POLITECNICO DI MILANO

School of Industrial and Information Engineering
Master of Science in Mathematical Engineering

Reduced Basis Method for Isogeometric Analysis: application to structural problems.

Internal Advisor: Prof. Alfio Quarteroni
External Advisor: Dr. Luca Dede'
Co-Advisor: Federico Negri

Master Thesis of:
Marina Rinaldi
Matr. 800498

Academic Year 2013/2014

Alla mia famiglia

Contents

List of Figures

List of Tables

Abstract

Isogeometric Analysis (IGA) is a computational methodology for the numerical approximation of Partial Differential Equations (PDEs). IGA is based on the *isogeometric concept*, for which the same basis functions, usually Non-Uniform Rational B-Splines (NURBS), are used both to represent the geometry and to approximate the unknown solutions of PDEs. Compared to the standard Finite Element method, NURBS-based IGA offers several advantages: ideally a direct interface with CAD tools, exact geometrical representation, simple refinement procedures, and smooth basis functions allowing to easily solve higher order problems, including structural shell problems. In these contexts, repeatedly solving a problem for a large set of geometric parameters might lead to high and eventually prohibitive computational cost. To cope with this problem, we consider in this work the Reduced Basis (RB) method for the solution of parameter dependent PDEs, specifically for which the NURBS representation of the computational domain is parameter dependent.

RB refers to a technique that enables a rapid and reliable approximation of parametrized PDEs by constructing low dimensional approximation spaces. In this work, for the construction of the reduced spaces we adopt two different strategies, namely the Proper Orthogonal Decomposition and the greedy algorithm.

In this thesis we combine RB and IGA for the efficient solution of parametrized problems for all the possible cases of NURBS geometrical parametrizations, which specifically include the NURBS control points, the weights, and both the control points and weights. In particular, we first focus on the solution of second order PDEs on parametrized lower dimensional manifolds, specifically surfaces in the three dimensional space. We consider geometrical parametrizations that entail a nonaffine dependence of the variational forms on the spatial coordinates and the geometric parameters. Thus, depending on the parametrization at hand and in order to ensure a suitable Offline/Online decomposition between the reduced order model construction and solution, we resort to the Empirical Interpolation Method (EIM) or the Matrix Discrete Empirical Interpolation Method (MDEIM), by comparing their performances.

As application, we solve a class of benchmark structural problems modeled by Kirchoff-Love shells for which we consider NURBS geometric parametrizations and we apply the RB method to the solution of this class of fourth order PDEs. We highlight by means of numerical tests, the performance of the RB method applied to standard IGA approximation of parametrized shell geometries.

Estratto

L'Analisi Isogeometrica (IGA) è una metodologia computazionale per l'approssimazione numerica delle Equazioni Differenziali alle Derivate Parziali (PDEs). IGA si basa sul *paradigma isogeometrico*, secondo il quale le stesse funzioni di base, di solito Non-Uniform Rational B-Splines (NURBS), vengono utilizzate sia per rappresentare la geometria che per approssimare le soluzioni delle PDEs. Rispetto al metodo degli Elementi Finiti, IGA offre diversi vantaggi: diretta interfacciabilità con gli strumenti CAD, rappresentazione esatta delle geometrie, semplici procedure di raffinamento e funzioni di base smooth che consentono di risolvere problemi di alto ordine, inclusi problemi strutturali di tipo shells. In questo contesto, la risoluzione ripetuta di un problema per un gran numero di parametri geometrici potrebbe richiedere costi computazionali eccessivamente elevati. Per far fronte a questo problema, in questa tesi consideriamo il metodo delle Basi Ridotte per la risoluzione di PDEs parametrizzate, per le quali, in particolare, la rappresentazione della geometria costruita tramite NURBS dipende da parametri.

La metodologia RB consente una valutazione rapida, efficiente ed accurata di queste PDEs parametrizzate tramite la costruzione di spazi approssimanti di piccole dimensioni. In questo lavoro per la costruzioni di tali spazi ridotti adottiamo due diverse strategie, nello specifico la Proper Orthogonal Decomposition e un algoritmo greedy.

Lo scopo di questa tesi è l'applicazione di RB al metodo IGA per una risoluzione efficiente di problemi parametrizzati per tutti i possibili casi di parametrizzazione delle geometrie costruite tramite NURBS, che nello specifico può riguardare i punti di controllo delle NURBS, i pesi delle NURBS, o sia i punti di controllo che i pesi. Dapprima ci focalizziamo sulla risoluzione di PDEs del secondo ordine su manifolds, in particolare su superfici nello spazio tridimensionale. Consideriamo parametrizzazioni geometriche che causano una dipendenza non affine delle forme variazionali sia rispetto alle coordinate spaziali che ai parametri geometrici. Pertanto, in base alla parametrizzazione considerata e per assicurare una decomposizione Offline/Online efficiente tra la costruzione del modello di ordine ridotto e la sua risoluzione, ricorriamo all' Empirical Interpolation Method (EIM) o al Matrix Empirical Interpolation Method (MDEIM), confrontandone le prestazioni.

Come applicazione, risolviamo una classe di problemi strutturali modellati da Kirchoff-Love shells per i quali consideriamo delle parametrizzazioni geometriche ed applichiamo il metodo RB per la risoluzione di questa classe di PDEs del quarto ordine. Attraverso test numerici mettiamo in evidenza le prestazioni del metodo RB applicato a problemi definiti su shells parametrizzate e approssimati tramite IGA.

Introduction

Isogeometric Analysis (IGA) is a computational methodology for the numerical approximation of PDEs introduced by Hughes et al. in ([CHB05]). The aim of IGA is to help designers of Engineering systems to make the computational geometries suitable for the numerical solution of PDEs, as well as facilitating the interaction of Computer Aided Design (CAD) and Finite Elements method (FEM) technologies ([CHB09]). IGA makes this possible by considering a unique geometric model directly usable as an analysis model thanks to the *isogeometric concept*, for which the same basis functions are used both to represent the geometry and to approximate the unknown solutions of the PDEs. The most widely used computational geometry in Engineering design are NURBS (Non-Uniform Rational B-splines) that are defined by combinations of B-splines and *weights* ([PT97]). In this context, all conic sections, circles, cylinders, spheres, ellipsoids, etc, can be exactly represented by means of geometrical mapping described by linear combinations of NURBS functions and *control points* ([PT97]). Based on the isogeometric concept, in this work, NURBS are used both to represent the computational domains and to build a basis for the analysis. Compared to classical FEA, IGA offers several advantages, such as: exact geometrical representation, simple refinement procedures, and smooth basis functions, eventually globally C^k-continuous, with $k \geq 0$. Moreover, IGA is attractive for solving PDEs under shape and geometrical parametrizations. As a matter of fact, since the IGA mapping involves only the NURBS *control points* and the NURBS basis functions, one can change the geometrical configuration of a computational domain in a straightforward manner, by considering the position of the NURBS control points or the weights as parameters.

With this motivation, in this work, we focus on the numerical solution of second order PDEs on parametrized lower dimensional manifolds, specifically surfaces in the three dimensional space. In this context, the standard case of second order PDEs in the two dimensional space is automatically recovered if the control points are in 2D. Moreover, the additional regularity of NURBS basis functions used in IGA allows to treat high order problems ([BDQ15, TDQ14]). Thus, as final application, we consider a benchmark class of structural problems modeled under the Kirchoff-Love shell theory, for which we solve parametrized problems. In all these cases, the problem is that, if one is interested in evaluating the solution of the problem for a large number of geometrical parameters or in real time, then the total computational cost could become extremely high and eventually prohibitive. To overcome this issue, we consider in this work, the Reduced Basis method ([GP05, Man12, Qua14, QRM11, SVH$^+$06]).

The Reduced Basis (RB) method is a technique that enables a rapid and reliable approximation of parametrized PDEs (PPDEs) by constructing low dimensional approximation spaces. Using these spaces for the discretization of the original problem, we can build a reduced order model which is a sufficiently accurate approximation of the full order original problem. Indeed, reduction strategies can be crucial in applications of high complexity. As a

matter of fact, although the increasing computer power allows nowadays to solve problems of very large dimension, a computational reduction is still necessary in *real-time* simulations and *many-query* contexts ([GP05, Man12, Qua14, QRM11, SVH$^+$06]). In case of optimal control or shape optimization problems, for instance, the computational cost is mainly associated to iterative optimization procedures, requiring recursive evaluations of the quantities of interest. More in general, in Engineering practice, one could be interested in the solution of a PPDE that describes a particular phenomenon whose parameters are related to physical properties, boundary conditions, or loads. In this work, as already anticipated, we are interested in taking into account shape variations of the computational domains, thus we consider geometric parameters.

Historically, RB methods have been built upon Finite Element discretizations (e.g. [Man12, RHP08, VPRP03]) even if spectral elements ([LMR06]), and finite volume ([HO08]) methods have been considered. In this thesis, we build a *RB approximation* upon a *IGA approximation*. To the best of our knowledge, there is only another work ([MSH15]) devoted to the application of RB to IGA; in particular, in [MSH15] the construction of the RB method relies on an Isogeometric Boundary Element Method and the geometric parametrization regards only the NURBS control points coordinates. Conversely, in this thesis, we consider the more general case of NURBS geometry parametrization, that, can affect, the control points, the weights, or both control points and weights. For all these cases, we show, for the first time, how RB and IGA can be jointly used in an efficient way.

We remark that the crucial point of the RB and IGA coupling regards the affinity assumption. In order to ensure a suitable Offline/Online decomposition between the reduced spaces construction and the evaluation of the solution of the reduced problem, the bilinear forms and the linear functionals of the variational formulation must depend in an affine way on the spatial coordinates and on the geometric parameters. In this manner, at the Offline stage one can assemble and store the parameter independent structures only once, while at the Online stage, for each new parameter, one can just evaluate the parameter dependent terms and solve the reduced system. However, the geometric NURBS parametrizations severally lead to a nonaffine dependence on these quantities, therefore requiring to resort to suitable methodologies to restore the affine dependence of the weak formulation of the problem.

We start by considering the case in which the parametrization of the surface concerns only the NURBS control points coordinates, while the NURBS weights are not parameter dependent; in this case, to restore the affinity assumption we map the problem on a reference configuration and appeal to the Empirical Interpolation Method (EIM) ([BMNP04]) to affinely approximate the parameter dependent terms appearing in the integral formulation of the problem. Then, we consider the case in which the parametrization concerns also the NURBS weights. However, we see that, when the NURBS weights are parametrized, the EIM approach is unusable, since one is unable to define a reference configuration and to apply in an efficient way EIM to the parameter dependent terms appearing in the integral formulation of the problem. As a matter of fact, if all the NURBS weights are parameter dependent, the Online evaluation stage would depend on the full dimension of the problem, which is precisely what should be avoided to build efficient RB. For these reasons, in order to restore the affinity of the forms and functionals, we appeal to a recent alternative methodology, the Matrix Discrete Empirical Interpolation Method (MDEIM) ([NMA15]), which directly operates the affine decomposition on the matrix and right hand side vector of the parametrized linear system. Moreover, MDEIM is very general and can be used with both parametrizations of NURBS control points and weights.

As final application, we focus on structural problems, specifically on thin shells. We remark that a model derivative reduction method based on a Isogeometric discretization has already been applied to the vibration analysis and nonlinear structural analysis of 3D problems ([WWS14]). However, in this thesis, we apply RB to thin structures modeled as Kirchoff-Love shells undergoing geometric parametrizations. We remark that Kirchoff-Love shells have been analyzed both in the FE framework ([SFR89, ACdSAF03, BB93]) and IGA framework ([KBLW09, CHB09]), even if never in a parametrized context. Finally, we approximate these models by means of MDEIM and apply, for the first time, the RB method for a rapid and reliable solution of parametrized fourth order PDEs.

The work is organized as follows:

Chapter 1. We introduce the main elements of IGA by describing B-splines, NURBS, and refinement techniques. Moreover, we recall the isogeometric concept and introduce IGA as a method for the numerical approximation of second order PDEs on lower dimensional manifolds, specifically surfaces in a three dimensional space. Then, we solve benchmark problems and introduce geometrically parametrized problems.

Chapter 2. We introduce the main features of the RB method and describe the setting of the parametrized formulations of the problems considered in this work. We show the steps to follow to obtain a reduced order model and, in particular, describe two sampling strategies to obtain the low dimensional spaces, namely the Proper Orthogonal Decomposition (POD) and the Greedy algorithm. Moreover, we recall an *a posteriori* error estimation, a fundamental ingredient for the application of the RB method and for the certification of the reduced solutions.

Chapter 3. We consider a class of problems described by PDEs defined on parametrized geometries, such that the parametrization affects only the NURBS control points coordinates. We describe EIM as a suitable technique to restore the affinity assumption. Then, we apply EIM to an academical test problem and solve it by means of the RB method. Finally, we show some numerical results to validate the procedure.

Chapter 4. We consider a class of problems described by PDEs defined on parametrized geometries, such that both the NURBS control points coordinates and weights are parameter dependent. By means of an example, we explain why EIM is not usable. We then describe the MDEIM as an alternative technique to restore the affinity assumption and apply it to an academic test problem solved by means of the RB strategy. Finally, we show some numerical results and a comparison between the EIM and the MDEIM performances.

Chapter 5. We describe the Kirchoff-Love theory of shells and provide its IGA discretization. Then, we introduce a parametrized version of this model, apply MDEIM to restore the affinity assumption and then obtain a RB approximation. Finally, we present the numerical results for some of the shell course problems.

Conclusions. We draw some conclusions on this work and suggest some further extensions and applications.

All our numerical simulations were performed by means of tools and software based on GeoPDEs (`http://geopdes.sourceforge.net`), Matlab (`http://www.mathworks.`

com), and a IGA library for the implementation of Kirchoff-Love shells models (http://sourceforge.net/projects/cmcodes/?source=typ_redirect). The visualizations were performed by means of ParaView (http://www.paraview.org/).

Sintesi

L'Analisi Isogeometrica (IGA) è una metodologia computazionale per l'approssimazione numerica delle PDEs introdotta da Hughes et al. in ([CHB05]). Lo scopo di IGA è di aiutare i designers dei sistemi Ingegneristici nel rendere le geometrie computazionali adatte alla risoluzione numerica delle PDEs, facilitando l'interazione tra gli strumenti di Computer Aided Design (CAD) ed il metodo agli Elementi Finiti (FEM) ([CHB09]). IGA rende ciò possibile considerando un'unica geometria utilizzabile direttamente come modello per l'analisi grazie al *paradigma isogeometrico*, secondo il quale le stesse funzioni di base vengono utilizzate sia per rappresentare la geometria che per approssimare le soluzioni delle PDEs. In questa tesi utilizziamo le NURBS (Non-Uniform Rational B-splines) sia per rappresentare i domini computazionali che come funzioni di base per l'analisi. Le NURBS sono definite come combinazione lineare di B-splines e *pesi* ([PT97]). Tramite le NURBS tutte le sezioni coniche, circolari, cilindriche, sferiche, elissoidali, etc, possono essere rappresentate esattamente tramite mappe geometriche descritte da combinazioni lineari di funzioni NURBS e *punti di controllo* ([PT97]). Rispetto al metodo degli Elementi Finiti, il metodo IGA offre diversi vantaggi: diretta interfacciabilità con gli strumenti CAD, rappresentazione esatta delle geometrie, semplici procedure di raffinamento e funzioni di base smooth con continuità globale eventualmente C^k, con $k \geq 0$. Inoltre, IGA risulta essere particolarmente adatta per la risoluzione di PDEs soggette a parametrizzazioni geometriche. Infatti, poiché la mappa IGA coinvolge i punti di controllo delle NURBS e le funzioni di base NURBS, è possibile cambiare una configurazione geometrica in maniera semplice, considerando la posizione dei punti di controllo o i pesi come parametri.

Per questi motivi, in questa tesi, ci focalizziamo sulla risoluzione numerica di PDEs del secondo ordine su manifolds di bassa dimensione, in particolare superfici nello spazio tridimensionale. Inoltre, la regolarità delle funzioni di base NURBS permette di trattare problemi di alto ordine ([BDQ15, TDQ14]). Pertanto, come applicazione finale, consideriamo una classe di problemi strutturali modellati da Kirchoff-Love shells e in questo contesto risolviamo problemi parametrici. In tutti questi casi, la risoluzione ripetuta di un problema per un gran numero di parametri geometrici potrebbe richiedere costi computazionali eccessivamente elevati. Per far fronte a questo problema, in questa tesi consideriamo il metodo delle Basi Ridotte ([GP05, Man12, Qua14, QRM11, SVH$^+$06]) per la risoluzione di PDEs parametrizzate, per le quali, in particolare, la rappresentazione della geometria costruita tramite NURBS è dipendente da parametri.

Il metodo alle Basi Ridotte è una metodologia che consente una valutazione rapida, efficiente ed accurata di PDEs parametrizzate tramite la costruzione di spazi approssimanti di piccole dimensioni. Tali strategie di riduzione possono essere fondamentali per applicazioni altamente complesse garantendo grandi riduzioni dei tempi computazionali richiesti ([GP05, Man12, Qua14, QRM11, SVH$^+$06]). Storicamente, il metodo RB è stato costruito

su discetizzazioni agli Elementi Finiti (e.g. [Man12, RHP08, VPRP03]), ma sono stati considerati anche metodi agli elementi spettrali ([LMR06]) e volumi finiti ([HO08]). In questa tesi, costruiamo un'approssimazione RB su una discretizzazione IGA, considerando il caso più generale di parametrizzazione di una geometria costruita tramite NURBS, che può riguardare i punti di controllo, i pesi, o sia i punti di controllo che i pesi. Per tutti questi casi vediamo, per la prima volta, come RB e IGA possono essere interfacciate tra loro.

Nella procedura di applicazione di RB a IGA, il punto cruciale riguarda il soddisfacimento dell'ipotesi di affinità. Al fine di assicurare una decomposizione Offline/Online efficiente tra la costruzione del modello di ordine ridotto e la sua risoluzione, le forme bilineari e i funzionali lineari della forma variazionale devono dipendere in maniera affine dalle coordinate spaziali e dai parametri geometrici. Le parametrizzazioni geometriche che considereremo, causeranno sempre una dipendenza non affine del problema da queste quantità e vedremo che qualora la parametrizzazione coinvolge solo i punti di controllo delle NURBS, l'affinità potrà essere recuperata tramite l'Empirical Interpolation Method (EIM) [BMNP04], mente se la parametrizzazione riguarda anche i pesi delle NURBS allora dovremo ricorrere ad un'altra metodologia, il Matrix Discrete Empirical Interpolation Method (MDEIM) [NMA15].

Come applicazione finale, consideriamo, come anticipato, una classe di problemi strutturali. In particolare, studiamo strutture sottili modellate da Kirchoff-Love shells e soggette a parametrizzazione geometrica. I modelli Kirchoff-Love sono stati ampliamente analizzati sia in ambito FE ([SFR89, ACdSAF03, BB93]) che in ambito IGA ([KBLW09, CHB09]), anche se non in contesto parametrizzato. Infine, approssimiamo questi modelli tramite MDEIM e applichiamo, per la prima volta, il metodo RB per una valutazione rapida ed accurata di PDEs del quarto ordine parametrizzate.

Chapter 1

Isogeometric Analysis

In this chapter we revisit the basic concepts and properties of Isogeometric Analysis (IGA), a computational methodology for the numerical approximation of Partial Differential Equations (PDEs) [HCB05, CHB09]. IGA is based on the *isogeometric concept*, for which the same basis functions are used both to represent the geometry and to approximate the unknown solutions of the PDEs. The aim of the methodology is to help designers of Engineering systems in making the geometries suitable for the numerical solution of PDEs. As a matter of fact, nowadays designers generate Computer Aided Design (CAD) geometries and use them as inputs for numerical methods aimed to find approximate solutions to boundary value problems for PDEs, usually the Finite Element Analysis (FEA) method. To this end, it is also requested to realize suitable meshes (piecewise approximations of the actual computational domain) that properly fit the geometries. The result is that the construction of the geometry and the mesh represents one of the most time consuming steps in FEA and, in addition to that, the geometrical approximation may introduce errors and can cause significant accuracy issues. The necessity to break down the barriers between engineering design and analysis led to Isogeometric Analysis. In IGA the gap between CAD and FEA is ideally filled by considering a unique geometric model directly usable as an analysis model thanks to the *isogeometric concept*. The most widely used computational geometry technology in Engineering design are NURBS (non-uniform rational B-splines) [PT97] that can exactly represent all conic sections, circles, cylinders, spheres, ellipsoids, etc. In this thesis, we will use NURBS both to represent our computational domains and to build a basis for the analysis. For the numerical approximation of PDEs, we will consider NURBS-based IGA in the framework of the Galerkin method to solve elliptic PDEs, although different numerical methods can be used, such as collocation or least square methods [CHB09]. In the following, we will deal with geometries that can be modeled as a single *patch*, a *physical domain* (surfaces in $\mathbb{R}^d$, $d = 2, 3$) topologically representable as a mapping of the *parameter domain* (a unit square in 2D).

This chapter is devoted to the introduction and description of the main elements and features of Isogeometric Analysis. In Sections 1.1, 1.2, 1.3, we start by presenting IGA from a geometric point of view, by describing B-splines, NURBS, and refinement techniques. In Section 1.4 we describe the *isogeometric concept* and present IGA based on the Galerkin method for solving second order partial differential problems defined on manifolds. In Section 1.5 we present a benchmark class of problems. In Section 1.6 we introduce a parametrization of the IGA domains.

1.1 B-splines

We start by introducing knot vectors, B-spline basis functions, and B-spline geometries that are the foundamental elements to build NURBS.

1.1.1 Knot vectors

A *knot vector* Ξ in one dimension is a non-decreasing set of coordinates in the parameter space such that $\Xi = \{\xi_1, \xi_2, .., \xi_{n+p+1}\}$, where $\xi_i \in \mathbb{R}$ is the i^{th} knot, with $i = 1, ..., n+p+1$ the knot index, p the polynomial degree, and n the number of basis functions used to construct the B-spline curve (see Section 1.1.2). By convention in 1D the *parameter domain* is $\hat{\Omega} = [0, 1] \subset \mathbb{R}$ and we assume $\xi_1 = 0$ and $\xi_{n+p+1} = 1$, so that $\hat{\Omega} = [\xi_1, \xi_{n+p+1}]$. The interval $[\xi_i, \xi_{i+1})$ is called the i^{th} *knot span*; if repeated knots exist, the corresponding knot span has zero length. The subdomain bounded by two distinct consecutive knots will be called *element* of the *mesh* in $\hat{\Omega}$. The knot vector can be *uniform* if the knots are equally spaced in the parameter space, otherwise it will be *non-uniform*; in addition Ξ is *open*, if its first and last knot values are repeated $p + 1$ times. The basis functions formed from open knot vectors are interpolatory at the extremes of the parameter space interval $[\xi_1, \xi_{n+p+1}]$ in one dimension, but in general they are not interpolatory at interior knots. We now propose an example in 2D.

Example 1.1. *Let us define an open knot vector in parametric direction* ξ, $\Xi = \{\xi_1, ..., \xi_{n_1+p+1}\} = \{\xi_1, \xi_2, \xi_3, \xi_4\} = \{0, 0, 1, 1\}$, *where we have chosen polynomial degree* $p = 1$, *and a second open knot vector in parametric direction* η, $H = \{\eta_1, ..., \eta_{n_2+q+1}\} = \{\eta_1, \eta_2, \eta_3, \eta_4, \eta_5, \eta_6\} = \{0, 0, 0, 1, 1, 1\}$, *where the polynomial degree is* $q = 2$. *In the 2D case we have that* $\hat{\Omega} = [0, 1] \times [0, 1] = [\xi_1, \xi_4] \times [\eta_1, \eta_6]$ *as shown in Fig. 1.1.*

1.1.2 Univariate and bivariate B-spline basis functions

Once introduced the knot vector, the B-spline basis functions $N_{i,p}$ are recursively defined for the different degrees p by using the *Cox-de Boor recursion formula* [CHB09]. The B-splines space built by univariate B-spline basis functions is denoted by

$$\hat{\mathcal{S}}_h := \text{span}\{N_{i,p}, i = 1, ..., n\}, \tag{1.1}$$

where the parameter h indicates its finite dimension. Due to the recursive definition of the B-spline basis functions, the derivatives of these functions are expressed in terms of B-spline basis functions of lower order. Given a polynomial degree p and a knot vector Ξ, the derivative of the i^{th} basis function is:

$$\frac{d}{d\xi} N_{i,p}(\xi) = \frac{p}{\xi_{i+p} - \xi_i} N_{i,p-1}(\xi) - \frac{p}{\xi_{i+p+1} - \xi_{i+1}} N_{i+1,p-1}(\xi). \tag{1.2}$$

This formula can be generalized in case of higher order derivatives [CHB09].

