. A low order virtual element formulation for finite elasto-plastic deformations. *Computer Methods in Applied Mechanics and Engineering*, 327:459–477, 2017.

[124] P. Wriggers, B. Hudobivnik, and F. Aldakheel. Serendipity virtual elements for general element shapes. *Computer Methods in Applied Mechanics and Engineering*, 2020.

[125] P. Wriggers, B. D. Reddy, W. Rust, and B. Hudobivnik. Efficient virtual element formulations for compressible and incompressible finite deformations. *Computational Mechanics*, 60:253–268, 2017.

[126] P. Wriggers, W. T. Rust, and B. D. Reddy. A virtual element method for contact. *Computational Mechanics*, 58(6):1039–1050, 2016.

[127] Y. Yu. mVEM: A MATLAB software package for the virtual element methods. *arXiv preprint arXiv:2204.01339*, 2022.

[128] B. Zhang, J. Zhao, Y. Yang, and S. Chen. The nonconforming virtual element method for elasticity problems. *Journal of Computational Physics*, 378:394–410, 2019.

[129] J. Zhao, B. Zhang, S. Mao, and S. Chen. The nonconforming virtual element method for the Darcy–Stokes problem. *Computer Methods in Applied Mechanics and Engineering*, 370:113251, 2020.

Appendix A

Curriculum vitae

Personal details

Name	Fabio Credali
Date of birth	June 18, 1996
Place of birth	Pavia, Italy
Citizenship	Italian

Education

09/2010 - 06/2015	High school: Istituto Tecnico "G. Cardano", Pavia, Italy
10/2015 - 09/2018	Bachelor degree in Mathematics, Università degli Studi di Pavia, Italy
10/2018 - 09/2020	Master degree in Mathematics, Università degli Studi di Pavia, Italy
10/2020 - 11/2023	Doctoral studies in Computational Mathematics and Decision Sciences, Università degli Studi di Pavia, Italy, and Università della Svizzera Italiana, Lugano, Switzerland
01/2021 - 11/2023	Doctoral studies in Applied Mathematics and Computer Science, King Abdullah University of Science and Technology, Thuwal, Saudi Arabia

Appendix B

Academic activity

B.1 Papers

The doctoral studies reported in this thesis have contributed to the publication of the following papers:

- D. Boffi, **F. Credali**, and L. Gastaldi. On the interface matrix for fluid-structure interaction problems with fictitious domain approach, *Computer Methods in Applied Mechanics and Engineering*, 401 (2022): 115650. (Chapter 2)

- D. Boffi, **F. Credali**, L. Gastaldi, and S. Scacchi. A parallel solver for fluid structure interaction problems with Lagrange multiplier, *arXiv preprint arXiv:2212.13410, submitted*, 2022. (Chapter 3)

- D. Boffi, **F. Credali**, L. Gastaldi, and S. Scacchi. A parallel solver for FSI problems with fictitious domain approach, *Mathematical and Computational Applications*, 28.2 (2023): 59. (Chapter 3)

- D. Boffi, **F. Credali**, and L. Gastaldi. Quadrature error estimates for a fictitious domain formulation of fluid–structure interaction problems, *in preparation*. (Chapter 2)

- **F. Credali**, S. Bertoluzza, and D. Prada. Reduced basis stabilization and post-processing or the virtual element method, *arXiv preprint arXiv:2310.00625, submitted*, 2023. (Chapter 5)

Off-topic papers:

- C. Astuto, D. Boffi, and **F. Credali**, Finite element discretization of a biological network formation system: a preliminary study, *arXiv preprint arXiv:2303.10625*, to appear in *Proceeding of the XVIII International Conference on Hyperbolic Problems: Theory, Numerics, Applications*, 2023

B.2 Conferences

The work presented in this thesis has been presented in the following conferences:

- CompMath 2022 - Spring Workshop Joint PhD UniPV–USI.
  Università degli Studi di Pavia, Italy. 16-17/03/2022.
  Talk: *Model order reduction in support of the Virtual Element Method.*

- European Finite Element Fair 2022.
  Aalto University, Finland. 03-04/06/2022.
  Talk: *Numerical approximation of fluid-structure interaction problem: a fictitious domain approach.*

- Second Young Applied Mathematicians Conference.
  Arenzano, Italy. 17-23/09/2022.
  Mini-course: *Numerical approximation of fluid-structure interaction problem.*

- Africomp5 - 5^{th} African Conference on Computational Mechanics.
  Cape Town, South Africa. 02-04/11/2022.
  Talk: *A parallel solver for FSI problems with fictitious domain approach.*

- 5^{th} KAUST CEMSE MaS Workshop on Modelling and Simulation.
  King Abdullah University of Science and Technology. 19/03/2023 - 22/03/2023.
  Talk: *A Lagrange multiplier formulation for finite element discretization of FSI problems.*

- European Finite Element Fair 2023.
  University of Twente, Netherlands. 12-13/05/2023.
  Talk: *Model order reduction in support of the virtual element method.*

- YIC 2023 - ECCOMAS Young Investigators Conference.
  University of Porto, Portugal. 19/06/2023 - 21/06/2023.
  Talk: *Finite element discretization of fluid-structure interaction problems with fictitious domain approach.*

- SIMAI 2023 - Bi-annual congress of the Italian Society of Applied and Industrial Mathematics.
  Università degli Studi della Basilicata, Matera, Italy. 27/08/2023 - 01/09/2023.
  Talk: *Finite element discretization of fluid–structure interaction problems with Lagrange multiplier: how to deal with the coupling term.*

- ENUMATH 2023 - European Conference on Numerical Mathematics and Advanced Applications.
  Lisbon, Portugal. 04/09/2023 - 08/09/2023.
  Talk: *A parallel solver for fluid structure interaction problems with Lagrange multiplier.*