In Fig. 1.2 we depict the linear ($p = 1$) and the quadratic ($q = 2$) B-spline basis functions determined respectively by the knot vectors Ξ and H introduced in Example 1.1, with $n_1 = 2$ and $n_2 = 3$. In Fig. 1.3 we depict the linear basis functions generated by the open knot vector $\Xi = \{0, 0, 1, 2, 3, 4, 5, 5\}$, while in Fig. 1.4 we report the quadratic basis functions generated by the open knot vector $\Xi = \{0, 0, 0, 1, 2, 3, 4, 5, 5, 5\}$.

We now list some important properties of the B-spline basis functions.

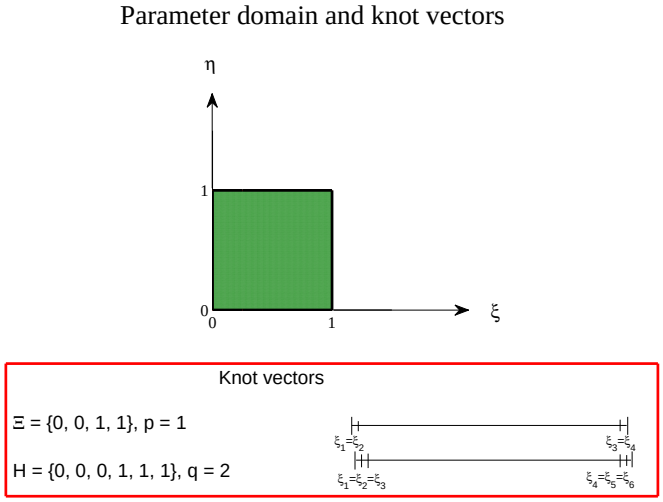


Figure 1.1: The open knot vectors of Example 1.1: $\Xi = \{\xi_1, ..., \xi_{n_1+p+1}\} = \{\xi_1, \xi_2, \xi_3, \xi_4\} = \{0, 0, 1, 1\}$, $H = \{\eta_1, ..., \eta_{n_2+q+1}\} = \{\eta_1, \eta_2, \eta_3, \eta_4, \eta_5, \eta_6\} = \{0, 0, 0, 1, 1, 1\}$, and the parameter domain $\hat{\Omega} = [0, 1] \times [0, 1] = [\xi_1, \xi_4] \times [\eta_1, \eta_6] \subset \mathbb{R}^2$.

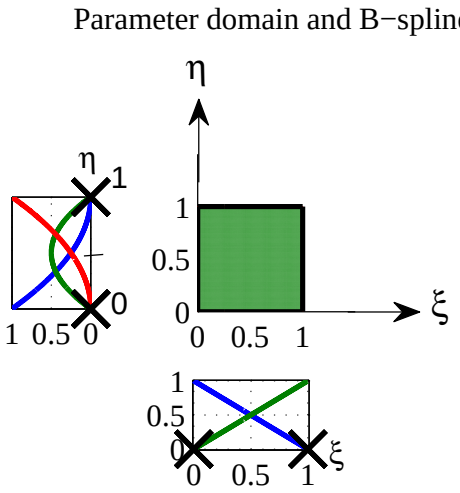


Figure 1.2: Linear basis functions in parametric direction ξ for the open knot vector $\Xi = \{0, 0, 1, 1\}$ and quadratic basis functions in parametric direction η for the open knot vector $H = \{0, 0, 0, 1, 1, 1\}$.

P1 *Local support property*: the support of the B-spline basis functions of degree p is always $p + 1$ knot spans, that is

$$N_{i,p}(\xi) = 0 \quad \text{if } \xi \text{ is outside the interval } [\xi_i, \xi_{i+p+1}). \tag{1.3}$$

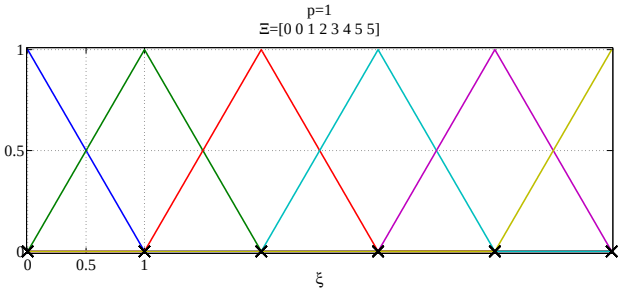


Figure 1.3: Univariate linear basis functions for the open knot-vector $\Xi = \{0, 0, 1, 2, 3, 4, 5, 5\}$.

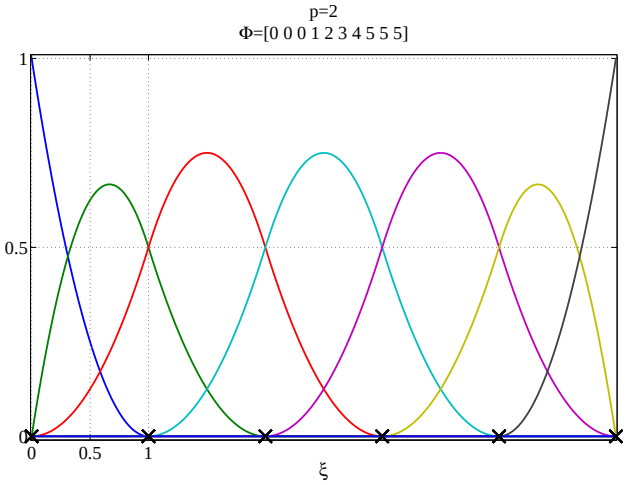


Figure 1.4: Univariate quadratic basis functions for the open knot-vector $\Xi = \{0, 0, 0, 1, 2, 3, 4, 5, 5, 5\}$.

As consequence, higher-degree basis functions have support over much larger portions of the domain than classical FEA functions do. Moreover, in the case of repeated knots, the support of each basis is still $p + 1$ knot spans (begins at ξ_i and ends at ξ_{i+p+1}, including repeated knots) even if some of these knot spans have zero measure.

P2 Each basis functions shares its support with $2p+1$ basis functions (including itself), as for the Lagrangian polynomial basis functions used in FEA. As consequence, the bandwidth (the number of nonzero entries) of the matrices associated to Galerkin formulations using B-spline spaces, which we are going to define in Section 1.4.4, does not change if repeated knots are present.

P3 *Partition of unity*, that is

$$\forall \xi \in \hat{\Omega} : \quad \sum_{i=1}^{n} N_{i,p}(\xi) = 1, \quad \forall p \geq 0. \tag{1.4}$$

P4 *Nonnegativity*: each basis function is pointwise nonnegative over the entire domain, that is

$$\forall i = 1, ..., n, \ p \geq 0, \ \text{and} \ \xi \in \hat{\Omega} : \quad N_{i,p}(\xi) \geq 0; \tag{1.5}$$

P5 $N_{i,p}$ is infinitely differentiable in the interior of the knot spans (where it is a polynomial). Each p^{th} degree function has $p-1$ continuous derivatives across each knot of multiplicity one. If a knot ξ_i has multiplicity $m_i \geq 1$, $N_{i,p}$ is only $p - m_i$ times continuous across it.

For other properties and a proof of properties P3 and P4 see [PT97].

By means of the tensor product of univariate basis functions $\{N_{i,p}, i = 1, ..., n_1\}$ and $\{M_{k,q}, k = 1, ..., n_2\}$, in ξ and η parametric directions respectively, we can define bivariate B-spline basis functions as

$$N_j(\xi, \eta) = N_{i,p}(\xi) M_{k,q}(\eta), \quad j = 1, ..., n \tag{1.6}$$

where $n = n_1 n_2$. The support of a given bivariate function N_j is $[\xi_i, \xi_{i+p+1}] \times [\eta_k, \eta_{k+q+1}]$, $(p+1)\times(q+1)$ knot spans. The number of continuous partial derivatives in a given parametric direction can be determined from the associated one dimensional knot vector and polynomial degree as in P5. The B-spline basis functions are pointwise nonnegative and constitute a partition of the unity since, $\forall(\xi, \eta) \in [\xi_i, \xi_{n_1+p+1}] \times [\eta_k, \eta_{n_2+q+1}]$,

$$\sum_{i=1}^{n_1} \sum_{k=1}^{n_2} N_{i,p}(\xi) M_{k,q}(\eta) = \left(\sum_{i=1}^{n_1} N_{i,p}(\xi) \right) \left(\sum_{k=1}^{n_2} M_{k,q}(\eta) \right) = 1. \tag{1.7}$$

The tensor product B-splines space, spanned by these basis functions, is defined as (1.1).

1.1.3 B-splines geometries

We now see how to use B-spline basis functions to define curves. To build these objects we need the following elements:

(i) $\{N_{i,p}(\xi), i = 1, 2, ..., n\}$, p^{th}-degree B-spline basis functions defined in the *parameter space* on the *knot vector* Ξ;

(ii) $\{\boldsymbol{B}_i, i = 1, 2, ..., n\} \in \mathbb{R}^d$, control points defined in the *physical space* in $\mathbb{R}^d$.

By means of these elements a *p^{th}-degree B-spline curve* in $\mathbb{R}^d$ is defined by

$$\boldsymbol{C} : \hat{\Omega} \to \Omega \subset \mathbb{R}^d, \quad \boldsymbol{C}(\xi) = \sum_{i=1}^{n} N_{i,p}(\xi) \boldsymbol{B}_i. \tag{1.8}$$

The polygon formed by the $\boldsymbol{B}_i$ is called the *control polygon* and represents a piecewise linear representation of the curve. We can interpret (1.8) as a transformation, from the parameter domain $\hat{\Omega}$ to the physical domain Ω, linking the knot vector Ξ in $\hat{\Omega}$ to the control points $\boldsymbol{B}_i$ in Ω. B-spline curves inherit most of their properties from the basis functions from which they are built. For instance, B-spline curves have at least as many continuous derivatives across an element boundary than its basis functions have across the corresponding knot value. Therefore, B-spline curves of degree p have $p-1$ continuous derivatives in the absence of repeated knots and repeated control points. Another property directly inherited by the basis functions is that of locality: as already said each basis function of order p has support on $p+1$ spans, therefore moving a single control point can affect the geometry of not more than $p+1$

mesh elements (images of the knot spans of non zero size in the physical domain) of the curve.

A *B-spline surface* in $\mathbb{R}^2$ is similarly defined. Given two knot vectors $\Xi = \{\xi_1, \xi_2, ..., \xi_{n_1+p+1}\}$ and $H = \{\eta_1, \eta_2, ..., \eta_{n_2+q+1}\}$, a bidirectional net of control points, and the tensor product of the univariate B-spline functions, a B-spline surface is given by

$$\boldsymbol{S}(\xi, \eta) = \sum_{i=1}^{n_1} \sum_{k=1}^{n_2} N_{i,p}(\xi) M_{k,q}(\eta) \boldsymbol{B}_{i,k} = \sum_{j=1}^{n} N_j(\xi, \eta) \boldsymbol{B}_j \tag{1.9}$$

where N_j are bivariate B-spline basis functions defined in (1.6), and $\{\boldsymbol{B}_j\} \in \mathbb{R}^2$, for $j = 1, ...n$ and $n = n_1 n_2$, is called *control net*. In Section 1.2.1 we will introduce manifolds, in particular surfaces in $\mathbb{R}^3$ with control points in $\mathbb{R}^3$. So, surfaces in 2D can be seen as a particular case of surfaces in 3D, for example when control points have a null third component. Also in this case, many of the properties of B-spline surfaces follow from their tensor product nature. For a similar definition of a *B-spline solid* and its properties see [CHB09].

1.2 Non-Uniform Rational B-splines

We now introduce Non-Uniform Rational B-splines (NURBS). The main difference between B-splines and NURBS is that B-splines are objects described by piecewise polynomials and cannot exactly represent conic sections (circles, arcs, cones, pipes, etc.). Conversely NURBS, exactly represent conic sections and a wider array of objects.

1.2.1 NURBS basis functions and NURBS geometries

As already done for B-splines, we need to construct a basis for the NURBS space from knot vectors and to build curves, surfaces, and solids by means of a linear combinations of basis functions and control points. In this way, NURBS inherit all the B-splines properties, except that they are not piecewise polynomials anymore. In order to build a NURBS basis, we introduce the *weighting function*, a real-valued scalar function defined by

$$W : \hat{\Omega} \to \mathbb{R}, \quad W(\xi) = \sum_{i=1}^{n} N_{i,p}(\xi) w_i, \tag{1.10}$$

where $N_{i,p}(\xi)$ is the standard univariate B-spline basis function and w_i are the weights (we assume $w_i > 0$) associated to each B-spline basis function. A univariate NURBS basis function is given by

$$R_{i,p} : \hat{\Omega} \to \mathbb{R}, \quad R_{i,p}(\xi) = \frac{N_{i,p}(\xi) w_i}{W(\xi)} = \frac{N_{i,p}(\xi) w_i}{\sum_{\hat{i}=\hat{1}}^{n} N_{\hat{i},p}(\xi) w_{\hat{i}}}, \tag{1.11}$$

which is a piecewise rational function. The univariate NURBS space over the parametric domain $\hat{\Omega}$ is denoted by:

$$\hat{\mathcal{B}}_h := \text{span}\{R_i, i = 1, ..., n\}. \tag{1.12}$$

The derivatives of NURBS basis functions are obtained by applying the quotient rule to (1.11), and we get

$$\frac{d}{d\xi} R_{i,p}(\xi) = w_i \frac{W(\xi) N'_{i,p}(\xi) - W'(\xi) N_{i,p}(\xi)}{(W(\xi))^2}, \tag{1.13}$$

where $N'_{i,p}(\xi) = \frac{d}{d\xi} N_{i,p}(\xi)$ and $W'(\xi) = \sum_{i=1}^{n} N'_{i,p}(\xi) w_i$.

Using the same approach adopted for B-splines, the linear combination of basis functions and control points, we arrive to a geometrical mapping for the NURBS curve:

$$\boldsymbol{C}(\xi) = \sum_{i=1}^{n} R_{i,p}(\xi) \boldsymbol{B}_i. \tag{1.14}$$

Most of the properties of NURBS are derived from the B-splines: they consitute a partition of unity, are pointwise nonnegative, and also their continuity and support properties follow from the knot vector. We also observe that B-splines are a special case of NURBS; as a matter of fact, if the weights assume the same values, then $R_{i,p}(\xi) \equiv N_{i,p}(\xi)$ and the curve is again a piecewise polynomial.

We define bivariate NURBS basis functions as

$$R_{i,k} : \hat{\Omega} \to \mathbb{R}, \quad R_{i,k}(\xi, \eta) = \frac{N_{i,p}(\xi) M_{k,q}(\eta) w_{i,k}}{\sum_{\hat{i}=1}^{n_1} \sum_{\hat{k}=1}^{n_2} N_{\hat{i},p}(\xi) M_{\hat{k},q}(\eta) w_{\hat{i},\hat{k}}}, \tag{1.15}$$

where, in order to simplify the notation, we do not indicate their polynomial degree. Finally we define a NURBS surface of degree p in the ξ parametric direction and degree q in the η parametric direction as

$$\boldsymbol{S}(\xi, \eta) = \sum_{i=1}^{n_1} \sum_{k=1}^{n_2} R_{i,k}(\xi, \eta) \boldsymbol{B}_{i,k}, \tag{1.16}$$

with $\boldsymbol{B}_{i,k}$ the control points in $\mathbb{R}^d$, $d = 2$ or 3. In order to make the notation of (1.15) simpler, in next sections we will use the following formulation for bivariate NURBS basis functions, where we have introduced a reordering of the basis functions

$$R_j(\boldsymbol{\xi}) := \frac{N_j(\boldsymbol{\xi}) w_j}{\sum_{\hat{j}=1}^{n} w_{\hat{j}} N_{\hat{j}}(\boldsymbol{\xi})}, \quad j = 1, ..., n \tag{1.17}$$

where $N_j(\boldsymbol{\xi})$ represent bivariate B-Spline basis functions defined in (1.6), $\boldsymbol{\xi} = (\xi, \eta)$, and $n = n_1 n_2$. The resulting NURBS space over the parametric domain $\hat{\Omega}$ reads as (1.12).

Assuming the notation in (1.17) we can express the NURBS surface (1.16) as a *geometrical mapping* $\boldsymbol{F}$ from the parameter domain $\hat{\Omega} = [0, 1]^\kappa$ where the NURBS basis $\{R_j\}_{j=1}^n$ is defined, to the physical space $\mathbb{R}^d$ where the set of control points $\{\boldsymbol{B}_j\}_{j=1}^n$, with $n = n_1 n_2$, is defined. It follows that

$$\boldsymbol{F} : \hat{\Omega} \subset \mathbb{R}^\kappa \to \Omega \subset \mathbb{R}^d, \quad \boldsymbol{\xi} \to \boldsymbol{F}(\boldsymbol{\xi}) = \sum_{j=1}^{n} R_j(\boldsymbol{\xi}) \boldsymbol{B}_j, \tag{1.18}$$

where $\boldsymbol{B}_j \in \mathbb{R}^d$, $j = 1, ..., n$ and $d > \kappa \geq 1$. In particular, since in this work we focus on the numerical solution of second order Partial Differential Equations (PDEs) on surfaces [DQ15], we will always assume $\kappa = 2$ and $d = 2$ or 3. When $d = \kappa = 2$, the map $\boldsymbol{F}$ describes a surface in $\mathbb{R}^2$. When $\kappa = 2$ and $d = 3$ the geometry described by $\boldsymbol{F}$ is a lower dimensional manifold with respect to the physical space, in this case a surface in R^3. It is clear that surfaces in 2D can be always seen as a particular case of surfaces in 3D when control points have a third

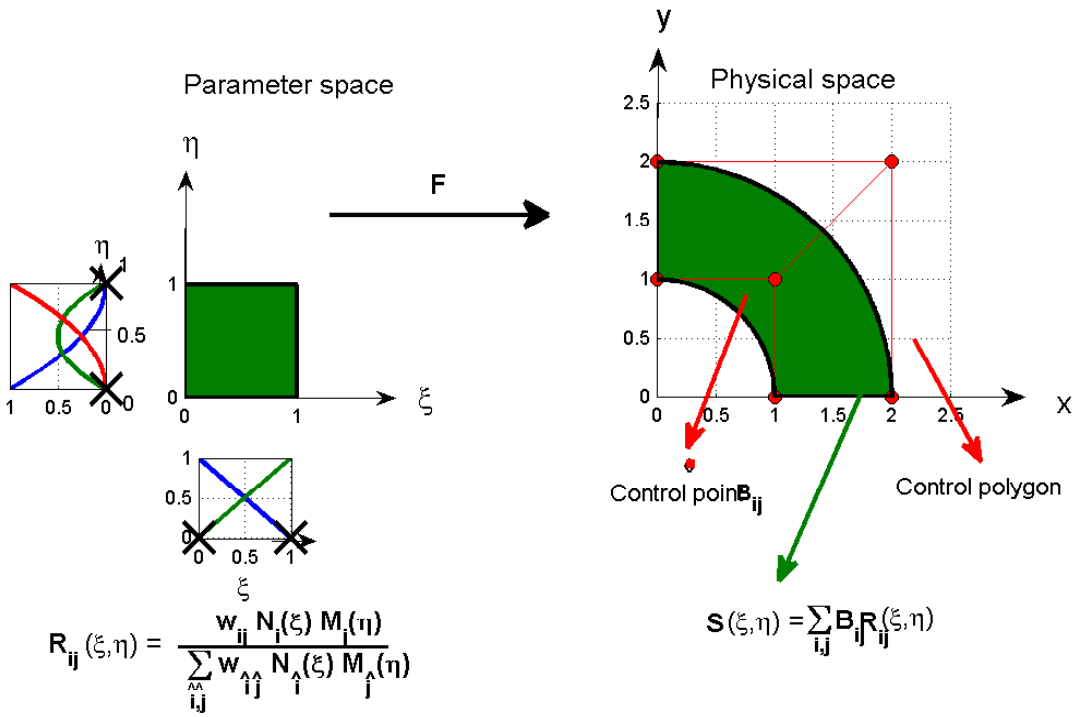


Figure 1.5: NURBS surface built as in Example 1.2

null component. In virtue of the geometrical mapping (1.18), we define the space of NURBS in the physical domain Ω as the push-forward of the space $\hat{\mathcal{B}}_h$ defined in (1.12) as

$$\mathcal{B}_h = \text{span}\{R_j \circ \boldsymbol{F}^{-1}, j = 1, ..., n\} = \text{span}\{\mathcal{R}_j, j = 1, ..., n\}. \tag{1.19}$$

We now present two examples of construction of a NURBS surfaces: in Example 1.2 we build a NURBS surface in 2D, while in Example 1.3 we build a surface in 3D.

Example 1.2. *We construct a quarter of an annulus by using the NURBS tensor product structure. We build univariate NURBS in the radial and angular directions, as already done in Fig. 1.2. The bivariate NURBS basis is defined by means of the knot vectors* $\Xi = \{0, 0, 1, 1\}$ *(for the radial direction) and* $H = \{0, 0, 0, 1, 1, 1\}$ *(for the angular direction) and weights* $w_{1,1} = w_{1,3} = 1, w_{1,2} = \sqrt{2}/2, w_{2,1} = w_{2,3} = 1, w_{2,2} = \sqrt{2}/2$*, where the subscripts refers to the parametric directions. By choosing the control points as* $\boldsymbol{B}_{1,1} = (1, 0)^T, \boldsymbol{B}_{1,2} = (1, 1)^T, \boldsymbol{B}_{1,3} = (0, 1)^T, \boldsymbol{B}_{2,1} = (2, 0)^T, \boldsymbol{B}_{2,2} = (2, 2)^T, \boldsymbol{B}_{2,3} = (0, 2)^T$ *, we obtain the desired NURBS surface shown in Fig. 1.5.*

Example 1.3. *We construct a quarter of a cylindrical shell by using the NURBS tensor product structure. We build the bivariate NURBS basis through the same structures of the parameter domain* $\hat{\Omega}$ *used in Example 1.2 to construct a 2D surface. In particular, we use the knot vectors* $\Xi = \{0, 0, 1, 1\}$ *for the axial direction and* $H = \{0, 0, 0, 1, 1, 1\}$ *for the angular direction. It follows that we have* $n = n_1 n_2 = 2 \times 3 = 6$ *basis functions. We assign weights* $w_{1,1} = w_{1,3} = 1, w_{1,2} = \sqrt{2}/2, w_{2,1} = w_{2,3} = 1, w_{2,2} = \sqrt{2}/2$*, control points* $\boldsymbol{B}_{1,1} = (0, 0, 1)^T, \boldsymbol{B}_{1,2} = (0, 1, 1)^T,$

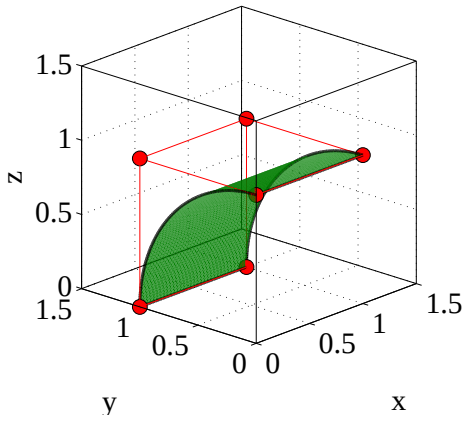


Figure 1.6: Cylindrical shell built as in Example 1.3

$\boldsymbol{B}_{1,3} = (0,1,0)^T, \boldsymbol{B}_{2,1} = (1,0,1)^T, \boldsymbol{B}_{2,2} = (1,1,1)^T, \boldsymbol{B}_{2,3} = (1,1,0)^T$ *, and obtain the desired NURBS surface shown in Fig. 1.6.*

1.3 Refinement

There exist several ways to enrich the B-splines and NURBS basis, by leaving the geometry and the parametrization unchanged in the enrichment process. We will now present the *knot insertion* (or h-refinement), the *order elevation* (or p-refinement), and the *k-refinement* methods [CHB09].

1.3.1 knot insertion (h-refinement)

The *knot insertion* is a refinement technique consisting in the insertion of one ore more knots in the knot vector. Given a knot vector, we call extended knot-vector the vector $\bar{\Xi} = \left\{\bar{\xi}_1 = \xi_1, \bar{\xi}_2, ..., \bar{\xi}_{n+m+p+1} = \xi_{n+p+1}\right\}$, such that $\Xi \subset \bar{\Xi}$. We obtain the new $n+m$ basis functions by applying the recursive Cox-de Boor recursion formula, and the new $n+m$ control points, $\bar{\mathcal{B}} = \left\{\bar{\boldsymbol{B}}_1, \bar{\boldsymbol{B}}_2, ..., \bar{\boldsymbol{B}}_{n+m}\right\}^T$, by applying a linear transformation $\boldsymbol{T}$ of the original control points $\mathcal{B} = \{\boldsymbol{B_1}, ..., \boldsymbol{B_n}\}^T$ [CHB09]. Instead of inserting new knots we could also repeat existing knots, but this will increase their multiplicity, by locally reducing the continuity of the basis functions (property P5). However the continuity of the curve will be preserved by computing the control points by means of the transformation $\boldsymbol{T}$.

1.3.2 Order elevation (p-refinement)

The *order elevation* procedure consists in increasing the polynomial degree of the basis functions used to represent the geometry. As already said, the basis functions have continuity $p - m_i$ across knots of multiplicity m_i, so if we increase the polynomial degree p, m_i must be increased as well if we want to preserve the local continuity of the basis functions. For order elevation, the multiplicity of each knot value is increased by one, but no new knot is inserted. A new set of control points is also determined. As with knot insertion, both the geometry and the parametrization are preserved.

1.3.3 k-refinement

We finally present the *k-refinement* procedure that consists in increasing the degree of a curve and subsequently realizing a knot insertion. This method is very meaningful in practice because it allows to build the geometry with the desired polynomial degree in each direction and to refine the mesh of the computational domain in order to improve the accuracy in the numerical solution of PDEs, as also the p and h-refinements do separately. It is important to note that the process of order elevation and knot insertion do not commute.

1.4 Isogeometric Analysis on surfaces

In this section we introduce Isogeometric Analysis (IGA) as a method for the numerical approximation of PDEs. In particular, in this chapter we focus on the numerical solution of second order PDEs on lower dimensional manifolds, specifically surfaces in a three dimensional space. In this context, the standard case of solution of second order PDEs on two dimensional surfaces is automatically recovered if the third components of the computational domains are null.

1.4.1 The isogeometric concept

The fundamental idea on which IGA is based is the *isogeometric concept*: the basis functions used to model the geometry are also used as basis for the approximation of the solution of PDEs computed by a suitable numerical method. In particular, in the following, we will use a NURBS-based IGA formulation in the framework of the Galerkin method for the approximation of PDEs. Given a NURBS basis $\{R_j\}_{j=1}^n$ as defined in (1.17) in the parameter domain $\hat{\Omega}$, a general real-valued function $\hat{g}$, belonging to the NURBS space $\hat{\mathcal{B}}_h$, can be represented in $\hat{\Omega}$ as

$$\hat{g} : \hat{\Omega} \to \mathbb{R}, \quad \hat{g}(\xi) \equiv \sum_{j=1}^{n} R_j(\xi)\alpha_j, \tag{1.20}$$

where coefficients $\{\alpha_j\}_{j=1}^n \subset \mathbb{R}$ are called *control variables*. Due to the non-interpolatory nature of the basis, we can not interpret the control variables as nodal values. The same function can be defined over the domain in the physical space by using the inverse of the *geometrical mapping* (1.18), that is

$$g : \Omega \to \mathbb{R}, \quad g(\boldsymbol{x}) = \hat{g}(\boldsymbol{\xi}) \circ \boldsymbol{F}^{-1}(\boldsymbol{\xi}). \tag{1.21}$$

1.4.2 NURBS as a basis for analysis

We start by introducing a problem in the physical domain, expressed in the following variational form: find $u \in V$ such that

$$a(u, v) = L(v), \quad \forall v \in V \tag{1.22}$$

where $V \subset H^1(\Omega)$ is a Hilbert space, $a : V \times V \to \mathbb{R}$ a bilinear form obtained by second order elliptic PDEs, and $L : V \to \mathbb{R}$ a linear functional. We assume that problem (1.22) is well posed and consider its Galerkin approximation: find $u^{\mathcal{N}} \in V^{\mathcal{N}}$ such that

$$a(u^{\mathcal{N}}, v^{\mathcal{N}}) = L(v^{\mathcal{N}}), \ \ \forall v^{\mathcal{N}} \in V^{\mathcal{N}}, \tag{1.23}$$

where $V^{\mathcal{N}} \subset V$, $\mathcal{N} = \dim(V^{\mathcal{N}}) < +\infty$, is a finite dimensional space approximating the infinite dimensional space V.

The adoption of an Isogeometric approach at the spatial discretization stage consists, as already specified by the isogeometric concept, in choosing the NURBS basis functions for the definition of the space $V^{\mathcal{N}}$. Let us consider the finite dimensional space $\hat{\mathcal{B}}_h = \text{span}\{R_j\}_{j\in J}$, where $R_j : \hat{\Omega} \to \mathbb{R}$ are NURBS in the parameter domain and J is the set of indexes of these functions. As consequence, the Galerkin approximation space $V^{\mathcal{N}}$ is chosen as a finite dimensional subspace of the space $\mathcal{B}_h$ of NURBS defined in (1.19) in the computational domain Ω, that is $V^{\mathcal{N}} = V \cap \mathcal{B}_h$. With this choice, as anticipated in Eq.(1.20) for a generic function, we can express the discrete solution of problem (1.23) as

$$u^{\mathcal{N}}(\boldsymbol{x}) = \sum_{j\in J:\mathcal{R}_j\in V^{\mathcal{N}}} \alpha_j \mathcal{R}_j(\boldsymbol{x}). \tag{1.24}$$

In next section we define the operators involved in the definition of a PDE problem on a lower dimensional manifold. In Section 1.4.4 we show how to realize the space discretization and how to choose the space $V^{\mathcal{N}}$ in the case of the Laplace-Beltrami equation with Dirichlet non homogeneous boundary conditions. When the computational domain is a surface in 2D, this problem can be easily reformulated as the standard Poisson problem in $\mathbb{R}^2$.

1.4.3 Functions and differential operators on manifolds

For the mapping (1.18), we introduce its jacobian:

$$\boldsymbol{J_F} : \widehat{\Omega} \to \mathbb{R}^{d\times\kappa}, \quad \boldsymbol{\xi} \to \boldsymbol{J_F}(\boldsymbol{\xi}), \quad \boldsymbol{J_F}_{i,j}(\boldsymbol{\xi}) := \frac{\partial \boldsymbol{F}_i}{\partial \boldsymbol{\xi}_j}(\boldsymbol{\xi}), \ \ i = 1, ..., d, \ \ j = 1, ..., \kappa. \tag{1.25}$$

We also define the first fundamental form of the mapping:

$$\widehat{\boldsymbol{G}} : \widehat{\Omega} \to \mathbb{R}^{\kappa\times\kappa}, \quad \boldsymbol{\xi} \to \widehat{\boldsymbol{G}}(\boldsymbol{\xi}), \quad \widehat{\boldsymbol{G}}(\boldsymbol{\xi}) := \left(\boldsymbol{J_F}(\boldsymbol{\xi})\right)^T \boldsymbol{J_F}(\boldsymbol{\xi}) \tag{1.26}$$

and its determinant as:

$$\widehat{g} : \widehat{\Omega} \to \mathbb{R}, \quad \boldsymbol{\xi} \to \widehat{g}(\boldsymbol{\xi}) \ \text{ such that } \ \widehat{g}(\boldsymbol{\xi}) := \sqrt{det\left(\widehat{\boldsymbol{G}}\left(\boldsymbol{\xi}\right)\right)}. \tag{1.27}$$

Once again, if $\kappa \equiv d$ the standard case is obtained:

$$\boldsymbol{J_F}(\boldsymbol{\xi}) \in \mathbb{R}^{d\times d}, \ \ \widehat{g}(\boldsymbol{\xi}) \equiv det\left(\boldsymbol{J_F}(\boldsymbol{\xi})\right) \ \ \text{(if positive)}. \tag{1.28}$$

In order to ensure the invertibility of the mapping in $\widehat{\Omega}$, we assume $\widehat{g}(\boldsymbol{\xi}) > 0$ in $\widehat{\Omega}$, with $\widehat{g}(\boldsymbol{\xi}) = 0$ in subsets $\widehat{Q} \subset \widehat{\Omega} \subset \mathbb{R}^\kappa$ of zero measure in the topology of $\mathbb{R}^\kappa$.

Let us consider a sufficiently regular function $\phi \in C^o(\Omega)$, for which, thanks to the invertibility of the geometrical mapping we have:

$$\phi(\boldsymbol{x}) = \widehat{\phi}(\boldsymbol{\xi}) \circ \boldsymbol{F}^{-1}(\boldsymbol{\xi}) \ \text{ and } \ \widehat{\phi}(\boldsymbol{\xi}) = \phi\left(\boldsymbol{F}(\boldsymbol{\xi})\right). \tag{1.29}$$

If $\phi \in C^1(\Omega)$, we can define its gradient on the manifold $\nabla_\Omega \phi$ and if $\phi \in C^2(\Omega)$, we can define the Laplace-Beltrami operator Δ_Ω associated to the manifold Ω. By using the geometrical mapping, we can rewrite the gradient and the Laplace-Beltrami operator on the manifol, as [DQ15]:

$$\nabla_\Omega \phi(\boldsymbol{x}) = [\boldsymbol{F}(\boldsymbol{\xi})\widehat{\boldsymbol{G}}^{-1}(\boldsymbol{\xi})\widehat{\nabla}\widehat{\phi}(\boldsymbol{\xi})] \circ \boldsymbol{F}^{-1}(\boldsymbol{\xi}), \tag{1.30}$$

$$\Delta_\Omega \phi(\boldsymbol{x}) = \left[\frac{1}{\widehat{g}(\boldsymbol{\xi})}\widehat{\nabla} \cdot \left(\widehat{g}(\boldsymbol{\xi})\widehat{\boldsymbol{G}}^{-1}(\boldsymbol{\xi})\widehat{\nabla}\widehat{\phi}(\boldsymbol{\xi})\right)\right] \circ \boldsymbol{F}^{-1}(\boldsymbol{\xi}), \tag{1.31}$$

where $\widehat{\nabla}\widehat{\phi} : \widehat{\Omega} \to \mathbb{R}^\kappa$ is the gradient operator in the parameter domain. For the differential we have $d\Omega = \widehat{g}(\boldsymbol{\xi})d\widehat{\Omega}$.

1.4.4 The Laplace-Beltrami equation

Let us consider the Laplace-Beltrami problem with Dirichlet boundary conditions defined on a lower dimensional manifold Ω, a surface in $3D$, described by a geometrical map of the form (1.18). The source function $f : \Omega \to \mathbb{R}$, $f \in L^2(\Omega)$, the diffusion coefficient $\delta \in \mathbb{R}$ assumed to be constant, are assigned. Moreover, we assume the Dirichlet datum $\gamma : \partial\Omega \to \mathbb{R}$ with $\gamma \in H^{1/2}(\partial\Omega)$. The problem reads: find $u : \Omega \to \mathbb{R}$ such that

$$-\nabla_\Omega \cdot (\delta(\boldsymbol{x})\nabla_\Omega u) = f \qquad \text{in } \Omega \tag{1.32a}$$

$$u = \gamma, \qquad \text{on } \partial\Omega \tag{1.32b}$$

In order to rewrite the problem in the variational formulation (1.22) we introduce the *test function space* V of the functions satysfing the homogeneous essential boundary conditions:

$$V = \{v \in H^1(\Omega) : v|_{\partial\Omega} = 0\} \equiv H_0^1(\Omega). \tag{1.33}$$

Since non homogeneous boundary conditions are imposed, we introduce a *lifting function* $\bar{\gamma}$ such that $\bar{\gamma} \in H^1(\Omega)$ and $\bar{\gamma}|_{\partial\Omega} = \gamma$. In this way the solution u belongs to a *trial function space* V_γ defined as:

$$V_\gamma := \{v \in H^1(\Omega) : v|_{\partial\Omega} = \gamma\} \tag{1.34}$$

and for $u \in V_\gamma$ there exists a unique $u_0 \in V$ such that $u = \bar{\gamma} + u_0$. In virtue of this substitution and assuming $\delta(\boldsymbol{x}) = \delta$ constant, the variational form of the problem reads as in (1.22) and

$$a(u_0, v) = \int_\Omega \delta \nabla_\Omega u_0 \cdot \nabla_\Omega v \, d\Omega \tag{1.35}$$

$$L(v) = \int_\Omega f v \, d\Omega - \int_\Omega \delta \nabla_\Omega \bar{\gamma} \cdot \nabla_\Omega v \, d\Omega. \tag{1.36}$$

We now approximate this problem by means of the Galerkin method as already done in (1.23). In this case the problem reads: find $u_0^{\mathcal{N}} \in V^{\mathcal{N}}$ such that

$$a(u_0^{\mathcal{N}}, v^{\mathcal{N}}) = L(v^{\mathcal{N}}), \quad \forall v^{\mathcal{N}} \in V^{\mathcal{N}}. \tag{1.37}$$

As previously anticipated, the space $V^{\mathcal{N}}$ is chosen as a finite dimensional subspace of the space V in virtue of (1.19). As a matter of fact, by exploiting the locality of the supports of NURBS basis functions, we can identify $\mathcal{N}$ basis functions that identically satisfy the homogeneous

essential boundary conditions and by reordering the indexes, we assume: $\mathcal{R}_j|_{\partial\Omega} = 0$, for $j = 1, ..., \mathcal{N}$, i.e. $\mathcal{N} = \dim(V^{\mathcal{N}})$. These functions represent a basis for the finite dimensional space and finally $V^{\mathcal{N}} = \text{span}\{\mathcal{R}_j, j = 1, ..., \mathcal{N}\}$. Since $u_0^{\mathcal{N}} \in V^{\mathcal{N}}$ we have that

$$u_0^{\mathcal{N}}(\boldsymbol{x}) = \sum_{j=1}^{\mathcal{N}} \alpha_j \mathcal{R}_j(\boldsymbol{x}). \tag{1.38}$$

Let now $u^{\mathcal{N}}$ be the discrete counterpart of u and $V_\gamma^{\mathcal{N}}$ a space such that $V_\gamma^{\mathcal{N}} := V_\gamma \cap \mathcal{B}_h$. It follows that, if $\bar{\gamma} \in V_\gamma^{\mathcal{N}}$, there exists a unique $u_0^{\mathcal{N}} \in V^{\mathcal{N}}$ such that $u^{\mathcal{N}} = \bar{\gamma} + u_0^{\mathcal{N}}$. In general, since not every function $\bar{\gamma}$ belongs to the NURBS space, we have to approximate it with a function $\bar{\gamma}^{\mathcal{N}}$ such that $\bar{\gamma}^{\mathcal{N}} \approx \bar{\gamma}$. In this way, $u^{\mathcal{N}} \in V_\gamma^{\mathcal{N}}$ is such that $u^{\mathcal{N}}|_{\partial\Omega} = \bar{\gamma}^{\mathcal{N}}|_{\partial\Omega} \approx \gamma$. In practice, for suitable coefficients $\{\bar{\gamma}_j\}_{j=1}^n$, $\bar{\gamma}^{\mathcal{N}}$ can be built as

$$\bar{\gamma}^{\mathcal{N}}(\boldsymbol{x}) = \sum_{j=\mathcal{N}+1}^{n} \mathcal{R}_j(\boldsymbol{x})\bar{\gamma}_j. \tag{1.39}$$

Notice that $\bar{\gamma}_1 = ... = \bar{\gamma}_{\mathcal{N}} = 0$ as they have no effect on its value on $\partial\Omega$ since $\{\mathcal{R}_j\}_{j=1}^{\mathcal{N}} = 0$.

By substituting expressions (1.38) and (1.39) in (1.37) and testing against every basis function, we obtain the system of equations:

$$\sum_{j=1}^{\mathcal{N}} \int_\Omega \delta\alpha_j \nabla_\Omega \mathcal{R}_j \cdot \nabla_\Omega \mathcal{R}_i d\Omega = \int_\Omega f\mathcal{R}_i d\Omega - \int_\Omega \delta\nabla_\Omega \bar{\gamma}^{\mathcal{N}} \cdot \nabla_\Omega \mathcal{R}_i d\Omega \quad i = 1, ..., \mathcal{N} \tag{1.40}$$

that can be written as a linear system:

$$\boldsymbol{A}\boldsymbol{\alpha} = \boldsymbol{L}, \tag{1.41}$$

where $\boldsymbol{A} \in \mathbb{R}^{\mathcal{N}\times\mathcal{N}}$ is the *stiffness matrix*, and $\boldsymbol{L} \in \mathbb{R}^{\mathcal{N}}$ is the *right-hand side vector*. In particular, the coefficients $\boldsymbol{A}_{ij}$ of the stiffness matrix are given by

$$\boldsymbol{A}_{ij} = \int_\Omega \delta\nabla_\Omega \mathcal{R}_j \cdot \nabla_\Omega \mathcal{R}_i d\Omega, \tag{1.42}$$

and the coefficients $\boldsymbol{L}_i$, of the right-hand side are given by

$$\boldsymbol{L}_i = \int_\Omega f\mathcal{R}_i d\Omega - \int_\Omega \delta\nabla_\Omega \bar{\gamma}^{\mathcal{N}} \cdot \nabla\mathcal{R}_i d\Omega. \tag{1.43}$$

Due to the locality of the supports of NURBS basis functions, $\boldsymbol{A}$ is a sparse banded matrix. As a matter of fact, although NURBS basis functions have support over much larger portions of the domain than classical FEA functions do (see P1, section 1.1.2), any basis function shares its support with $2p + 1$ basis functions, as for the Lagrangian polynomial basis functions used in FEA (see P2, section 1.1.2). As consequence, the larger support of the B-spline basis functions, does not have any implication on the bandwidth of the stiffness matrix $\boldsymbol{A}$. By solving system (1.41), we obtain the vector $\boldsymbol{\alpha}$ and by substituting the $\{\alpha_j\}_{j=1}^{\mathcal{N}}$ in (1.38), we can finally write the solution $u^{\mathcal{N}}$ as

$$u^{\mathcal{N}} = \sum_{j=1}^{\mathcal{N}} \alpha_j \mathcal{R}_j + \sum_{j=\mathcal{N}+1}^{n} \bar{\gamma}_j \mathcal{R}_j. \tag{1.44}$$

The evaluation of the NURBS basis functions, of their derivatives, and of the other data in points of the physical domain Ω is performed by evaluating their counterpart in the parameter domain $\hat{\Omega}$. By using the operators defined in Section 1.4.3, the integrals 1.42 and 1.43 can be rewritten in the parameter domain as follows:

$$\hat{\boldsymbol{A}}_{ij} = \int_{\hat{\Omega}} \delta\hat{\nabla} R_j \cdot \left(\hat{\boldsymbol{G}}^{-1}\hat{\nabla} R_i\right) \hat{g} d\hat{\Omega}, \tag{1.45a}$$

$$\hat{\boldsymbol{L}}_i = \int_{\hat{\Omega}} \hat{f} R_i \hat{g} d\hat{\Omega} - \int_{\hat{\Omega}} \delta\hat{\nabla}\hat{\bar{\gamma}}^{\mathcal{N}} \cdot \hat{\nabla} R_i d\hat{\Omega}, \tag{1.45b}$$

where $\hat{f}(\boldsymbol{\xi}) = f(\boldsymbol{F}(\boldsymbol{\xi}))$ and $\hat{\bar{\gamma}}^{\mathcal{N}}(\boldsymbol{\xi}) = \bar{\gamma}^{\mathcal{N}}(\boldsymbol{F}(\boldsymbol{\xi}))$. Both the integrals (1.45a) and (1.45b) are numerically approximated by a suitable quadrature rule (e.g. Gauss-Legendre). Let us introduce the set $\widehat{K}_h = \left\{\widehat{K}_k\right\}_{k=1}^{N_e}$ of non-overlapping mesh elements that is a partition of the parametric domain $\widehat{\Omega}$. Let us remark that these are generally taken as the knot spans of non zero size. Through the parametrization $\mathbf{F}$ we simply obtain a partition of the physical domain:

$$\Omega = \bigcup_{k=1}^{N_e} \mathbf{F}(\widehat{K}_k) = \bigcup_{k=1}^{N_e} K_k. \tag{1.46}$$

We will refer to $\widehat{K}_k$ and K_k as the mesh *elements* and we will define a quadrature rule on every element $\widehat{K}_k$. In ordet to do that, we introduce, for each of these rules, the set of Gauss-Legendre *quadrature nodes* n_k [DR75]

$$\{\boldsymbol{\xi}_{l,k}\} \subset \widehat{K}_k, \quad l = 1, ..., n_k \tag{1.47}$$

and *quadrature weights*

$$\{q_{l,k}\} \subset \mathbb{R}, \quad l = 1, ..., n_k \tag{1.48}$$

where k is the elements index, and $n_k = (p+1)(q+1)$, with p and q the polynomial degrees of the univariate B-splines basis functions in ξ and η parametric directions, respectively. The approximation rule for a function $\phi \in L^1(K_k)$ is

$$\int_{K_k} \phi \, d\Omega = \int_{\widehat{K}_k} \phi(\mathbf{F}(\boldsymbol{\xi}))|\hat{g}(\boldsymbol{\xi})| d\hat{\Omega} \approx \sum_{l=1}^{n_k} q_{l,k}\phi(\boldsymbol{x}_{l,k})|\hat{g}(\boldsymbol{\xi}_{l,k})|, \tag{1.49}$$

where $\boldsymbol{x}_{l,k} = \boldsymbol{F}(\boldsymbol{\xi}_{l,k})$ are the images of the nodes in the physical space. We can now numerically compute the $\hat{\boldsymbol{A}}_{ij}$ and $\hat{\boldsymbol{L}}_i$ coefficients:

$$\hat{\boldsymbol{A}}_{ij} = \int_{\hat{\Omega}} \delta\hat{\nabla} R_j \cdot \left(\hat{\boldsymbol{G}}^{-1}\hat{\nabla} R_i\right) \hat{g} d\hat{\Omega} \tag{1.50}$$

$$\approx \sum_{k=1}^{N_e}\sum_{l=1}^{n_k} \delta \; q_{l,k}\hat{\nabla} R_j(\boldsymbol{\xi}_{l,k}) \cdot \left(\hat{\boldsymbol{G}}^{-1}\hat{\nabla} R_i(\boldsymbol{\xi}_{l,k})\right) \hat{g}(\boldsymbol{\xi}_{l,k})|\hat{K}_k| \tag{1.51}$$

$$\hat{\boldsymbol{L}}_i = \int_{\hat{\Omega}} \hat{f}(\boldsymbol{\xi}) R_i(\boldsymbol{\xi}) \hat{g}(\boldsymbol{\xi}) d\hat{\Omega} - \int_{\hat{\Omega}} \delta \hat{\nabla} \hat{\tilde{\gamma}}^{\mathcal{N}}(\boldsymbol{\xi}) \cdot \hat{\nabla} R_i(\boldsymbol{\xi}) d\hat{\Omega} \tag{1.52}$$

$$\approx \sum_{k=1}^{N_e} \sum_{l=1}^{n_k} q_{l,k} \hat{f}(\boldsymbol{\xi}_{l,k}) R_i(\boldsymbol{\xi}_{l,k}) \hat{g}(\boldsymbol{\xi}_{l,k}) |\hat{K}_k| \tag{1.53}$$

$$- \sum_{k=1}^{N_e} \sum_{l=1}^{n_k} \delta q_{l,k} \hat{\nabla} \hat{\tilde{\gamma}}^{\mathcal{N}}(\boldsymbol{\xi}_{l,k}) \cdot \hat{\nabla} R_i(\boldsymbol{\xi}_{l,k}) |\hat{K}_k| \tag{1.54}$$

When the computational domain Ω is a surface in 2D, problem (1.32) assume the classical formulation of the Poisson problem: fund $u : \Omega \to \mathbb{R}$ such that

$$-\nabla \cdot (\delta(\boldsymbol{x}) \nabla u) = f \qquad \text{in } \Omega \tag{1.55a}$$

$$u = \gamma, \qquad \text{on } \partial\Omega. \tag{1.55b}$$

In this case, the space discretization procedure and the variational formulation of the problem are the same seen in this section for a problem defined on a lower dimensional manifold, with the difference that the operators involved in the formulation degenerate in the ones reported in Section 1.4.3 for the standard case $\kappa = d = 2$.

1.5 Isogeometric Analysis for the solution of benchmark problems

Example 1.4. *We consider the numerical solution of the Laplace-Beltrami problem on a surface in $\mathbb{R}^3$, we evaluate the errors between the numerical and exact solution in norm $L^2(\Omega)$, and we estimate the convergence orders of the error for the NURBS basis of degrees $p = 2$ and $p = 3$. We recall the definition of the error in norm $L^2(\Omega)$:*

$$||u - u^{\mathcal{N}}||_{L^2(\Omega)} = \left(\int_{\Omega} (u - u^{\mathcal{N}})^2 d\Omega \right)^{1/2}. \tag{1.56}$$

We consider as computational domain Ω a quarter of a cylindrical surface by building the geometry with $p = 1$ in axial direction and $q = 2$ in angular direction (Fig. 1.7). We perform a k-refinement: first we realize a p-refinement by increasing p of one degree, so that $p = q = 2$, then we realize an h-refinement to increase the number of the mesh elements. The resulting basis functions are C^1 continuous. We choose $\delta = 1$, $\gamma = 0$, and f such that the exact solution is $u = 2xy \sin(5\pi z)$. Numerical integration is performed by means of a quadrature formula with $(p+1)(q+1)$ quadrature points in each mesh element. In Fig. 1.8 we report the exact solution and in Fig. 1.9 we report the numerical solution for different meshes. Then, starting again from the geometry in Fig. 1.7, we perform a new k-refinement: first we increase p of two orders and q of one order so that $p = q = 3$, then we perform the same h-refinement as before, and solve the same problem. In this case the basis functions are C^2 continuous. In Fig. 1.10 for both the case $p = q = 2$ and $p = q = 3$, we verify that the convergence rate of the errors in norm $L^2(\Omega)$ is $p+1$, according to the theory [DQ15]. In Table 1.1 we report the data concerning the computational times requested to assemble and solve system (1.41) *and the errors in norm $L^2(\Omega)$ for different mesh sizes and for the polynomial degrees considered.*

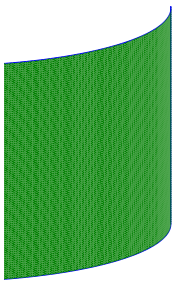


Figure 1.7: Example 1.4. A quarter of a cylindrical NURBS surface.

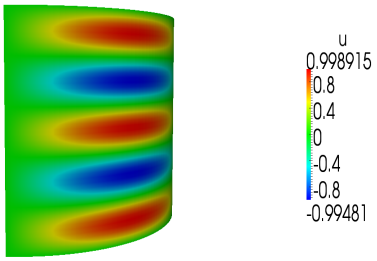
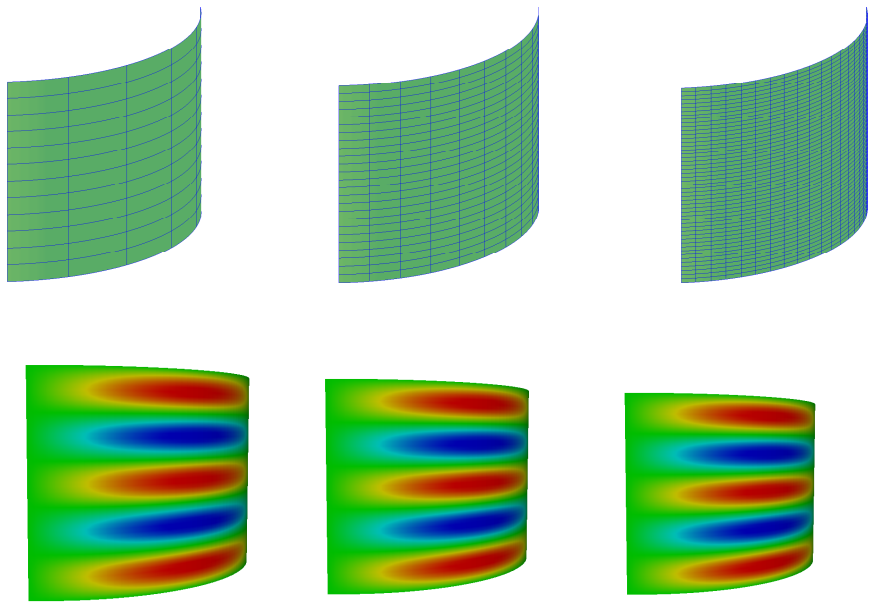


Figure 1.8: Example 1.4. Exact solution for Laplace-Beltrami problem.

Example 1.5. *We now compute the numerical solution of the Poisson problem (1.55) by considering as computational domain the quarter of annulus built in Example 1.2 and shown in Fig. 1.5. We also evaluate the errors between the numerical and exact solutions and estimate the convergence orders of these errors for the polynomial degrees $p = 2$ and 3 of the NURBS basis functions. We recall the definition of the error in $H^1(\Omega)$ norm as:*

$$||u - u^{\mathcal{N}}||_{H^1(\Omega)} = \left(||u - u^{\mathcal{N}}||^2_{L^2(\Omega)} + \int_\Omega |\nabla_\Omega (u - u^{\mathcal{N}})|^2 d\Omega \right)^{1/2}, \tag{1.57}$$

for the definition of the error in norm $L^2(\Omega)$ see (1.56). In order to improve the accuracy of the approximate solution we enrich the space $\mathcal{B}_h$, introduced in (1.19), by performing a k-refinement. In particular, referring to Example (1.2), where we have degrees $p = 1$ in radial direction and $q = 2$ in angular direction, we increase p of one degree in order to have $p = q = 2$, and then perform an h-refinement to increase the number of the mesh elements and obtain C^1 basis functions. We choose $\delta = 1$, $\gamma = 0$, and f such that the exact solution of the problem reads $u = \sin(\frac{\pi}{3}(x^2 + y^2 - 1))$. Numerical integration is performed by means of a quadrature formula with $(p + 1)(q + 1)$ quadrature points in each mesh element. In Fig. 1.11 we report the exact solution, while in Fig. 1.12 we report the numerical solutions for different meshes. Then, in order to see how the solution approximation improves by increasing the polynomial degree, always starting from the geometry of Fig. 1.5, we perform a new k-refinement. In this case, we increase the polynomial degree p (in radial direction) of 2 and the



1.9.4 $u^{\mathcal{N}}, N_{el} = 60$ 1.9.5 $u^{\mathcal{N}}, N_{el} = 250$ 1.9.6 $u^{\mathcal{N}}, N_{el} = 1000$

Figure 1.9: Example 1.4. Laplace-Beltrami problem on a quarter of a cylinder surface. Refined meshes (top), corresponding numerical solutions $u^{\mathcal{N}}$ (bottom).

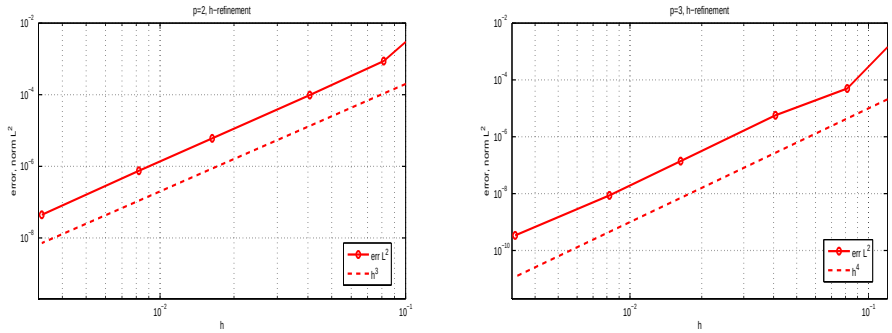


Figure 1.10: Example 1.4. Convergenge of errors in norm $L^2(\Omega)$ and reference convergence rate $p+1$ vs. the mesh size h for $p = 2$ (left) and $p = 3$ (right).

polynomial degree q (in angular direction) of 1, and so obtain the desired $p = q = 3$. We then perform, also in this case, an h-refinement. In this case the resulting basis functions are C^2 continuous. In Fig. 1.10 for both the cases $p = 2$ and $p = 3$, we verify that the convergence rates of the errors in norm $L^2(\Omega)$ and $H^1(\Omega)$ are $p + 1$ and p respectively, according to the theory [DQ15]. In Table 1.2 we report the data concerning the computational times requested to assemble and solve system (1.41) *and the errors in norm L^2 and H^1 for different mesh sizes and for the polynomial degrees considered.*

Table 1.1: *Example 1.4. Computational times requested to assemble and solve system* (1.41) *for the Laplace-Beltrami problem and errors in norm $L^2(\Omega)$ for different mesh sizes and polynomial degrees 2 and 3.*

Degree	N_{el}	$\mathcal{N}$	h	t [s]	$\|\|u-u_h\|\|_{L^2(\Omega)}$
P2	60	98	0.1309	0.25	$1.15 \cdot 10^{-2}$
	250	324	0.0814	0.51	$8.68 \cdot 10^{-4}$
	1000	1144	0.0407	1.66	$9.75 \cdot 10^{-5}$
	6250	6604	0.0163	10.34	$6.04 \cdot 10^{-6}$
	25000	25704	0.0082	57.54	$7.52 \cdot 10^{-7}$
P3	60	120	0.1309	0.38	$3.02 \cdot 10^{-3}$
	250	364	0.0814	0.81	$1.03 \cdot 10^{-5}$
	1000	1219	0.0407	2.84	$5.68 \cdot 10^{-6}$
	6250	6784	0.0163	21.06	$1.39 \cdot 10^{-7}$
	25000	26059	0.0082	115.93	$8.69 \cdot 10^{-9}$

Table 1.2: *Example 1.5. Computational times requested to assemble and solve system* (1.41) *for the Poisson problem of Example 1.55 and errors in norms $L^2(\Omega)$ and $H^1(\Omega)$ for different mesh sizes and polynomial degrees 2 and 3.*

Degree	N_{el}	$\mathcal{N}$	h	t [s]	$\|\|u-u_h\|\|_{L^2(\Omega)}$	$\|\|u-u_h\|\|_{H^1(\Omega)}$
P2	60	98	0.1309	0.345	$3.61 \cdot 10^{-3}$	$9.70 \cdot 10^{-2}$
	250	324	0.0814	0.63	$2.55 \cdot 10^{-4}$	$1.18 \cdot 10^{-2}$
	1000	1144	0.0407	1.29	$3.79 \cdot 10^{-5}$	$5.62 \cdot 10^{-3}$
	6250	6604	0.0163	8.22	$2.32 \cdot 10^{-6}$	$8.90 \cdot 10^{-4}$
	25000	25704	0.0082	47.45	$2.87 \cdot 10^{-7}$	$2.22 \cdot 10^{-4}$
P3	60	120	0.1309	0.40	$5.82 \cdot 10^{-4}$	$1.95 \cdot 10^{-2}$
	250	364	0.0814	0.76	$3.81 \cdot 10^{-5}$	$2.49 \cdot 10^{-3}$
	1000	1219	0.0407	2.03	$2.51 \cdot 10^{-6}$	$3.26 \cdot 10^{-4}$
	6250	6784	0.0163	14.67	$6.65 \cdot 10^{-8}$	$2.16 \cdot 10^{-5}$
	25000	26059	0.0082	87.57	$4.21 \cdot 10^{-9}$	$2.73 \cdot 10^{-6}$

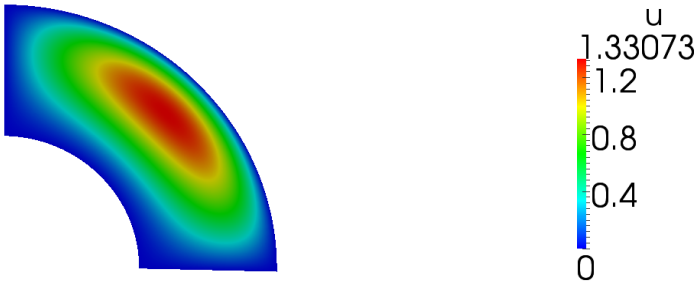
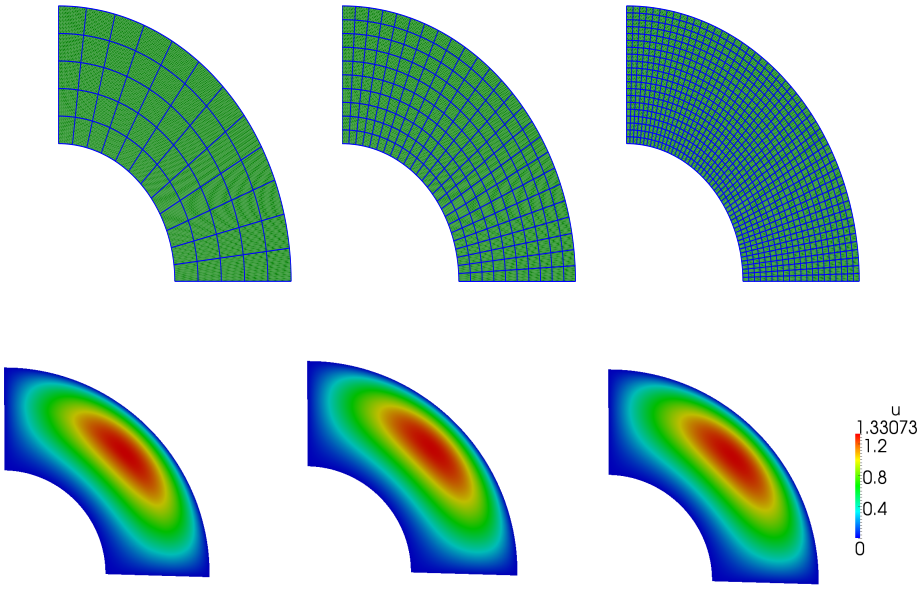


Figure 1.11: Example 1.5. Exact solution for Poisson problem.



1.12.4 $u^{\mathcal{N}}, N_{el} = 60$ 1.12.5 $u^{\mathcal{N}}, N_{el} = 250$ 1.12.6 $u^{\mathcal{N}}, N_{el} = 1000$

Figure 1.12: Example 1.5. Poisson problem on a quarter of an annulus. Refined meshes (top), corresponding numerical solutions $u^{\mathcal{N}}$ (bottom).

1.6 Isogeometric Analysis for parametrized domains

In the following chapters we are interested in taking into account shape variations of the computational domains, for example for the geometries considered in Examples 1.2 and 1.3. To this end, we will introduce some parameters in the definition of the geometries. Since the geometrical map (1.18) involves the control points and the NURBS functions, for both the 2D surfaces or the manifolds, the parametrization can concern the control points coordinates, the NURBS weights, or both the coordinates and weights. In Example 1.6, for instance, we build a 2D surface such that some of its control points coordinates are parameter dependent. There is nothing, of course, to prevent us from parametrizing its NURBS weights or both its coordinates and weights. In Example 1.7, for instance, we consider a surface in 3D such that some of its weights are parametrized, while in Example 1.8 we assume that both control

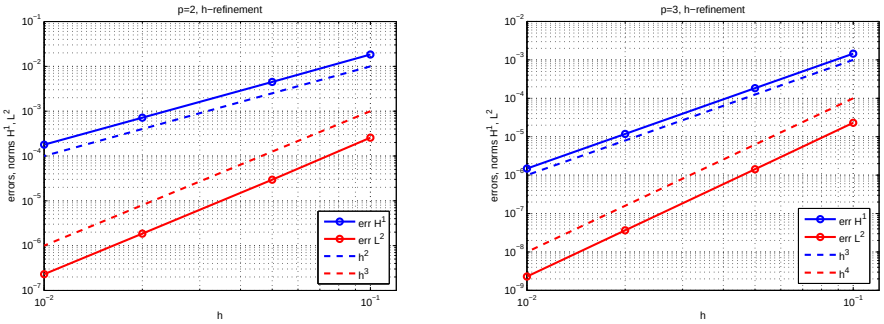
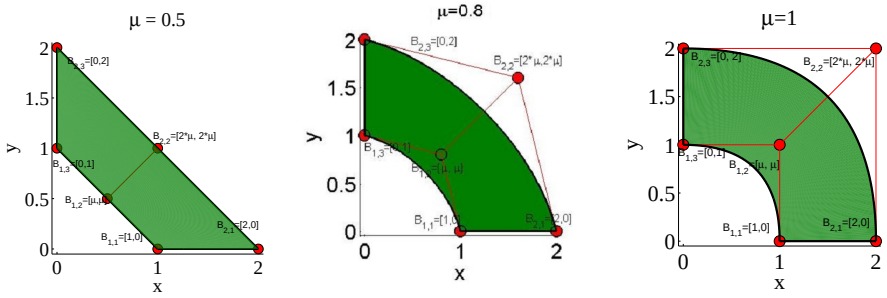


Figure 1.13: Example 1.5. Convergenge of errors in norms $L^2(\Omega)$ and $H^1(\Omega)$ and reference convergence rates p and $p+1$ vs. the mesh size h for $p=2$ (left) and $p=3$ (right).

points coordinates and NURBS weights are parameter dependent.

Example 1.6. *We consider the quarter of annulus built in Example 1.2 and depicted in Fig. 1.5 and modify it by introducing a geometric parameter μ. Let us make, for instance, the control points $\boldsymbol{B}_{1,2}$ and $\boldsymbol{B}_{2,2}$ parameter dependent. We assume that their coordinates are parametrized by means of a parameter μ that can take values in a prescribed range, $\mu \in [0.5, 1]$. In particular, μ refers to the horizontal and vertical coordinates of these control points, as $\boldsymbol{B}_{1,2} = [\mu, \mu]$ and $\boldsymbol{B}_{2,2} = [2\mu, 2\mu]$. Moreover, just in order to show an example regarding a B-spline surface, we consider the NURBS weights to be all equal to each other by setting $w_{i,j} = 1$, for $1 \leq i, j \leq 2$. In Fig. 1.14 we report the geometric configurations for $\mu = 0.5$, 0.8, 1.*



1.14.1 $\mu = 0.5$ 1.14.2 $\mu = 0.8$ 1.14.3 $\mu = 1$

Figure 1.14: Example 1.6. B-spline 2D geometries with parametrized control points coordinates.

Example 1.7. *We consider the cylindrical shell built in Example 1.3 and depicted in Fig. 1.6. Let us make the NURBS weights associated to the points $\boldsymbol{B}_{1,2}$ and $\boldsymbol{B}_{2,2}$ to be equal to a parameter $\mu \in [0.5, 1.2]$; the other weights are equal to 1. In Fig. 1.15 we report three different geometric configurations, for instance for $\mu = 0.5$, 0.8, and 1.2.*

Example 1.8. *We consider the cylindrical shell built in Example 1.3 and depicted in Fig. 1.6. Let us make the coordinates of the control points $\boldsymbol{B}_{1,3}$ and $\boldsymbol{B}_{2,3}$, and the weights associated to the control points $\boldsymbol{B}_{1,2}$ and $\boldsymbol{B}_{2,2}$ parameter dependent. To do that, we take a parameter*

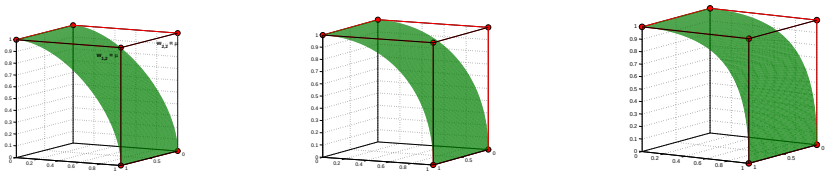


Figure 1.15: Example 1.7. NURBS manifold with parametrized weights; from left to right $\boldsymbol{\mu} = 0.5, 0.8, 1.2$.

$\boldsymbol{\mu} = (\mu_1, \mu_2)$ *such that* $\boldsymbol{B}_{1,3} = [0, 0, \mu_1]$, $\boldsymbol{B}_{2,3} = [1, 0, \mu_1]$, *while the weights of* $\boldsymbol{B}_{1,2}$ *and* $\boldsymbol{B}_{2,2}$ *are set to be* μ_2*; the other weights are equal to 1. In Fig. 1.16 we report three different geometric configurations, for instance for* $\boldsymbol{\mu} = (1, 0.5)$, $\boldsymbol{\mu} = (1.5, 0.8)$, $\boldsymbol{\mu} = (2, 1.2)$.

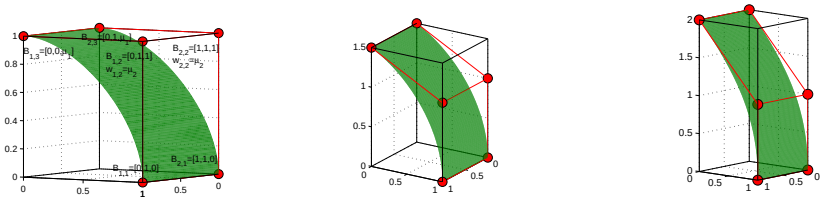


Figure 1.16: Example 1.8. NURBS manifold with parametrized control points coordinates and weights; from left to right $\boldsymbol{\mu} = [1, 0.5]$, $\boldsymbol{\mu} = [1.5, 0.8]$, $\boldsymbol{\mu} = [2, 1.2]$.

We now suppose to be interested in solving a specific problem on different geometric configurations of a computational domain. In order to start from the simplest situation, we consider a geometric parametrization that involves only the control points coordinates. In Example 1.9, for instance, we suppose to be interested in solving the Poisson problem on the parametrized geometry built in Example 1.6.

Example 1.9. *We are interested in solving the Poisson problem* (1.55) *with* $f = 1$ *and* $\gamma = 0$*, for different geometric configurations following Example 1.6. It is clear that, for each parameter sample, it is more convenient to modify the position of the interested control points before performing a p or a h-refinement. In this case for example, for each parameter* μ *we modify the geometry, and then perform a k-refinement such that* $p = q = 2$ *and the total number of elements is 1500. In Fig. 1.17 we report the numerical solutions for the configurations corresponding to* $\mu = 0.5$*, 0.8, 1. In Table 1.3 we report the computational times requested, for a specific parameter sample, to modify the computational domain, assemble the stiffness matrix and the right hand side vector, apply the boundary conditions by means of a* L^2*-projection method, and solve the sytstem, for a mesh of 1500 elements and polynomial degrees 2, 3, and 4 (we refer to these cases as to P2, P3, and P4, respectively). It is clear that, since by performing a k-refinement the number of degrees of freedom increases, the computational time requested to perform the actions listed before increases as well, and would be higher if we make the polynomial degree higher or the mesh finer.*

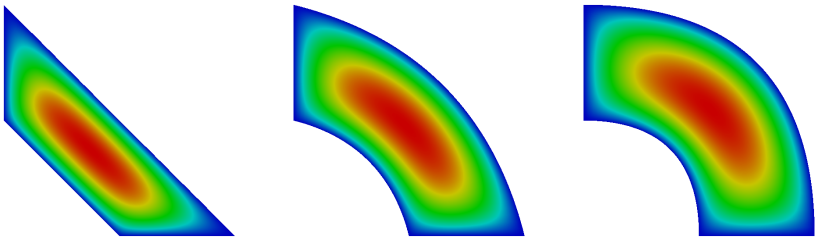


Figure 1.17: Numerical solutions for Poisson problem of Example 1.9, for polynomial degrees $p = q = 2$ on a grid of 1500 elements, for the three geometrical configurations in Fig. 1.14 correspondig to $\mu = 0.5, 0.8, 1$.

Table 1.3: *computational times requested, for a specific parameter sample, to modify the computational domain, assemble the stiffness matrix and the right hand side vector, apply the boundary conditions L^2-projection method, and solve the sytstem, for a mesh of 1500 elements and polynomial degrees 2, 3, and 4 (we refer to these cases as to P2, P3, and P4, respectively).*

Degree	P2	P3	P4
$\mathcal{N}$	1500	1586	1674
Modify geometry	0.20 s	0.24 s	0.37 s
Assemble Matrix and rhs	2.75 s	5.27 s	10.71 s
Apply BCs	0.71 s	0.99 s	1.53 s
Solve linear system	0.04 s	0.04 s	0.06 s
Total time	3.70 s	6.54 s	12.66 s

Similarly to what done in Example 1.9, we could solve the Poisson problem on 2D surfaces parametrized with respect to the NURBS weights or both control points coordinates and weights, or solve the Laplace-Beltrami problem on a surface in 3D, also in this case parametrized with respect to the control points coordinates, the weights, or both the coordinates and weights. In all these cases the problem would be that, as anticipated in Example 1.9, if we are interested in evaluating the solution of the problem for a large number of parameters $\boldsymbol{\mu}$, the total time requested for the evaluations could become extremely high and so computationally prohibitive. That is why in Chapter 2 we appeal to the Reduced Basis Method (RB), a strategy for the rapid and reliable solution of Parametrized Partial Differential Equations (PPDEs) allowing large computational savings. In Chapters 3 and 4 we will finally show how IGA and RB can be jointly used for the practical resolutions of PPDEs. In particular, we will see that in order to make possible the reduction procedure, we will have to distinguish and treat differently the case of NURBS with parametrized control points and the one of NURBS with parametrized weight (or both control points and weights).

Chapter 2

Reduced Basis method for parametrized elliptic PDEs

In this chapter we introduce the reduced basis (RB) approximation as a technique for the *rapid* and *reliable* computation of the solutions of elliptic partial differential equations (PDEs), both with affine or non-affine parameter dependence. Reduction strategies can be crucial in applications of high complexity. As a matter of fact, although the increasing computer power allows nowadays to solve problems of very large dimensions that model complex phenomena, a computational reduction is still necessary for *real-time* simulations and *many-query* contexts [GP05, Man12, Qua14, QRM11, SVH$^+$06]. Both these situations are common in Engineering practice and analysis that requires the prediction of some quantities of interest, as in design, optimization, and control contexts. These quantities of interest, called *outputs*, can be related for example to energies, forces, stresses, strains, flowrates, pressure drops, temperatures or fluxes. These outputs are usually expressed as functionals of *field variables* associated with a parametrized partial differential equation (PPDE) which describes a particular phenomenon. The parameters, which we shall denote as *inputs*, are related to geometry, physical properties, boundary conditions, or loads, and identify a particular configuration of the component. In this work, as anticipated at the end of Chapter 1, we are particularly interested in the evaluation of the field variables, solutions of PDEs *parametrized* with respect to the input parameter p-vector $\boldsymbol{\mu}$ belonging to the *input-parameter* domain $\mathcal{D} \subset \mathbb{R}^p$ and which represents geometric parametrizations of the domain.

In Section 2.1 we introduce the main features of RB and provide an overview of the contents of the rest of the chapter. In Sections 2.2 and 2.3, we present more in detail the setting of the parametrized elliptic PDEs we are going to consider in this work. In Section 2.4 we describe the main steps and theoretical aspects of the RB method. Finally, in Section 2.5 we recall an *a posteriori* error estimation, a fundamental ingredient for the application of the RB method and for the certification of the solutions provided by this method. In Section 2.6 we consider RB for parametrized surfaces.

2.1 Reduced basis: a general framework

In this Chapter and in the rest of this work, we will be interested in the evaluation of the solutions of PDEs *parametrized* with respect to an input parameter vector $\boldsymbol{\mu} \in \mathcal{D} \subset \mathbb{R}^p$, which represents geometric parametrizations of the domain. We will thus deal with a relationship

$\boldsymbol{\mu} \to u(\boldsymbol{\mu})$, whose evaluation requires the solution of a PPDE. In the elliptic case, for instance, the problem reads: given $\boldsymbol{\mu} \in \mathcal{D}$, find $u(\boldsymbol{\mu}) \in V$ such that

$$a(u(\boldsymbol{\mu}), v; \boldsymbol{\mu}) = f(v; \boldsymbol{\mu}) \quad \forall v \in V, \tag{2.1}$$

where V is a suitable Hilbert space, while $a(\cdot, \cdot; \boldsymbol{\mu})$ and $f(\cdot; \boldsymbol{\mu})$, which are parameter dependent, are the bilinear form and the linear functional associated to the PDE, respectively.

As already said, the repeated solution of problem (2.1) for several different parameters $\boldsymbol{\mu}$ may be computationally prohibitive, thus requiring a suitable model order reduction strategy. The idea behind the reduced basis approximation technique is that the field variable $u(\boldsymbol{\mu})$, that actually belongs to the infinite dimensional function space V associated with the PDE, can be assumed to reside on a *low-dimensional* and *smooth* parametrically induced manifold [RHP08]

$$\mathcal{M} = \{u(\boldsymbol{\mu}) \in V : \boldsymbol{\mu} \in \mathcal{D}\}. \tag{2.2}$$

In general, we cannot find the exact solution of the PPDE, so we replace it with a discrete approximation for example through a Galerkin method, as the FEA one or IGA, and look for this discrete solution in a subspace $V^{\mathcal{N}} \subset V$ of dimension $\mathcal{N} < +\infty$. By supposing to solve problem (2.1) for each $\boldsymbol{\mu} \in \mathcal{D}$, we can define an approximation $\mathcal{M}^{\mathcal{N}}$ of the manifold $\mathcal{M}$

$$\mathcal{M}^{\mathcal{N}} = \left\{u^{\mathcal{N}}(\boldsymbol{\mu}) \in V^{\mathcal{N}} : \boldsymbol{\mu} \in \mathcal{D}\right\}. \tag{2.3}$$

The problem is that, in order to get a high fidelity *truth* approximation $u^{\mathcal{N}}(\boldsymbol{\mu})$ for $u(\boldsymbol{\mu})$, $\mathcal{N}$ must be chosen very large, and consequently the evaluation $\boldsymbol{\mu} \to u^{\mathcal{N}}(\boldsymbol{\mu})$ becomes too expensive in the already cited real-time and many-query contexts. To avoid this drawback, we adopt a RB approach and, in this way, we build an approximation $u^N(\boldsymbol{\mu})$ for $u^{\mathcal{N}}(\boldsymbol{\mu})$ of dimension $N \ll \mathcal{N}$. To this end, we first build $\mathcal{V}^N$, a low-dimensional approximation of the manifold $\mathcal{V}^{\mathcal{N}}$. We do this by selecting a certain number of parameters from the domain $\mathcal{D}$ and compute the corresponding discrete solutions that can be seen as *snapshots* of the *truth* manifold $\mathcal{M}^{\mathcal{N}}$; Then, according to the problem, we define V_N, a proper subspace of $\mathcal{V}^{\mathcal{N}}$ of dimension N, through a linear combination of these precomputed snapshots, and look here for the reduced problem solution $u_N(\boldsymbol{\mu})$. In this way, through the Galerkin method we approximate $u(\boldsymbol{\mu})$ with $u^{\mathcal{N}}(\boldsymbol{\mu})$ and, in turn, through the RB method we approximate $u^{\mathcal{N}}(\boldsymbol{\mu})$ with $u_N(\boldsymbol{\mu})$, where $u_N(\boldsymbol{\mu}) \in V_N$ and $N \ll \mathcal{N}$. This evident *dimensional reduction*, in conjunction with the adoption of an *Offline-Online* procedure that for each $\boldsymbol{\mu}$ allows to solve a problem of complexity independent on $\mathcal{N}$, leads to large computational savings in the class of problems listed before.

In summary, the main components of the RB approach, that we will explore in depth in the next paragraphs, are [QRM11, RHP08, SVH$^+$06]:

(i) a rapidly convergent global reduced basis approximation (Galerkin projection) onto a space spanned by solutions of a PPDE computed in N properly selected parameter values of the parameter domain;

(ii) a rigorous *reduced* error estimation for the problem solution, used for both the reduced basis selection and the certification of the solution;

(iii) an Offline-Online computational procedure that consists of two different phases: an expensive (with complexity depending on $\mathcal{N}$) Offline phase in which we generate a small

reduced basis space; an Online phase (with complexity depending on N) in which, for each new parameter, we rapidly evaluate the solution of the corresponding RB problem and the associated *a posteriori* error bound.

In the following, we will consider coercive linear elliptic partial differential equations, with affine or non-affine parameter dependence; for parabolic problems, see [GP05, NRP09], for nonlinear problems, including the incompressible Navier-Stokes equations [VPRP03, CVP05, VPP03, VP05], and for non-coercive problems [VPRP03].

In this section we refer to a generic Galerkin approximation method for the truth problem. As anticipated, in several works a finite element discretization approach has been adopted (e.g. [Man12, RHP08, VPRP03]). Also spectral elements [LMR06], and finite volumes [HO08] methods have been considered. In this thesis, we will build a *RB approximation* upon a *IGA approximation*, that means that the space $V^{\mathcal{N}}$ will be chosen, according to the problem, as a proper subset of the space of NURBS $\mathcal{B}_h$ defined in (1.19), the same NURBS functions used to build the computational domain. In this respect, as anticipated at the end of Chapter 1, we will consider geometrical parameters $\boldsymbol{\mu}$ which characterize the NURBS geometries and this parametrization will affect, as consequence, also the forms and functionals of the problems formulation. At the current state of the art there is another work [MSH15] devoted to the RB and IGA coupling; in particular, in this work the construction of the RB method relies on as Isogeometric Boundary Element Method and the geometric parametrization regards only the control points coordinates. In this thesis, we will consider all the possible cases of parametrization of a NURBS geometry, that, as already said in Section 1.6, can regard the control points, the NURBS weights, or both control points and weights. For all these cases we show how RB and IGA can be jointly used in a feasible way.

2.2 Elliptic parametric PDEs

We introduce the formulation of affinely parametrized elliptic coercive problems in the case $a(\cdot,\cdot;\boldsymbol{\mu})$ is a symmetric bilinear form. We consider problems with $u(\boldsymbol{\mu}) \in V$, $V = V(\Omega)$ a suitable Hilbert space, and Ω a suitable regular, bounded, and open spatial domain in $\mathbb{R}^d$, with $d = 2$ or 3. We also introduce the inner product $(\cdot,\cdot)_V$, and the induced norm $||\cdot||_V$, associated with the Hilbert space V and assume that $[H_0^1(\Omega)]^\nu \subset V \subset [H^1(\Omega)]^\nu$, where $\nu = 1$ or $\nu = d$ for a scalar or vector field, respectively. Here $H^1(\Omega) = \left\{v \in L^2(\Omega), \nabla v \in (L^2(\Omega))^d\right\}$, $H_0^1(\Omega) = \left\{v \in H^1(\Omega) |\ v|_{\partial\Omega} = 0\right\}$, and L^2 is the space of square-integrable functions over Ω.

We shall assume that the bilinear form $a(\cdot,\cdot;\boldsymbol{\mu}) : V \times V \to \mathbb{R}$ is continuous and coercive over V for all $\boldsymbol{\mu} \in \mathcal{D}$, that is

$$\gamma(\boldsymbol{\mu}) := \sup_{w\in V}\sup_{v\in V} \frac{a(w,v;\boldsymbol{\mu})}{\|w\|_V\|v\|_V} < +\infty, \quad \forall \boldsymbol{\mu} \in \mathcal{D} \tag{2.4}$$

$$\exists \alpha_0 > 0 : \alpha(\boldsymbol{\mu}) := \inf_{w\in V} \frac{a(w,w;\boldsymbol{\mu})}{\|w\|_V^2} \geq \alpha_0, \quad \forall \boldsymbol{\mu} \in \mathcal{D} \tag{2.5}$$

and that $f(\cdot,\boldsymbol{\mu})$ is a continuous functional over V, $\forall \boldsymbol{\mu} \in \mathcal{D}$. Under these hypoteses, the Lax-Milgram lemma ensures that problem (2.1) admits a unique solution $\forall \boldsymbol{\mu} \in \mathcal{D}$.

In order to perform a computationally effective Offline-Online procedure, we will also assume that the bilinear form a and the linear functional f, are affine in the parameter $\boldsymbol{\mu}$. Under this assumption, for some finite M_a and M_f, $a(\cdot,\cdot;\boldsymbol{\mu})$ and $f(\cdot;\boldsymbol{\mu})$ can be expressed as

$$a(w, v; \boldsymbol{\mu}) = \sum_{m=1}^{M_a} \Theta_a^m(\boldsymbol{\mu}) a^m(w, v), \qquad \forall v, w \in V, \quad \forall \boldsymbol{\mu} \in \mathcal{D} \tag{2.6a}$$

$$f(v; \boldsymbol{\mu}) = \sum_{m=1}^{M_f} \Theta_f^m(\boldsymbol{\mu}) f^m(v), \qquad \forall v \in V, \quad \forall \boldsymbol{\mu} \in \mathcal{D} \tag{2.6b}$$

where $\Theta_a^m : \mathcal{D} \to \mathbb{R}$, for $1 \leq m \leq M_a$ and $\Theta_f^m : \mathcal{D} \to \mathbb{R}$, for $1 \leq m \leq M_f$ are the $\boldsymbol{\mu}$-dependent functions, while a^m, for $1 \leq m \leq M_a$ and f^m, for $1 \leq m \leq M_f$ are the $\boldsymbol{\mu}$-independent bilinear forms and linear functionals, respectively. In this way, in the Offline stage we can compute and store once and for all the $\boldsymbol{\mu}$-independent terms, so that, in the Online stage, for each new $\boldsymbol{\mu}$ we just have to compute the coefficients $\Theta_a^m(\boldsymbol{\mu})$ and $\Theta_f^m(\boldsymbol{\mu})$, multiply them for the already computed $\boldsymbol{\mu}$-independent terms as in (2.6), and finally solve the problem. It is clear that this allows large computational savings. Anyway, when the affinity property is not fulfilled, as it is the case of geometric parametrizations based on NURBS, it can be recovered by approximating the nonaffine problem through suitable methods, as we will see in Chapters 3 and 4 (see also [BMNP04]).

2.3 Parametrized formulation: a starting point for RB

We consider a linear elliptic problem which respects the hypotheses introduced in Section 2.2. For simplicity we consider a scalar field ($\nu = 1$) in two space dimension ($d = 2$). In the following we will be interested in the study of problems involving geometrical parameters. For this reason, we rewrite problem (2.1) and make the parametric dependence of the domain explicit. We will refer to this problem, posed in the *parameter-dependent* domain $\Omega_o = \Omega_o(\boldsymbol{\mu})$, as to the *original problem* (subscript o), that reads: given $\boldsymbol{\mu} \in \mathcal{D}$, find $u_o(\boldsymbol{\mu}) \in V_o(\boldsymbol{\mu})$ such that

$$a_o(u_o(\boldsymbol{\mu}), v; \boldsymbol{\mu}) = f_o(v; \boldsymbol{\mu}), \quad \forall v \in V_o(\boldsymbol{\mu}), \tag{2.7}$$

where $V_o(\boldsymbol{\mu})$ is a suitable Hilbert space defined on $\Omega_o(\boldsymbol{\mu})$.

As already said, to build a RB space, we have to compute the discrete solutions, for example approximated by IGA as we will do in Chapter 3, corresponding to the properly selected parameters. It means that, if also the computational domain depends on a parameter, these solutions would be computed on different domains and could not be combined and compared as RB requires. To avoid this drawback, the snapshots must be computed with respect to a common spatial configuration. For this reason, we will introduce a reference *parameter-independent* domain Ω and define the "transformed"problem on this reference domain. The original and the reference domains will be linked via a mapping

$$T(\cdot; \boldsymbol{\mu}) : \Omega \to \Omega_o(\boldsymbol{\mu}), \tag{2.8}$$

such that $\Omega_o(\boldsymbol{\mu}) = T(\Omega; \boldsymbol{\mu})$. The weak formulation of the reference problem in an abstract form reads: find $u(\boldsymbol{\mu}) \in V = V(\Omega)$ such that

$$a(u(\boldsymbol{\mu}), v; \boldsymbol{\mu}) = f(v; \boldsymbol{\mu}), \quad \forall v \in V. \tag{2.9}$$

Through the parameter dependent geometric transformation T we have reformulated the original problem (2.7) in a reference configuration Ω and so obtained a parametrized problem (2.9) where, as we will see in some practical examples in Chapter 3, the effect of geometry variations is traced back and limited only to the parametrized data in the formulation of the problem in Ω. In Chapter 3, we will also see how, in the IGA context, the map T will be related to the geometric map provided by NURBS in a single patch domain (1.18); we will also write the reference problem in the reference domain.

2.4 The reduced basis method

Now that we have defined the parametrized problem in the reference domain (2.9), we can proceed with the construction of its reduced basis approximation. We start by introducing the *truth* discrete approximation of problem (2.9). Let $V^{\mathcal{N}} \subset V$ be a sequence of Galerkin approximation subspaces of V such that $\dim(V^{\mathcal{N}}) = \mathcal{N} < +\infty$. The *truth* approximation of the reference problem (2.9) reads: for any $\boldsymbol{\mu} \in \mathcal{D}$, find $u^{\mathcal{N}}(\boldsymbol{\mu}) \in V^{\mathcal{N}}$ such that

$$a(u^{\mathcal{N}}(\boldsymbol{\mu}), v^{\mathcal{N}}; \boldsymbol{\mu}) = f(v^{\mathcal{N}}; \boldsymbol{\mu}), \quad \forall v^{\mathcal{N}} \in V^{\mathcal{N}}. \tag{2.10}$$

Now, we can list the principal steps of the actual reduced spaces algorithm construction [RHP08, Man12, Qua14]:

1. we introduce a positive integer N_{max} and a *master set* of properly selected parameter values $\boldsymbol{\mu}^n \in \mathcal{D}$, $1 \leq n \leq N_{max}$; then, through a sampling strategy that we will detail in Section 2.4.2, we define, for a given $N \in 1, ..., N_{max}$, the Lagrange parameter samples

$$S_N = \left\{\boldsymbol{\mu}^1, ..., \boldsymbol{\mu}^N\right\}; \tag{2.11}$$

2. we solve problem (2.10) for $\boldsymbol{\mu}^n$, $1 \leq n \leq N_{max}$. In this way, we obtain the solutions $u^{\mathcal{N}}(\boldsymbol{\mu}^n)$ that are often referred to as *retained snapshots* of the parametric manifold $\mathcal{M}^{\mathcal{N}}$. With these snapshots, we build the Lagrange RB spaces associated with the Lagrange parameter samples

$$V_N = \text{span}\left\{u^{\mathcal{N}}(\boldsymbol{\mu}^{\boldsymbol{n}}), 1 \leq n \leq N\right\}. \tag{2.12}$$

   For $N = 1, ..., N_{max}$, V_N is a N-dimensional subspace of $V^{\mathcal{N}}$. We further observe that the Lagrange spaces just built are nested (or hierarchical), that is $V_1^{\mathcal{N}} \subset V_2^{\mathcal{N}} \subset ... \subset V_{N_{max}}^{\mathcal{N}} \subset V^{\mathcal{N}}$. This is a fundamental condition for the (memory) efficiency of the resulting RB approximation. It is clear that if the manifold $\mathcal{M}^{\mathcal{N}}$ is low-dimensional and smooth, we could approximate any member of the manifold, i.e. any solution $u^{\mathcal{N}}(\boldsymbol{\mu})$ for some $\boldsymbol{\mu} \in \mathcal{D}$, in terms of relatively few retained snapshots.

3. through an interpolation procedure (a Galerkin projection) of these precomputed snapshots, we build a low-dimensional approximation $\mathcal{M}^N$ of $\mathcal{M}^{\mathcal{N}}$ (see Section 2.4.1);

4. Offline-Online procedure. We decouple the computational effort in two stages: an expensive (parameter independent) offline stage and an inexpensive (parameter dependent) online stage (see Section 2.4.4).

2.4.1 Galerkin projection

Let us suppose to have already selected the N parameter samples of the set S_N defined in (2.11). In Sections 2.4.2 and 2.4.3 we will detail two strategies to perform such a selection. For each of these parameters $\boldsymbol{\mu}^n$, we can compute the retained snapshot $u^{\mathcal{N}}(\boldsymbol{\mu}^n)$, that is the solution of the particular problem dependent on $\boldsymbol{\mu}^n$, and define the Lagrange RB space V_N associated to the Lagrange parameter samples as already done in (2.12), that is a N-dimensional subspace of $V^{\mathcal{N}}$. We can now finally compute the reduced approximation $u_N(\boldsymbol{\mu})$ through a Galerkin projection of the PDE onto the space V_N: given $\boldsymbol{\mu} \in \mathcal{D}$, find $u_N(\boldsymbol{\mu}) \in V_N \subset V^{\mathcal{N}}$ such that

$$a(u_N(\boldsymbol{\mu}), v_N; \boldsymbol{\mu}) = f(v_N; \boldsymbol{\mu}), \quad \forall v_N \in V_N. \tag{2.13}$$

As said in Section 2.2, the bilinear form $a(\cdot, \cdot; \boldsymbol{\mu})$ is coercive and symmetric for any given $\boldsymbol{\mu}$. Under these assumptions, it defines a (energy) scalar product given by $((w, v))_{\boldsymbol{\mu}} := a(w, v; \boldsymbol{\mu})$ $\forall w, v \in V$ and the induced energy norm is given by $|||w|||_{\boldsymbol{\mu}} = ((w, w))_{\boldsymbol{\mu}}^{1/2}$. As consequence, we have the classical optimality result in the energy norm [RHP08]:

$$|||u^{\mathcal{N}} - u_N|||_{\boldsymbol{\mu}} \leq \inf_{w \in V_N} |||u^{\mathcal{N}}(\boldsymbol{\mu}) - w|||_{\boldsymbol{\mu}}, \tag{2.14}$$

that means that the Galerkin procedure automatically selects the best combination of snapshots.

We now apply Gram-Schmidt procedure with respect to the $(\cdot, \cdot)_V$ inner product to the snapshots $u^{\mathcal{N}}(\boldsymbol{\mu}^n)$, $1 \leq n \leq N$, to obtain mutually $(\cdot, \cdot)_V$-orthonormal basis functions ζ_n, $1 \leq n \leq N$. Then, the RB solution of problem (2.13) can be expressed as:

$$u_N(\boldsymbol{\mu}) = \sum_{j=1}^{N} u_{N_j}(\boldsymbol{\mu}) \zeta_j. \tag{2.15}$$

By taking $v = \zeta_i$, $1 \leq i \leq N_{max}$ into (2.13), and using (2.15), we obtain the set of linear algebraic equations

$$\sum_{j=1}^{N} a(\zeta_j, \zeta_i; \boldsymbol{\mu}) u_{N_j}(\boldsymbol{\mu}) = f(\zeta_i; \boldsymbol{\mu}), \quad 1 \leq i \leq N \tag{2.16}$$

for the reduced basis coefficients u_{N_j}, $1 \leq j \leq N$. The system (2.16) can be expressed in matrix form as

$$\boldsymbol{A}_N(\boldsymbol{\mu}) \boldsymbol{u}_N(\boldsymbol{\mu}) = \boldsymbol{f}_N(\boldsymbol{\mu}), \tag{2.17}$$

where $(\boldsymbol{u}_N(\boldsymbol{\mu}))_j = u_{N_j}(\boldsymbol{\mu})$, and the matrix $\boldsymbol{A}_N$ and the vector $\boldsymbol{f}_N$ are given by

$$(\boldsymbol{A}_N(\boldsymbol{\mu}))_{ij} = a(\zeta_j, \zeta_i; \boldsymbol{\mu}), \quad (\boldsymbol{f}_N(\boldsymbol{\mu}))_i = f(\zeta_i; \boldsymbol{\mu}),$$

respectively.

In Section 2.4.4 we will present the Offline-Online procedure that allows to solve in an efficient way, with computational costs independent of $\mathcal{N}$, the linear system (2.17). In Sections 2.4.2 and 2.4.3 we describe, as already anticipated, two methods for the construction of the

reduced basis, namely the greedy algorithm and the Proper Orthogonal Decomposition (POD) method.

2.4.2 Sampling strategies: greedy algorithm

In this section we discuss the greedy algorithm as a possible technique to sample the parameter space $\mathcal{D}$, construct the Lagrange parameter samples S_N, and finally the Lagrange RB spaces V_N. We start by introducing a finite set Ξ of parameters in $\mathcal{D}$ that will be used to understand and assess the quality of the RB approximation and a posteriori error estimators. We also define Ξ_{train} as the particular samples set that will be used to generate the RB approximation. The cardinality of Ξ_{train} will be denoted as $|\Xi_{train}| = n_{train}$. In the greedy algorithm that we report below, the term $\Delta_N(\boldsymbol{\mu})$ will refer to an *a posteriori error bound* or *error estimator* for $\|u^{\mathcal{N}}(\boldsymbol{\mu}) - u_N(\boldsymbol{\mu})\|_V$, where $u_N(\boldsymbol{\mu})$ is the reduced approximation obtained through a (Galerkin-like) projection of the PDE onto V_N as we have seen in Section 2.4.1. The expression of the estimator $\Delta_N(\boldsymbol{\mu})$ and its role will be better explained in Section 2.5. Finally, we introduce ϵ^*_{tol}, a chosen tolerance for the stopping criterium of the greedy algorithm. The greedy sampling strategy then is [RHP08]:

Set $S_1 = \{\boldsymbol{\mu}^1\}$; compute $u^{\mathcal{N}}(\boldsymbol{\mu}^1)$ solving (2.10); compute $\Delta_1(\boldsymbol{\mu}^1)$;
$V_1 = \text{span}\{u^{\mathcal{N}}(\boldsymbol{\mu}^1)\}$;
for $N = 2 : N_{max}$
  $\boldsymbol{\mu}^N = \arg\max_{\boldsymbol{\mu}\in\Xi_{train}} \Delta_{N-1}(\boldsymbol{\mu})$;
  $\epsilon_{N-1} = \Delta_{N-1}(\boldsymbol{\mu}^N)$;
  if $\epsilon_{N-1} < \epsilon^*_{tol}$
    set $N_{max} = N - 1$;
  end
  compute $u^{\mathcal{N}}(\boldsymbol{\mu}^N)$ solving (2.10); compute $\Delta_N(\boldsymbol{\mu}^N)$;
  $S_N = S_{N-1} \cup \{\boldsymbol{\mu}^N\}$;
  $V_N = V_{N-1} \cup \text{span}\{u^{\mathcal{N}}(\boldsymbol{\mu}^N)\}$;
end

The idea of the greedy procedure is that, starting with the train sample Ξ_{train} we adaptively select N parameters $\boldsymbol{\mu}^1, ..., \boldsymbol{\mu}^N$. In particular, at each iteration N, the greedy algorithm appends to the previously retained snapshots $\{u^{\mathcal{N}}(\boldsymbol{\mu}^n)\}_{n=1}^{N-1}$ that particular snapshots $u^{\mathcal{N}}(\boldsymbol{\mu}^N)$, over all the candidates $u^{\mathcal{N}}(\boldsymbol{\mu})$, with $\boldsymbol{\mu} \in \Xi_{train}$, that is worst approximated by V_{N-1}. To this end, we should compute for each $\boldsymbol{\mu} \in \Xi_{train}$ the norm of the error $\|u^{\mathcal{N}}(\boldsymbol{\mu}) - u_{N-1}(\boldsymbol{\mu})\|_V$, being $u_{N-1}(\boldsymbol{\mu}) \in V_{N-1}$. Since it would be computationally disadvantageous, we actually use the a posteriori error estimator $\Delta_N(\boldsymbol{\mu})$ for the reduced basis error $\|u^{\mathcal{N}}(\boldsymbol{\mu}) - u_N(\boldsymbol{\mu})\|_V$ (see section 2.5). Following this procedure we end up with the hierarchical sequence of reduced basis spaces

$$V_N = \text{span}\{u^{\mathcal{N}}(\boldsymbol{\mu}^n), 1 \le n \le N\}, \quad 1 \le N \le N_{max}. \tag{2.18}$$

In order to recover an orthonormal well-conditioned set of basis functions and to guarantee a good algebraic stability, we apply the Gram-Schmidt process in the $(\cdot;\cdot)_V$ inner product to the snapshots $u^{\mathcal{N}}(\boldsymbol{\mu})$, as anticipated in Section 2.4.1. In this way, we obtain mutually orthonormal basis functions ζ_n, $1 \le n \le N$. Moreover, since each basis function ζ_j is an element of the truth dimensional space $V^{\mathcal{N}}$, they can be expressed as linear combination of any basis function $\{\Phi_k\}_{k=1}^{\mathcal{N}}$ of $V^{\mathcal{N}}$, that is

$$\zeta_i = \sum_{k=1}^{\mathcal{N}} \zeta_{i_k} \Phi_k, \quad 1 \leq i \leq N_{max}, \tag{2.19}$$

where $\zeta_{ik} = (\boldsymbol{\zeta}_i)_k$, with $\boldsymbol{\zeta}_i \in \mathbb{R}^{\mathcal{N}}$. Finally, we define a basis matrix $\boldsymbol{\mathcal{Z}} \in \mathbb{R}^{\mathcal{N} \times N}$, $1 \leq N \leq N_{max}$, having the basis $\boldsymbol{\zeta_n}$ as columns:

$$(\boldsymbol{\mathcal{Z}})_{jl} = \zeta_{j_l}, \text{ i.e. } \boldsymbol{\mathcal{Z}} = [\boldsymbol{\zeta}_1, ..., \boldsymbol{\zeta}_N] \in \mathbb{R}^{\mathcal{N} \times N}, 1 \leq N \leq N_{max}. \tag{2.20}$$

Finally we can express the basis space V_N as $V_N = \text{span}(\boldsymbol{\mathcal{Z}})$. For a general analysis of greedy algorithm and related convergence rates see [BCD$^+$11].

2.4.3 Sampling strategies: Proper Orthogonal Decomposition

The POD method allows to define a subspace $V_N \subset V^{\mathcal{N}}$ approximating the data of $V^{\mathcal{N}}$ in an optimal least-squares sense. In particular, the POD technique reduces the dimensionality of a system by transforming the original unknowns into new variables (called POD modes or principal components) such that the first few modes retain most of the energy present in the system [Man12, BHL93]. The POD method relies on the use of the singular value decomposition (SVD) algorithm [Vol11]. Let us consider a set of n_{train} snapshots $\{\boldsymbol{u}_1, ..., \boldsymbol{u}_{n_{train}}\} = \{\boldsymbol{u}(\boldsymbol{\mu}^1), ..., \boldsymbol{u}(\boldsymbol{\mu}^{n_{train}})\} \in V^{\mathcal{N}}$. By performing the SVD on the matrix $\boldsymbol{\mathcal{U}} \in \mathbb{R}^{\mathcal{N} \times n_{train}}$ having the n_{train} snapshots as columns, $\boldsymbol{\mathcal{U}} = [\boldsymbol{u}^1, ..., \boldsymbol{u}^{n_{train}}]$, we obtain the following singular value decomposition of $\boldsymbol{\mathcal{U}}$:

$$\boldsymbol{\mathcal{U}} = \boldsymbol{\mathcal{V}} \boldsymbol{\Sigma} \boldsymbol{\mathcal{W}}^T \tag{2.21}$$

where

$$\boldsymbol{\mathcal{V}} = [\boldsymbol{\zeta}_1, ..., \boldsymbol{\zeta}_{n_{train}}] \in \mathbb{R}^{\mathcal{N} \times n_{train}} \tag{2.22}$$

and

$$\boldsymbol{\mathcal{W}} = [\boldsymbol{\Psi}_1, ..., \boldsymbol{\Psi}_{n_{train}}] \in \mathbb{R}^{n_{train} \times n_{train}} \tag{2.23}$$

are orthogonal matrices whose columns are the left and right singular vectors of $\boldsymbol{\mathcal{U}}$ respectively, and

$$\boldsymbol{\Sigma} = \text{diag}(\sigma_1, ..., \sigma_{n_{train}}) \in \mathbb{R}^{n_{train} \times n_{train}} \tag{2.24}$$

is a diagonal matrix such that $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_{n_{train}} \geq 0$ are the computed singular values of $\boldsymbol{\mathcal{U}}$. The POD basis of dimension N is defined by the first N left singular vectors $[\boldsymbol{\zeta}_1, ..., \boldsymbol{\zeta}_N]$ of $\boldsymbol{\mathcal{U}}$, that correspond to the N largest singular values:

$$\boldsymbol{\mathcal{Z}} = [\boldsymbol{\zeta}_1, ..., \boldsymbol{\zeta}_N], \quad 1 \leq N \leq N_{max} \tag{2.25}$$

The POD basis is *optimal* in the sense that, for a basis of size N, it minimizes the least squares error of snapshot reconstruction,

$$\min_{\boldsymbol{\mathcal{Z}} \in \mathbb{R}^{\mathcal{N} \times N}} \sum_{i=1}^{n_{train}} \|\boldsymbol{u}_i - \boldsymbol{\mathcal{Z}} \boldsymbol{\mathcal{Z}}^T \boldsymbol{u}_i\|_2^2 = \sum_{i=N+1}^{n_{train}} \sigma_i^2. \tag{2.26}$$

As results from (2.26), the error in snapshots representation is given by the sum of the squares of the singular values corresponding to those left singular vectors not included in the POD basis. In particular, N_{max} is tipically chosen as the smallest N such that:

$$I(N) = \frac{\sum_{i=1}^{N} \sigma_i^2}{\sum_{i=1}^{n_{train}} \sigma_i^2} > 1 - \text{tol}_{POD}. \tag{2.27}$$

The numerator of (2.27) is often referred to as the *energy* captured by the POD modes. So with (2.27) we are requiring that the energy retained by the last $n_{train} - N$ modes is equal to tol_{POD}, being tol_{POD} as small as desired. Finally, the basis space V_N can be expressed as $V_N = \text{span}(\boldsymbol{\mathcal{Z}})$.

2.4.4 Offline-Online procedure

The system (2.16), that we have obtained thanks to Galerkin projection, can be recast in a set of N linear algebraic equations in N unknowns, and so a system of relatively small size if compared to (2.10). Unfortunately, some $\mathcal{N}$-dimensional elements (specifically, the basis functions ζ_j, $1 \leq j \leq N$) are still involved in the stiffness matrix and the right-hand side assembling. However, the affine parameter dependence help us in avoiding this drawback by allowing a very efficient Offline-Online procedure. As a matter of fact, by using (2.6) and (2.16), system (2.13) can be rewritten as

$$\sum_{j=1}^{N} \left(\sum_{m=1}^{M_a} \Theta_a^m(\boldsymbol{\mu}) a^m(\zeta_j, \zeta_i) \right) u_{N_j}(\boldsymbol{\mu}) = \sum_{m=1}^{M_f} \Theta_f^m(\boldsymbol{\mu}) f^m(\zeta_i), \tag{2.28}$$

for $1 \leq i \leq N$. The equivalent matrix form is

$$\left(\sum_{m=1}^{M_a} \Theta_a^m(\boldsymbol{\mu}) \boldsymbol{A}_N^m \right) \boldsymbol{u}_N(\boldsymbol{\mu}) = \sum_{m=1}^{M_f} \Theta_f^m(\boldsymbol{\mu}) \boldsymbol{f}_N^m, \tag{2.29}$$

where

$$(\boldsymbol{u}_N(\boldsymbol{\mu}))_j = u_{N_j}(\boldsymbol{\mu}) \qquad (\boldsymbol{A}_N^m)_{ij} = a^m(\zeta_j, \zeta_i) \qquad (\boldsymbol{f}_N^m)_i = f^m(\zeta_i)$$

for $1 \leq i, j \leq N_{max}$. We notice that in (2.19) we have assembled the N_{max} basis functions, but we will see that in the Online stage we will use only $N \leq N_{max}$ of them. As consequence, the RB stiffness matrix and RB right-hand side vector can be expressed as follows:

$$(\boldsymbol{A}_N^m)_{ij} = a^m(\zeta_j, \zeta_i) = \sum_{k=1}^{\mathcal{N}} \sum_{l=1}^{\mathcal{N}} \zeta_{jl} a^m(\Phi_l, \Phi_k) \zeta_{ik} \quad 1 \leq i, j \leq N, \;\; 1 \leq m \leq M_a \tag{2.30}$$

$$(\boldsymbol{f}_N^m)_i = f^m(\zeta_i) = \sum_{k=1}^{\mathcal{N}} \zeta_{ik} f^m(\Phi_k), \quad 1 \leq i \leq N, \;\; 1 \leq m \leq M_f; \tag{2.31}$$

then, we can rewrite (2.30) and (2.31) as

$$\boldsymbol{A}_N^m = \boldsymbol{\mathcal{Z}}^T \boldsymbol{\mathcal{A}}_{\mathcal{N}}^m \boldsymbol{\mathcal{Z}}, \qquad \boldsymbol{f}_N^m = \boldsymbol{\mathcal{Z}}^T \boldsymbol{\mathcal{F}}_{\mathcal{N}}^m \tag{2.32}$$

being $\boldsymbol{\mathcal{Z}} \in \mathbb{R}^{\mathcal{N}\times N}$ the basis matrix defined in (2.20) or in (2.25), if a greedy or a POD sampling strategy has been adopted respectively, and

$$(\boldsymbol{\mathcal{A}}_{\mathcal{N}}^m)_{kl} = a^m(\Phi_l, \Phi_k), \qquad (\boldsymbol{\mathcal{F}}_{\mathcal{N}}^m)_k = f^m(\Phi_k).$$

By means of this procedure, we can decouple the computational effort required to assemble system (2.29) in two stages: an *Offline* stage and an *Online* stage. The Offline stage consists in the computation and storage of the matrices $\{\boldsymbol{\mathcal{A}}_{\mathcal{N}}^m\}_{m=1}^{M_a}$ and vectors $\{\boldsymbol{\mathcal{F}}_{\mathcal{N}}^m\}_{m=1}^{M_f}$, the snapshots $\boldsymbol{u}^{\mathcal{N}}(\boldsymbol{\mu}^n)$, and the orthonormal basis $\{\boldsymbol{\zeta}_j\}_{j=1}^{N_{max}}$. The Offline operation count depends on N_{max}, M_a, M_f, $\mathcal{N}$. This is an expensive ($\boldsymbol{\mu}$-independent) stage that we likely have to perform only once. In the Online ($\boldsymbol{\mu}$-dependent) stage, for any given $\boldsymbol{\mu}$, we have to evaluate the parameter dependent functions $\Theta_a^m(\boldsymbol{\mu})$ and $\Theta_f^m(\boldsymbol{\mu})$ and use the precomputed $\boldsymbol{A}_N^m$, $\boldsymbol{f}_N^m$ to assemble the $N \times N$ system (2.29) and solve it to finally obtain $\boldsymbol{u}_N(\boldsymbol{\mu})$. In this way, we obtain the RB approximation $\boldsymbol{u}_N(\boldsymbol{\mu})$, by assembling a full but very small matrix, whose associated system has dimension independent of $\mathcal{N}$. In particular, the computational cost of the Online stage is $O(M_a N^2)$ to assemble the matrix and $O(N^3)$ to solve the system (2.29) with a direct method. The Online storage count is, thanks to the hierarchical condition (see Section 2.4), only $O(M_a N_{max}^2)$. As a matter of fact, for any given N we can extract the necessary RB $N \times N$ matrices (and the N-vectors) as principal submatrices of the larger $N_{max} \times N_{max}$ matrix (and N_{max}-vector). The online cost to evaluate $u_N(\boldsymbol{\mu})$ is thus independent of $\mathcal{N}$.

2.5 *A posteriori* error estimation

The development of a rigorous error bound for the quantity of interest, the solution of the PDE problem or particular output functionals depending on the solution of the PDE, is crucial for both the *efficiency* and the *reliability* of the RB approximation method.

- As regards to the *efficiency*, the error bound has a crucial role in the sampling strategy. In the greedy algorithm, the evaluation of the error for all $\boldsymbol{\mu} \in \Xi_{train}$ at each iteration permits to select properly the basis functions in such a way to achieve the best accuracy with the smallest number of basis functions.

- As regards the *reliability*, at the Online phase for each new value of parameter $\boldsymbol{\mu} \in \mathcal{D}$, the a posteriori estimator $\Delta_N(\boldsymbol{\mu})$ permits to bound the error of the RB approximation with respect to the underlying truth approximation, as we will see in the following of this section.

However, in order to ensure that the error bound possesses these qualities, we have to place some requirements on it. In particular, the error bounds must be:

- *rigorous*, i.e. must ensure the reliability for all $N \leq N_{max}$ and for all $\boldsymbol{\mu} \in \mathcal{D}$;

- *sharp*, since an overly conservative error bound can yield inefficient approximations (N too large with respect to the originally required level of error);

- *efficient*, i.e. the Online operation count and storage to compute the RB error bounds must be independent of $\mathcal{N}$.

An important equation in *a posteriori* theory is the error residual relationship. In particular, the error $e(\boldsymbol{\mu}) := u^{\mathcal{N}}(\boldsymbol{\mu}) - u_N(\boldsymbol{\mu}) \in V^{\mathcal{N}}$ satisfies

$$a(e(\boldsymbol{\mu}), v; \boldsymbol{\mu}) = r(v; \boldsymbol{\mu}), \quad \forall v \in V^{\mathcal{N}}, \tag{2.33}$$

where $r(v, \boldsymbol{\mu}) \in (V^{\mathcal{N}})'$, with $(V^{\mathcal{N}})'$ the dual space of $V^{\mathcal{N}}$, is the residual, defined as

$$r(v; \boldsymbol{\mu}) := f(v; \boldsymbol{\mu}) - a(u_N(\boldsymbol{\mu}), v; \boldsymbol{\mu}), \quad \forall v \in V^{\mathcal{N}}. \tag{2.34}$$

We introduce the Riesz representation of $r(v; \boldsymbol{\mu})$

$$\big(\hat{e}(\boldsymbol{\mu}), v\big)_V = r(v; \boldsymbol{\mu}), \quad \forall v \in V^{\mathcal{N}}. \tag{2.35}$$

In this way, we are allowed to rewrite (2.33) as follows

$$a(e(\boldsymbol{\mu}), v; \boldsymbol{\mu}) = \big(\hat{e}(\boldsymbol{\mu}), v\big)_V, \quad \forall v \in V^{\mathcal{N}} \tag{2.36}$$

and the dual norm of the residual can be evaluated through the Riesz representation

$$||r(\cdot; \boldsymbol{\mu})||_{(V^{\mathcal{N}})'} := \sup_{v \in V^{\mathcal{N}}} \frac{r(v; \boldsymbol{\mu})}{||v||_V} = ||\hat{e}(\boldsymbol{\mu})||_V; \tag{2.37}$$

from the coercivity property of the bilinear form $a(\cdot; \cdot; \boldsymbol{\mu})$, valid $\forall \boldsymbol{\mu} \in \mathcal{D}$, immediately follows that

$$||e(\boldsymbol{\mu})||_V \leq \frac{||r(\cdot; \boldsymbol{\mu}))||_{V'}}{\alpha^{\mathcal{N}}(\boldsymbol{\mu})} = \frac{||\hat{e}(\boldsymbol{\mu})||_V}{\alpha^{\mathcal{N}}(\boldsymbol{\mu})}. \tag{2.38}$$

Moreover, we need a lower bound function $\alpha^{\mathcal{N}}_{LB}$ for the bilinear form coercivity constant $\alpha^{\mathcal{N}}(\boldsymbol{\mu})$ defined as

$$\alpha^{\mathcal{N}}(\boldsymbol{\mu}) = \inf_{w \in V^{\mathcal{N}}} \frac{a(w, w; \boldsymbol{\mu})}{||w||^2_V}, \tag{2.39}$$

such that

$$0 < \alpha^{\mathcal{N}}_{LB}(\boldsymbol{\mu}) \leq \alpha^{\mathcal{N}}(\boldsymbol{\mu}), \quad \forall \boldsymbol{\mu} \in \mathcal{D}. \tag{2.40}$$

With this aim, the so-called Successive Constraint Method (SCM) has been developed [Man12, HRSP07, RHP08]. SCM is an iterative procedure based on the successive solution of suitable linear optimization problems. Despite of its generality, SCM often suffers of slow convergence, thus requiring to use more efficient procedures. In this work, for instance, we will use the Radial Basis Functions (RBF) procedure. This technique consists in the construction of suitable radial basis interpolants, through an adaptive choice of the interpolation points in the parameter space. In this way, it is possible to obtain reliable approximation, whose Offline construction and Online evaluation prove to be extremely faster [Man12].

By assuming for the moment to have at our disposal the lower bound $\alpha^{\mathcal{N}}_{LB}(\boldsymbol{\mu})$ we finally define the error estimator for the solution

$$\Delta_N(\boldsymbol{\mu}) := ||\hat{e}(\boldsymbol{\mu})||_V/(\alpha_{LB}^N(\boldsymbol{\mu}))^{1/2}, \tag{2.41}$$

From the inequality (2.38) it follows that

$$||u^{\mathcal{N}}(\boldsymbol{\mu}) - u_N(\boldsymbol{\mu})|| \leq \Delta_N(\boldsymbol{\mu}). \tag{2.42}$$

We also introduce the effectivity indeces associated with this error estimator

$$\eta_N(\boldsymbol{\mu}) := \Delta_N(\boldsymbol{\mu})/\|u^{\mathcal{N}}(\boldsymbol{\mu}) - u_N(\boldsymbol{\mu})\|_V. \tag{2.43}$$

The effectivities measure the quality of the proposed estimator. For the rigorousity of the estimator, we will insist upon effectivity indeces which are greater or equal to 1, while for the sharpness we will insist on values as close to the unity as possible. Nevertheless, the error bounds Δ_N is not useful without an accompanying Offline-Online computational approach. The computationally crucial component of all the error bounds introduced is the dual norm $||\hat{e}(\boldsymbol{\mu})||_V$ of the residual, which can be indeed computed through an Offline-Online procedure (see [QRM11, Neg11]). To develop this Offline-Online procedure, we introduce the residual expansion

$$r(v;\boldsymbol{\mu}) = f(v;\boldsymbol{\mu}) - a(\sum_{n=1}^{N} u_{N_n}(\boldsymbol{\mu})\zeta_n^{\mathcal{N}}, v;\boldsymbol{\mu}) \tag{2.44}$$

$$= \sum_{m=1}^{M_f} \Theta(\boldsymbol{\mu}) f^m(v) - \sum_{n=1}^{N} u_{N_n}(\boldsymbol{\mu}) \sum_{m=1}^{M_a} \Theta_a^m(\boldsymbol{\mu}) a^m(\zeta_n^{\mathcal{N}}, v). \tag{2.45}$$

From (2.44) and (2.35), it follows that

$$(\hat{e}(\boldsymbol{\mu}), v)_V = \sum_{m=1}^{M_f} \Theta(\boldsymbol{\mu}) f^m(v) - \sum_{n=1}^{N} u_{N_n}(\boldsymbol{\mu}) \sum_{m=1}^{M_a} \Theta_a^m(\boldsymbol{\mu}) a^m(\zeta_n^{\mathcal{N}}, v), \tag{2.46}$$

and consequently

$$\hat{e}(\boldsymbol{\mu}) = \sum_{m=1}^{M_f} \Theta(\boldsymbol{\mu}) \mathcal{F}^m - \sum_{n=1}^{N} u_{N_n}(\boldsymbol{\mu}) \sum_{m=1}^{M_a} \Theta_a^m(\boldsymbol{\mu}) \mathcal{L}_n^m, \tag{2.47}$$

where $\mathcal{F}^m$ is the Riesz representation of $f^m(\cdot)$ and $\mathcal{L}_n^m$ is the Riesz representation of $a^m(\zeta_n^{\mathcal{N}}, \cdot)$, that is $\forall v \in V$

$$(\mathcal{F}^m, v)_V = f^m(v), \qquad 1 \leq m \leq M_f \tag{2.48}$$

$$(\mathcal{L}_n^m, v)_V = -a^m(\zeta_n^{\mathcal{N}}, v), \quad 1 \leq m \leq M_a, \ 1 \leq n \leq N. \tag{2.49}$$

Finally we obtain

$$\|\hat{e}(\boldsymbol{\mu})\|_V^2 = \sum_{m=1}^{M_f}\sum_{m'=1}^{M_f} \Theta_f^m(\boldsymbol{\mu})\Theta_f^{m'}(\boldsymbol{\mu})(\mathcal{F}^m, \mathcal{F}^{m'})_V + \sum_{m=1}^{M_a}\sum_{n=1}^{N} \Theta_a^m(\boldsymbol{\mu}) u_{N_n}(\boldsymbol{\mu}) \Big(\tag{2.50}$$

$$2\sum_{m'=1}^{M_f} \Theta_f^{m'}(\boldsymbol{\mu})(\mathcal{F}^{m'}, \mathcal{L}_n^m)_V + \sum_{m'=1}^{M_a}\sum_{n'=1}^{N} \Theta_a^{m'}(\boldsymbol{\mu}) u_{N_n'}(\boldsymbol{\mu})(\mathcal{L}_n^m, \mathcal{L}_n^{m'})_V \Big). \tag{2.51}$$

Such a decomposition is mandatory in order to efficiently evaluate the error indicator (2.41) and then to select the snapshots in the greedy algorithm. Anyway, we observe that since this expression of the dual norm of the residual depends on the number of the affine terms M_a and M_f, in some cases it can dramatically affects the Offline time required by each step of the greedy algorithm.

2.6 RB on parametrized surfaces

In Chapters 3 and 4 we will be interested in solving practical examples of problems defined on parametrized surfaces. As seen at the end of Chapter 1, when a problem has to be solved for a large set of parameters $\boldsymbol{\mu}$, the total time requested for each evaluation could be computationally prohibitive. That is why, in both the chapters, we will apply the Reduced Basis technique.

In Chapter 3 the parametrization of the surface will concern only the control points coordinates, while the NURBS weights will not be parameter dependent. As test case, we will consider the Poisson problem on a 2D surface, but the approach that we will describe can be used for any kind of problem defined on a NURBS geometry, as a surface in 3D, whose control points coordinates are parametrized, but not its weights. For the practical solution of the problem considered, we will see that the bilinear forms and the linear functionals of its variational form, will depend in a nonaffine way on the spatial coordinate and on the geometric parameter. We recall that, to solve a problem by adopting an RB method, the affinity assumption must be respected. For this reason, we will appeal to the Empirical Interpolation Method (EIM) [BMNP04] as a suitable technique to restore the affinity of the forms and functionals. We stress the fact that, although the test case regards a 2D surface, the same approach must be followed when considering a problem defined on a surface in 3D parametrized with respect to its control points coordinates. Moreover, the approach that we are going to consider is valid for a generic NURBS surface, that means that the weights of the geometry can assume any *fixed* value; it follows that the case of parametrized control points of B-Splines lays in this category.

In Chapter 4 the parametrization of the surface will concern also the NURBS weights. As test case, we will consider the Laplace-Beltrami problem on a surface in 3D. Also in this case, the bilinear forms and the linear functionals of its variational form, will depend in a nonaffine way on the spatial coordinate and on the geometric parameter. Our first attempt to restore the affinity assumption will be to follow the same procedure adopted in Chapter 3, that is to apply EIM. Anyway, since the weights are parameter dependent, the NURBS functions are parameter dependent and this fact, as we will see, does not allow to use EIM. For this reason, in order to restore the affinity of the forms and functionals, we will appeal to another methodology, the Matrix Discrete Eimpirical Interpolation Method (MDEIM). So we will see that when the NURBS weights are parametrized, we will be obliged to use MDEIM no matter if the control points coordinates are parametrized or not.

Chapter 3

Reduced Basis for IGA: Parametrized NURBS control points

In this chapter we consider a class of problems described by PDEs defined on parametrized geometries. In this context, we are interested in solving the PDE for different configurations of the computational domain represented by B-splines or NURBS. As anticipated, here we treat the case where only the NURBS control points coordinates are parametrized, but the NURBS weights are kept fixed.

In order to solve the problem efficiently for each new parameter $\boldsymbol{\mu}$, namely the control points coordinates, we apply the RB method. As said in Chapter 2, by adopting this strategy, the approximated solution corresponding to any new parameter value is taken as a linear combination of a set of solutions of the same PDE, precomputed on a reference configuration. In this respect, we investigate how to map the problem defined in the parameter dependent domain (the original problem) into the one defined in a reference domain (the reference problem), and how to suitably choose this latter. For the spatial discretization of the problems of interest, we consider IGA, for which we choose the reduced space as a proper subset of the space of NURBS $\mathcal{B}_h$ defined in (1.19), i.e. the same NURBS functions used to build the computational domain.

The parametrization of the NURBS control points coordinates induces a non affine geometrical transformation, that in turn induces a nonaffine dependence of the problem and its formulation with respect to the parameters. We recall that, in order to fully exploit the computational savings allowed by RB, the problem at hand should satisfy the affinity assumption. If this assumption is not fulfilled, as in this case, it must be restored by means of suitable methods. In order to restore the affinity of the bilinear form and the linear functional of this problem, we appeal to the Empirical Interpolation Method (EIM), [BMNP04, GMNP07]. The idea of EIM is to approximate a non linear function by projecting it onto a low dimensional subspace. Thus, EIM represents an efficient technique to be applied to complex nonaffine shape parametrization mappings.

We underline that the procedure of mapping the problem in a reference domain and apply EIM is valid for a general NURBS surface in 2D or 3D, whose weights assume any *fixed* value; in this context, it naturally follows that when the NURBS weights are all equal and the NURBS surface degenerates in a B-spline surface, the problem can still be solved with

this procedure. As numerical test, we consider the Poisson problem on a 2D surface in order to highlight the reduction strategy.

In Chapter 4 we will consider the case in which the NURBS weights are parameter dependent. In this context, we will highlight the limitations of EIM, that could not be used anymore. For this reason, in order to restore the affinity assumption of the parametrized problem, we will appeal to another technique, the Matrix Discrete Empirical Interpolation Method (MDEIM) [NMA15].

3.1 Problem formulation on a parametrized surface

In this chapter we aim to solve a problem defined on a parametrized NURBS surface by combining IGA and RB. We start by considering a test problem, the Poisson problem on a 2D computational domain. We recall that, as said in Section 1.4.4, the Poisson problem (1.55) can be obtained as particular case of the Laplace-Beltrami problem (1.32) when both the parametric domain $\hat{\Omega}$ and the physical domain Ω are in $\mathbb{R}^2$.

In order to exploit the advantages of the RB methodology, we introduce a parametrized version of the Poisson problem, by making the computational domain parameter dependent. For instance, we introduce a parameter $\boldsymbol{\mu} = \mu \in \mathcal{D} \subset \mathbb{R}$ for the displacement of some control points coordinates (as done in Example 1.6). The NURBS weights, instead, are not parameter dependent but can assume any fixed value. From now on we will refer to the parametrized computational domain as to $\Omega_o(\boldsymbol{\mu})$, and indicate with the subscript o (*original*) all the spaces, functions, and structures that refer to this domain. In particular, the parametrized geometry $\Omega_o(\boldsymbol{\mu})$ is built by means of the NURBS space $\mathcal{B}_{ho}$ defined as

$$\mathcal{B}_{ho}(\boldsymbol{\mu}) = \text{span}\{\mathcal{R}_{oj}(\boldsymbol{\mu})\}_{j=1}^{n} = \text{span}\{R_{oj} \circ \boldsymbol{F}^{-1}\}_{j=1}^{n}, \tag{3.1}$$

where $\mathcal{R}_{oj}$ are the NURBS basis functions and $\boldsymbol{F}$ the geometrical mapping defining the parametrized domain as follows:

$$\boldsymbol{F} : \hat{\Omega} \times \mathcal{D} \to \Omega_o(\boldsymbol{\mu}) \subset \mathbb{R}^d, \quad (\boldsymbol{\xi}; \boldsymbol{\mu}) \to \boldsymbol{F}(\boldsymbol{\xi}; \boldsymbol{\mu}) = \sum_{j=1}^{n} R_{oj}(\boldsymbol{\xi}) \boldsymbol{B}_j(\boldsymbol{\mu}), \tag{3.2}$$

where $\boldsymbol{B}_j(\boldsymbol{\mu})$ are parametrized control points in $\mathbb{R}^d$, for $d = 2$. The parametrized version of the Poisson problem (1.55) then reads: given $\boldsymbol{\mu} \in \mathcal{D}$, find $u_o : \Omega_o(\boldsymbol{\mu}) \to \mathbb{R}$ such that

$$-\nabla \cdot (\delta \nabla u_o(\boldsymbol{\mu})) = f_o(\boldsymbol{\mu}) \qquad \text{in} \quad \Omega_o(\boldsymbol{\mu}) \tag{3.3a}$$

$$u_o(\boldsymbol{\mu}) = 0 \qquad \text{on} \quad \partial\Omega_o(\boldsymbol{\mu}) \tag{3.3b}$$

where we have assumed Dirichlet homogeneous boundary conditions. We introduce the weak formulation of the problem and refer to it as to the *original problem*. The variational form of the problem reads: given $\boldsymbol{\mu} \in \mathcal{D}$, find $u_o(\boldsymbol{\mu}) \in V_o$ such that

$$a_o(u_o(\boldsymbol{\mu}), v; \boldsymbol{\mu}) = L_o(v; \boldsymbol{\mu}), \quad \forall v \in V_o \tag{3.4}$$

where $V_o = V_o(\boldsymbol{\mu}) = H_0^1(\Omega_o(\boldsymbol{\mu}))$. The associated parameter dependent bilinear form a_o :

$V_o \times V_o \to \mathbb{R}$ and the linear functional $L_o : V_o \to \mathbb{R}$ can be expressed as

$$a_o(w, v; \boldsymbol{\mu}) = \int_{\Omega_o(\boldsymbol{\mu})} \left[\frac{\partial w}{\partial x_{o1}}, \frac{\partial w}{\partial x_{o2}}\right] \boldsymbol{\nu}_o(\boldsymbol{x}; \boldsymbol{\mu}) \left[\frac{\partial v}{\partial x_{o1}}, \frac{\partial v}{\partial x_{o2}}\right]^T d\Omega_o(\boldsymbol{\mu}) \tag{3.5a}$$

$$L_o(v; \boldsymbol{\mu}) = \int_{\Omega_o(\boldsymbol{\mu})} f_o(\boldsymbol{x}; \boldsymbol{\mu})\, v\, d\Omega_o(\boldsymbol{\mu}), \tag{3.5b}$$

where $\boldsymbol{\nu}_o \in \mathbb{R}^{2\times 2}$ is a (symmetric positive definite) diffusivity tensor, in this case $\boldsymbol{\nu}_o = \delta \boldsymbol{I}$, with $\boldsymbol{I} \in \mathbb{R}^{2\times 2}$ the identity matrix, and $f_o : \mathbb{R}^2 \times \mathcal{D} \to \mathbb{R}$ are prescribed coefficients.

As anticipated in Section 2.3, the RB method requires a parameter independent domain in order to compute and combine the IGA solutions that will be used as basis of the RB approximation space. For this reason, we need to map the original domain $\Omega_o(\boldsymbol{\mu})$ to a reference domain Ω to recast problem (3.4) in the form (2.9).

3.2 Parametrized formulation on a reference domain

In this section we treat the shape parametrization by transforming equation (3.4) in $\Omega_o(\boldsymbol{\mu})$ to a new equation on the fixed reference domain Ω. To do this, we use the geometrical map $\boldsymbol{T}$ between Ω and $\Omega_o(\boldsymbol{\mu})$ introduced in Section 2.3.

$$\boldsymbol{T}(\cdot; \boldsymbol{\mu}) : \Omega \to \Omega_o(\boldsymbol{\mu}), \tag{3.6}$$

such that $\Omega_o(\boldsymbol{\mu}) = \boldsymbol{T}(\Omega; \boldsymbol{\mu})$. The abstract weak formulation of the *reference problem* reads: find $u(\boldsymbol{\mu}) \in V$ such that

$$a(u(\boldsymbol{\mu}), v; \boldsymbol{\mu}) = L(v; \boldsymbol{\mu}), \quad \forall v \in V, \tag{3.7}$$

where $V = H_0^1(\Omega)$ and the bilinear form and the linear functional can be expressed as

$$a(w, v; \boldsymbol{\mu}) = \int_{\Omega} \left[\frac{\partial w}{\partial x_1}, \frac{\partial w}{\partial x_2}\right] \boldsymbol{\nu}(\boldsymbol{x}; \boldsymbol{\mu}) \left[\frac{\partial v}{\partial x_1}, \frac{\partial v}{\partial x_2}\right]^T d\Omega \tag{3.8a}$$

$$L(v; \boldsymbol{\mu}) = \int_{\Omega} f(\boldsymbol{x}; \boldsymbol{\mu})\, v\, d\Omega, \tag{3.8b}$$

$\boldsymbol{\nu}(\boldsymbol{x}; \cdot) : \mathbb{R}^2 \times \mathcal{D} \to \mathbb{R}^{2\times 2}$ is a parametrized tensor given by

$$\boldsymbol{\nu}(\boldsymbol{x}; \boldsymbol{\mu}) = (\boldsymbol{J_T}(\boldsymbol{x}; \boldsymbol{\mu}))^{-1} \boldsymbol{\nu}_o(\boldsymbol{J_T}(\boldsymbol{x}; \boldsymbol{\mu}))^{-T} \det(\boldsymbol{J_T}(\boldsymbol{x}; \boldsymbol{\mu})), \tag{3.9}$$

$\boldsymbol{J_T}(\boldsymbol{x}; \boldsymbol{\mu}) : \mathbb{R}^2 \times \mathcal{D} \to \mathbb{R}^{2\times 2}$ is the Jacobian matrix of the map $\boldsymbol{T}(\boldsymbol{x}; \boldsymbol{\mu})$, defined as

$$\boldsymbol{J_T}(\boldsymbol{x}; \boldsymbol{\mu})_{ij} = \frac{\partial (\boldsymbol{T})_i}{\partial x_j}(\boldsymbol{x}; \boldsymbol{\mu}) \quad i, j = 1, ..., d$$

and $\det(\boldsymbol{J_T}(\boldsymbol{x}; \boldsymbol{\mu}))$ its determinant. Finally, $f(\boldsymbol{x}; \cdot) : \mathbb{R}^2 \times \mathcal{D} \to \mathbb{R}$ is given by

$$f(\boldsymbol{x}; \boldsymbol{\mu}) = f_o(\boldsymbol{x}; \boldsymbol{\mu}) \det(\boldsymbol{J_T}(\boldsymbol{x}; \boldsymbol{\mu})).$$

Through the parametric transformation, we have reformulated the original problem on a reference configuration Ω and so obtained a parametrized problem, where the effect of geometry variations is traced back onto its parametrized transformation tensors. As we will see in Section 3.4 also Ω will be a NURBS surface.

3.3 Numerical approximation for parametrized problem

We can now derive the numerical approximation of the variational formulation of the problem at hand through a NURBS-based IGA Galerkin method as done in (1.37) and so look for a discrete solution of (3.7) in the finite dimensional space $V^{\mathcal{N}} \subset V$. Let us suppose to have already specified how to define the reference domain Ω. In virtue of the *isogeometric concept*, the space $V^{\mathcal{N}}$ will be built by means of the same NURBS basis functions defining the reference geometry. In particular, by indicating these functions with $\mathcal{R}_i$, it results that $V^{\mathcal{N}} = V \cap \mathcal{B}_h$, with $\mathcal{B}_h = \text{span}(\{\mathcal{R}_j\}_{j=1}^n)$. Following the same procedure adopted in (1.38) we assume that the solution is expressed as

$$u^{\mathcal{N}}(\boldsymbol{x};\boldsymbol{\mu}) = \sum_{j=1}^{\mathcal{N}} \mathcal{R}_j(\boldsymbol{x}) u_j(\boldsymbol{\mu}). \tag{3.10}$$

By using this expression of $u^{\mathcal{N}}$ and choosing $v = \mathcal{R}_i$, $1 \le i \le \mathcal{N}$ we can rewrite problem (3.7) as a linear system

$$\boldsymbol{A}(\boldsymbol{\mu})\boldsymbol{u}(\boldsymbol{\mu}) = \boldsymbol{L}(\boldsymbol{\mu}), \tag{3.11}$$

where from (3.8) it results that

$$(\boldsymbol{A}(\boldsymbol{\mu}))_{ij} = a(\mathcal{R}_j, \mathcal{R}_i; \boldsymbol{\mu}) = \sum_{q=1}^{Q} a^q(\mathcal{R}_j, \mathcal{R}_i; \boldsymbol{\mu}), \tag{3.12a}$$

$$(\boldsymbol{L}(\boldsymbol{\mu}))_i = L(\mathcal{R}_i; \boldsymbol{\mu}). \tag{3.12b}$$

with $Q = 4$ and

$$a^1(\mathcal{R}_j, \mathcal{R}_i; \boldsymbol{\mu}) = \int_\Omega \nu_{11}(\boldsymbol{x};\boldsymbol{\mu}) \frac{\partial \mathcal{R}_j}{\partial x_1} \frac{\partial \mathcal{R}_i}{\partial x_1} d\Omega, \tag{3.13a}$$

$$a^2(\mathcal{R}_j, \mathcal{R}_i; \boldsymbol{\mu}) = \int_\Omega \nu_{12}(\boldsymbol{x};\boldsymbol{\mu}) \frac{\partial \mathcal{R}_j}{\partial x_1} \frac{\partial \mathcal{R}_i}{\partial x_2} d\Omega, \tag{3.13b}$$

$$a^3(\mathcal{R}_j, \mathcal{R}_i; \boldsymbol{\mu}) = \int_\Omega \nu_{21}(\boldsymbol{x};\boldsymbol{\mu}) \frac{\partial \mathcal{R}_j}{\partial x_2} \frac{\partial \mathcal{R}_i}{\partial x_1} d\Omega, \tag{3.13c}$$

$$a^4(\mathcal{R}_j, \mathcal{R}_i; \boldsymbol{\mu}) = \int_\Omega \nu_{22}(\boldsymbol{x};\boldsymbol{\mu}) \frac{\partial \mathcal{R}_j}{\partial x_2} \frac{\partial \mathcal{R}_i}{\partial x_2} d\Omega, \tag{3.13d}$$

$$L(\mathcal{R}_i; \boldsymbol{\mu}) = \int_\Omega f(\boldsymbol{x};\boldsymbol{\mu}) \mathcal{R}_i d\Omega. \tag{3.13e}$$

In order to fully exploit the Offline-Online splitting, we need to write a and L as in (2.6) thus expressing them as a linear combination of $\boldsymbol{\mu}$-dependent functions and $\boldsymbol{\mu}$-independent matrices and vectors. This form is obtained naturally if the problem has an affine parametric dependence. Unfortunately, this is not our case because $\nu_{ij}(\boldsymbol{x};\boldsymbol{\mu})$ and $f(\boldsymbol{x};\boldsymbol{\mu})$, for $1 \le i, j \le 2$, are non-linear functions of the spatial coordinate $\boldsymbol{x}$ and of the vector parameter $\boldsymbol{\mu}$. Since

the isogeometric parametrization entails a non affine parametric dependence, we need to approximate the nonaffinely parametric dependent terms. This procedure will be carried out by means of the Empirical Interpolation Method in Section 3.5.

3.4 Reference domain

In this section we explain the criteria followed to choose the reference domain Ω. First, let us summarize how the parametric, the reference, and the original domains interact among each other in the integration operations involved in the IGA-RB framework:

- RB requires to trace back the bilinear form and the linear functional (3.5) from the original domain $\Omega_o(\boldsymbol{\mu})$ to the reference domain Ω (3.8);
- in IGA the assembly of $\boldsymbol{A}$ and $\boldsymbol{L}$ in (3.11) is made at the element level by pulling back the basis functions having support on a mesh element of the reference domain K_k (1.46) into the corresponding element of the parameter domain $\hat{K}_k$ and then performing numerical integration by means of quadrature formulas (1.50);
- in a IGA-RB combined methodology, to assemble (3.11), we have first to trace back problem (3.4) from the original domain $\Omega_o(\boldsymbol{\mu})$ to the reference domain Ω, obtaining (3.8) that in matrix form reads as in (3.11); the latter is then pulled back to the parametric domain $\hat{\Omega}$ to perform the numerical integration (1.50).

In this scenario, in addition to the already known maps $\boldsymbol{F}$ and $\boldsymbol{T}$, it is convenient to introduce a new map from the parametric domain to the reference one, $\boldsymbol{S} : \widehat{\Omega} \to \Omega$. We now recap the three maps we are considering and give a representation of them in Fig. 3.1

- $\boldsymbol{S} : \hat{\Omega} \to \Omega$, map from the NURBS *parametric domain* to the NURBS *reference domain*;
- $\boldsymbol{T}(\boldsymbol{\mu}) : \Omega \to \Omega_o(\boldsymbol{\mu})$, map from the NURBS *reference domain* to the NURBS *original domain*;
- $\boldsymbol{F}(\boldsymbol{\mu}) : \hat{\Omega} \to \Omega_o(\boldsymbol{\mu})$, map from the NURBS *parametric domain* to the NURBS *original domain*;

So far, we have referred to a generic reference configuration Ω. Once clarified the scenario we are dealing with, we can finally proceed with the construction of a proper reference domain and give an explicit expression of the parametrization $\boldsymbol{T}$ that leads from Ω to Ω_o. In practice there are several techniques for the construction of this parametric domain map $\boldsymbol{T}$, like the Free-Form Deformation and the Radial Basis Function techniques (see [Man12]). Anyway, in a Isogeometric framework the natural choice is to use, in a proper way, the geometrical map $\mathbf{F}$ so to be allowed to directly use its explicit expression (1.18), its jacobian, and its determinant as requested in (3.5). In this section, with the simple aim to define the reference domain, we consider as original domain Ω_o the one depicted in Fig. 3.2, but the procedure we describe is independent of the original domain considered.

A first idea could be to assume as reference domain the parametric one. In this way, since $\hat{\Omega} \equiv \Omega$ and $\boldsymbol{S} = \boldsymbol{I}$, with $\boldsymbol{I}$ the identity map, the map $\boldsymbol{T}(\boldsymbol{\mu})$ would automatically coincide with $\boldsymbol{F}(\boldsymbol{\mu})$; then, we could easily go from $\Omega_o(\boldsymbol{\mu})$ back to $\widehat{\Omega}$, and perform on it all the operations illustrated in Section 3.2. To this end, we should simply define a unit square geometry (as the

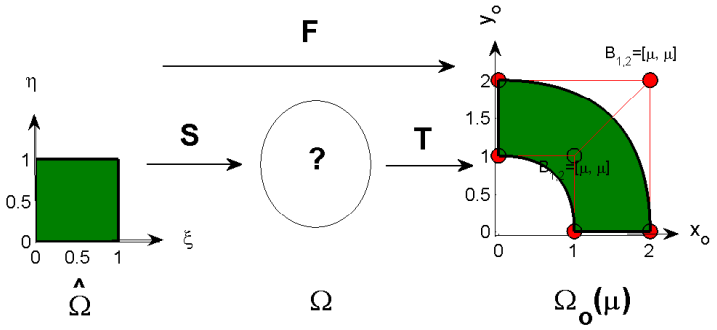


Figure 3.1: The three maps $\boldsymbol{S}$, $\boldsymbol{T}$, $\boldsymbol{F}$, and the three domains, parametric $\hat{\Omega}$, reference Ω, original $\Omega_o(\boldsymbol{\mu})$, involved in the IGA-RB methodology.

parametric 2D domain) and assemble on it the matrices and vectors of the reduced system we want to solve. To this end, we have to decide how to properly set the control points and NURBS weights to represent the unit square.

We start by building Ω by means of the space $\mathcal{B}_h = \text{span}\{N_i \circ \boldsymbol{S}^{-1}, i = 1, ..., 4\}$ of B-spline basis functions of degree $p = q = 1$ in both directions as in Fig. 3.2. It is clear that in this way $\Omega_o(\boldsymbol{\mu})$ and Ω are not built by means of the same NURBS basis functions, since $\Omega_o(\boldsymbol{\mu})$ is built by the space $\mathcal{B}_{oh} = \text{span}\{R_{oi} \circ \boldsymbol{F}^{-1}, i = 1, ..., 6\}$ of NURBS basis functions of degree $p = 1$ in radial direction and $q = 2$ in angular direction. The problem is that, when tracing back the bilinear form and the linear functional from $\Omega_o(\boldsymbol{\mu})$ to Ω, the basis functions used in the integration must be the same. In particular, since both the integration on Ω_o and Ω are actually traced back into $\hat{\Omega}$, we should have that $\hat{\mathcal{B}}_{oh} \equiv \hat{\mathcal{B}}_h$. However, in our case the spaces $\mathcal{B}_{oh}$ and $\mathcal{B}_h$ traced back in the parametric domain $\hat{\Omega}$ read as $\hat{\mathcal{B}}_{oh} = \text{span}\{R_{oi}, i = 1, ..., 6\}$ and $\hat{\mathcal{B}}_h = \text{span}\{N_i, i = 1, ..., 4\}$, respectively. So we should decide how to set the control points and NURBS weights in such a way Ω is a 2D unit square built by using the same NURBS basis functions used to build $\Omega_o(\boldsymbol{\mu})$ so that $\hat{\mathcal{B}}_{oh} \equiv \hat{\mathcal{B}}_h$. To this end, we build the square by choosing $p = 1$ (in x direction) and $q = 2$ (in y direction) and by assigning the same NURBS weights used for the original domain as shown in Fig. 3.3.

With this choice of the reference domain it results that $\hat{\mathcal{B}}_{oh} \equiv \hat{\mathcal{B}}_h$, $\boldsymbol{F} = \boldsymbol{T} \circ \boldsymbol{S}$ and as consequence

$$\boldsymbol{J_F} = \boldsymbol{J}[(\boldsymbol{F})(\boldsymbol{\xi})] = \boldsymbol{J}[(\boldsymbol{T} \circ \boldsymbol{S})(\boldsymbol{\xi})] = \boldsymbol{J}[\boldsymbol{T}(\boldsymbol{S}(\boldsymbol{\xi}))]\boldsymbol{J}[\boldsymbol{S}(\boldsymbol{\xi})], \tag{3.14}$$

and so

$$\boldsymbol{J_T} = \boldsymbol{J}[\boldsymbol{T}(\boldsymbol{S}(\boldsymbol{\xi}))] = \boldsymbol{J}[\boldsymbol{F}(\boldsymbol{\xi})](\boldsymbol{J}[\boldsymbol{S}(\boldsymbol{\xi})])^{-1} = \boldsymbol{J_F}\boldsymbol{J_S}^{-1}, \tag{3.15}$$

and

$$\det(\boldsymbol{J_T}) = \det(\boldsymbol{J_T}(\boldsymbol{S}(\boldsymbol{\xi}))) = \det(\boldsymbol{J_F})\det(\boldsymbol{J_S}^{-1}) = \det(\boldsymbol{J_F})(\det(\boldsymbol{J_S}))^{-1}. \tag{3.16}$$

Finally, expressions (3.15) and (3.16) can be substituted in (3.9) and (3.2) for the actual evaluation of the solution of the reference problem.

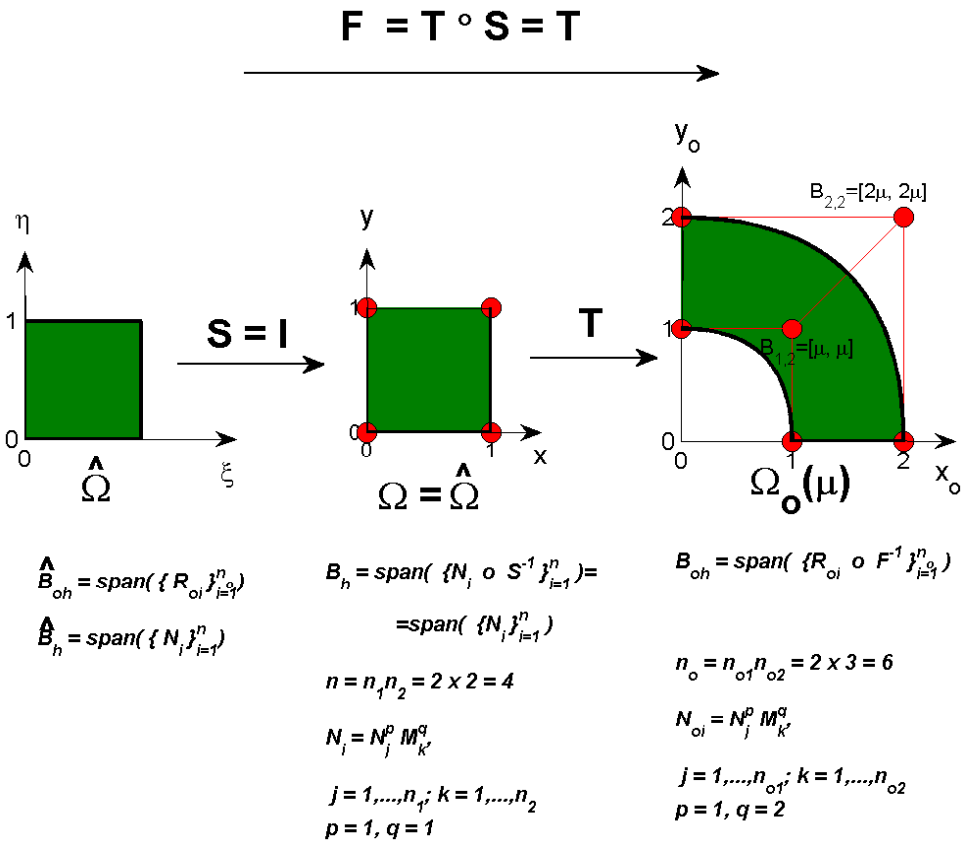


Figure 3.2: If we choose $\Omega \equiv \hat{\Omega}$, the NURBS space in the original domain ($\mathcal{B}_{oh}$) and the one in the reference domain ($\mathcal{B}_h$), if traced back into the parameter domain as $\hat{\mathcal{B}}_{oh}$ and $\hat{\mathcal{B}}_h$ respectively, do not coincide.

3.5 Empirical Interpolation Method

In this section we apply the Empirical Interpolation Method (EIM) [BMNP04, GMNP07, BMS14], a technique that allows to efficiently treat nonaffine problems in the RB framework. As a matter of fact, EIM recovers the assumption of affine parametric dependence thanks to a suitable approximation of the nonaffine terms appearing in the problem operators. However, in Chapter 4 we will see that EIM can not be applied if the NURBS weights are parametrized and in this case we will use another methodology.

We describe EIM for the general case of a non linear function $f : \Omega \times \mathcal{D} \to f(\boldsymbol{x}, \boldsymbol{\mu}) \subset \mathbb{R}$, depending both on $\boldsymbol{x}$ and $\boldsymbol{\mu}$ in a nonaffine way. In particular, we present a discrete version of

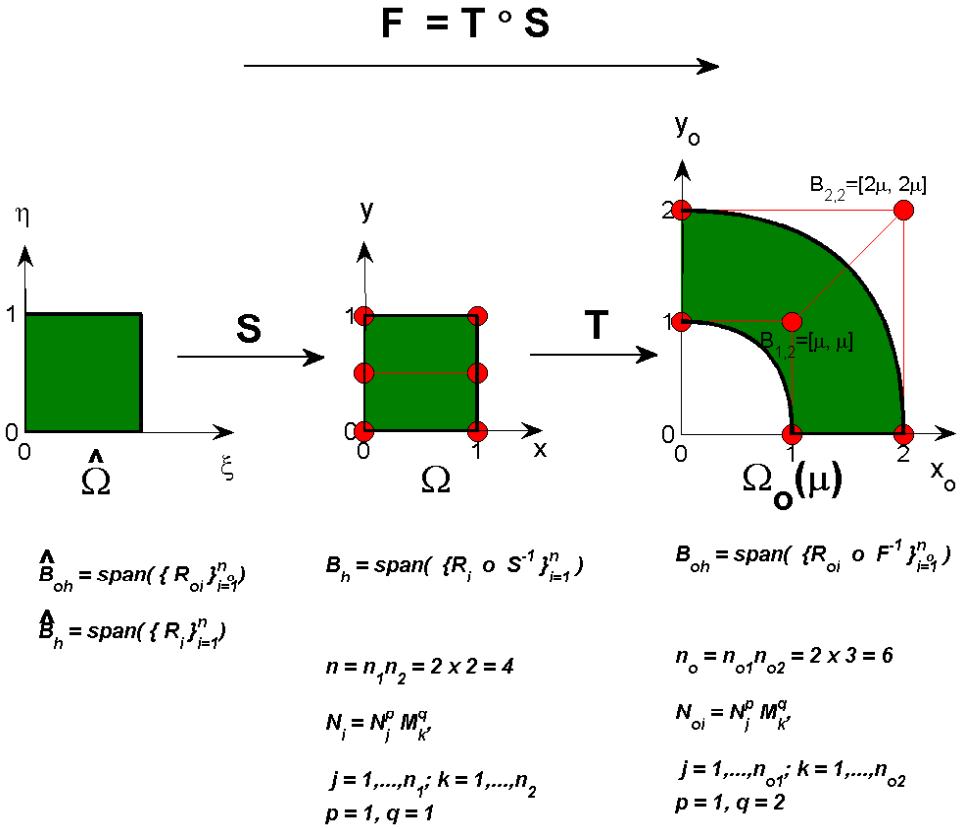


Figure 3.3: The reference domain Ω is assembled as a 2D unit square by means of the same NURBS basis functions used to build the original domain $\Omega_o(\boldsymbol{\mu})$. These spaces are traced back into the parameter domain as $\hat{\mathcal{B}}_{oh}$ and $\hat{\mathcal{B}}_h$, respectively. In this case $\hat{\mathcal{B}}_{oh}$ coincides with $\hat{\mathcal{B}}_h$.

EIM that operates on $\Omega_h \subset \Omega$, with Ω_h a set of n points in $\mathbb{R}^d$, $d = 2$ or 3. In the following we will assume a reordering of the points of Ω_h in a matrix structure $\boldsymbol{Q} \in \mathbb{R}^{n \times d}$. In the next section we will consider Ω_h to be the set of the quadrature nodes of the computational domain.

Thus, we aim to approximate $\boldsymbol{f} : \mathcal{D} \to \mathbb{R}^n$, where $f_i = f(\boldsymbol{x}_i, \boldsymbol{\mu})$, with $\boldsymbol{x}_i$ a row of $\boldsymbol{Q}$ for $i = 1, ..., n$. The idea of EIM is to find an approximate expansion of the form

$$\boldsymbol{f}(\boldsymbol{\mu}) \approx \boldsymbol{f}_M(\boldsymbol{\mu}) = \boldsymbol{\Phi}\boldsymbol{\Theta}(\boldsymbol{\mu}), \tag{3.17}$$

where $\boldsymbol{\Phi} = [\boldsymbol{\phi}_1, ..., \boldsymbol{\phi}_M] \in \mathbb{R}^{n \times M}$ is a basis that can be computed offline, and $\boldsymbol{\Theta}(\boldsymbol{\mu}) \in \mathbb{R}^M$ is a coefficient vector to be computed online. Moreover, as we are going to see in the algorithm,

EIM seeks a set of indexes $\mathcal{F} \subset \{1, ..., n\}$, $|\mathcal{F}| = M$, such that to each index corresponds a point of $\boldsymbol{Q}$ and the selected points are called *magic points*. Being an interpolation procedure, EIM uses the properly selected indexes (or magic points) and the basis in order to compute the coefficient vector $\boldsymbol{\Theta}(\boldsymbol{\mu})$, by solving the following interpolation problem:

$$\boldsymbol{\Phi}_{\mathcal{F}}\boldsymbol{\Theta}(\boldsymbol{\mu}) = \boldsymbol{f}_{\mathcal{F}}(\boldsymbol{\mu}), \tag{3.18}$$

where $\boldsymbol{\Phi}_{\mathcal{F}} \in \mathbb{R}^{M\times M}$ is the matrix formed by the $\mathcal{F}$ rows of $\boldsymbol{\Phi}$, and $\boldsymbol{f}_{\mathcal{F}}(\boldsymbol{\mu}) \in \mathbb{R}^M$ is the vector we obtain by evaluating $\boldsymbol{f}(\boldsymbol{\mu})$ in the points of $\boldsymbol{Q}$ whose index is in $\mathcal{F}$. EIM selects both the interpolation indexes and basis functions by means of a greedy algorithm. Let us denote by $\Xi_{train} \in \mathcal{D}$ a large training set of parameter samples, M_{max} the maximum number of terms in the expansion, tol_{EIM} a fixed tolerance, and select an initial parameter $\boldsymbol{\mu}^1$. The EIM procedure is as follows:

$M = 1$;
$\boldsymbol{\phi}_1 = \boldsymbol{f}(\boldsymbol{\mu}^1)$; compute $i = \arg\max |\boldsymbol{\phi}_1|$; $\mathcal{F} = \{i\}$;
$\boldsymbol{\phi}_1 = \boldsymbol{\phi}_1/(\boldsymbol{\phi}_1)_i$; $\boldsymbol{\Phi} = [\boldsymbol{\phi}_1]$;
for $M = 2 : M_{max}$
  solve $\boldsymbol{\mu}^M := \arg\max_{\boldsymbol{\mu}\in\Xi_{train}} \|\boldsymbol{f}(\boldsymbol{\mu}) - \boldsymbol{f}_{M-1}(\boldsymbol{\mu})\|_\infty$
  set $\boldsymbol{\phi}_M = \boldsymbol{f}(\boldsymbol{\mu}^M)$; $\boldsymbol{\Phi} = [\boldsymbol{\Phi}, \boldsymbol{\phi}_M]$;
  set $\boldsymbol{\sigma}_M = \boldsymbol{\Phi}_{\mathcal{F}}^{-1}(\boldsymbol{\phi}_M)_{\mathcal{F}}$;
  compute $\boldsymbol{\phi}_M = \boldsymbol{\phi}_M - \sum_{j=1}^{M-1}(\boldsymbol{\sigma}_M)_j\boldsymbol{\phi}_j$;
  compute $i = \arg\max |\boldsymbol{\phi}_M|$; $\mathcal{F} = \mathcal{F} \cup \{i\}$;
  if $\max_{\boldsymbol{\mu}\in\Xi_{train}} \|\boldsymbol{f}(\boldsymbol{\mu}) - \boldsymbol{f}_{M-1}(\boldsymbol{\mu})\|_\infty < \mathrm{tol}_{EIM}$
    $M_{max} = M - 1$;
  end
end

In the following we apply EIM to problem (3.7) by applying the previous algorithm to the parameter dependent functions appearing in the integrals (3.13).

3.6 EIM application

We now apply EIM to the nonaffine bilinear forms and the linear functional in (3.13). More precisely, we approximate with EIM the coefficients $\nu_{ij}(\boldsymbol{x};\boldsymbol{\mu})$, for $1 \le i, j \le 2$, and $f(\boldsymbol{x};\boldsymbol{\mu})$. So, as done in Section 3.5, we build the sets of indexes $\mathcal{F}_{\nu_{11}}$ of cardinality $M^{\nu^{11}}$,..., $\mathcal{F}_{\nu_{22}}$ of cardinality $M^{\nu^{22}}$, $\mathcal{F}_f$ of cardinality M^f, and basis matrices

$$\boldsymbol{\Phi}^{\nu_{11}} = [\boldsymbol{\phi}_1^{\nu_{11}}, ..., \boldsymbol{\phi}_{M^{\nu_{11}}}^{\nu_{11}}], ..., \boldsymbol{\Phi}^{\nu_{22}} = [\boldsymbol{\phi}_1^{\nu_{22}}, ..., \boldsymbol{\phi}_{M^{\nu_{22}}}^{\nu_{22}}], \boldsymbol{\Phi}^f = [\boldsymbol{\phi}_1^f, ..., \boldsymbol{\phi}_{M^f}^f]. \tag{3.19}$$

First, we choose as set of points Ω_h on which to perform EIM the mesh quadrature nodes. For this reason, we start by realizing the same h-refinement on both the original domain Ω_o and the reference domain Ω; in this way, we have the same number of mesh elements between the two domains and, since we have built $\Omega_o(\boldsymbol{\mu})$ and Ω by means of the same basis functions, they will have the same number of quadrature nodes as well. In the following, we indicate

with Ω_h the set of the images of the quadrature nodes in the reference domain Ω

$$\Omega_h = \{\boldsymbol{x}_{l,k} \in \Omega : \boldsymbol{x}_{l,k} = \boldsymbol{S}(\boldsymbol{\xi}_{l,k}), \text{for } l = 1, ..., N_e \text{ and } k = 1, ..., n_k\},$$

and with $\Omega_{oh}(\boldsymbol{\mu})$ the set of the images of the quadrature nodes in the original domain Ω_o

$$\Omega_{oh}(\boldsymbol{\mu}) = \{\boldsymbol{x}^o_{l,k} \in \Omega_o(\boldsymbol{\mu}) : \boldsymbol{x}^o_{l,k} = \boldsymbol{F}(\boldsymbol{\xi}_{l,k}; \boldsymbol{\mu}), \text{for } l = 1, ..., N_e \text{ and } k = 1, ..., n_k\},$$

where $\boldsymbol{\xi}_{l,k}$ are the quadrature nodes in the parametric domain (1.47), N_e is the number of mesh elements, and n_k is the number of quadrature nodes for each mesh element. We notice that, since the reference domain does not depend on $\boldsymbol{\mu}$, the points of Ω_h can be computed once and for all, while the ones of $\Omega_{oh}(\boldsymbol{\mu})$ must be recomputed for each new parameter $\boldsymbol{\mu}$. We introduce a reordering of the quadrature nodes and a global numbering, and rewrite them in matrix form as: $\boldsymbol{Q} \in \mathbb{R}^{n \times d}$ and $\boldsymbol{Q_o} \in \mathbb{R}^{n \times d}$, with n the total number of quadrature nodes and d the dimension of the physical space. Then, we perform EIM on this set of points by following the algorithm reported in the previous section.

Let us focus, for instance, on the term $\nu_{11}(\boldsymbol{x}; \boldsymbol{\mu})$. Given the parameter $\boldsymbol{\mu}^1$, we modify the control points coordinates of the original domain, compute $\boldsymbol{Q}_o(\boldsymbol{\mu})$, evaluate on this points $\boldsymbol{J_F}$, the Jacobian of the transformation $\boldsymbol{F}$, and compute once and for all $\boldsymbol{J_S}$, the Jacobian of the transformation $\boldsymbol{S}$ in the points of the set $\boldsymbol{Q}$. Then, computing (3.9) for each point of $\boldsymbol{Q}_o$, we finally obtain the vector $\boldsymbol{\nu}_{11} \in \mathbb{R}^n$. Then, we set $\boldsymbol{\phi}_1^{\nu_{11}} = \boldsymbol{\nu}_{11}$ and apply the EIM algorithm illustrated in Section 3.5. Finally, as output of the algorithm, we obtain the set of indexes $\mathcal{F}_{\nu_{11}}$ of cardinality $M^{\nu^{11}}$ and the basis matrix $\boldsymbol{\Phi}^{\nu_{11}} = [\boldsymbol{\phi}_1^{\nu_{11}}, ..., \boldsymbol{\phi}_{M^{\nu_{11}}}^{\nu_{11}}]$. In this way we obtain the desired approximation

$$\boldsymbol{\nu}_{11}(\boldsymbol{\mu}) \approx \boldsymbol{\nu}_M(\boldsymbol{\mu}) = \boldsymbol{\Phi}^{\nu_{11}} \boldsymbol{\Theta}^{\nu_{11}}(\boldsymbol{\mu}) \tag{3.20}$$

where the $\boldsymbol{\mu}$-dependent coefficient vector $\boldsymbol{\Theta}^{\nu_{11}}(\boldsymbol{\mu})$ is obtained by solving online the Lagrange interpolation problem (3.18) that in this case can be rewritten as

$$\boldsymbol{\Theta}^{\nu_{11}}(\boldsymbol{\mu}) = (\boldsymbol{\Phi}^{\nu_{11}}_{\mathcal{F}})^{-1}(\boldsymbol{\nu}_{11})_{\mathcal{F}}. \tag{3.21}$$

In practice online, given a parameter $\boldsymbol{\mu}$, we modify the original domain and compute $\nu_{11}(\cdot; \boldsymbol{\mu})$ only in the images of the quadrature nodes corresponding to the indexes of $\mathcal{F}_{\nu_{11}}$, that is in the images of the *magic points*, and finally compute (3.21). Moreover, for each basis $\boldsymbol{\phi}_m^{\nu_{11}}$, we define a function $\psi_m^{\nu_{11}}$ such that for each quadrature node $\boldsymbol{\xi}_{l,k}$:

$$\psi_m^{\nu_{11}}(\boldsymbol{\xi}_{l,k}) = (\boldsymbol{\phi}_m^{\nu_{11}})_i, \tag{3.22}$$

where i is the index of that quadrature node in the global numbering introduced at the begin. The reason of introducing these functions will be clarified later.

By repeating the procedure for the other nonaffine terms, we obtain

$$\boldsymbol{\nu}_{M_{ij}}(\boldsymbol{\mu}) = \boldsymbol{\Phi}^{\nu_{ij}} \boldsymbol{\Theta}^{\nu_{ij}}(\boldsymbol{\mu}) \quad \text{for } 1 \leq i, j \leq 2 \tag{3.23}$$

$$\boldsymbol{f}_M(\boldsymbol{\mu}) = \boldsymbol{\Phi}^f \boldsymbol{\Theta}^f(\boldsymbol{\mu}). \tag{3.24}$$

In the following, in order to simplify the notation, we will rename $\nu^1 = \nu_{11}$, $\nu^2 = \nu_{12}$, $\nu^3 = \nu_{21}$, $\nu^4 = \nu_{22}$. By substituting these expressions in (3.13) we get the final EIM approximate system:

$$A_M(\boldsymbol{\mu}) \boldsymbol{u}_M(\boldsymbol{\mu}) = L_M(\boldsymbol{\mu}), \tag{3.25}$$

that explicitely reads:

$$\left(\sum_{q=1}^{Q}\sum_{m=1}^{M^{\nu^q}}\Theta_m^{\nu_q}(\boldsymbol{\mu})\boldsymbol{A}^{qm}\right)\boldsymbol{u}_M(\boldsymbol{\mu})=\sum_{m=1}^{M^f}\Theta_m^f(\boldsymbol{\mu})\boldsymbol{L}^m, \tag{3.26}$$

where for example for q=1

$$\boldsymbol{A}_{ij}^{1m}=a^1(R_j,R_i,\psi_m^{\nu^1})=\int_\Omega\psi_m^{\nu^1}\frac{\partial R_j}{\partial x_1}\frac{\partial R_i}{\partial x_1}d\Omega, \tag{3.27a}$$

$$\boldsymbol{L}_i^m=L(R_i)=\int_\Omega\psi_m^f R_i d\Omega \tag{3.27b}$$

Here $\boldsymbol{A}^{qm}\in\mathbb{R}^{\mathcal{N}\times\mathcal{N}}$, $1\le q\le Q$, $1\le m\le M^{\nu^q}$, and $\boldsymbol{L}^m\in\mathbb{R}^{\mathcal{N}}$, $1\le m\le M_f$. It is now clear that we have introduced the functions ψ defined in (3.22), just to take into account the contribution of the basis in the integration. These matrices and vectors do not depend on $\boldsymbol{\mu}$ and can be precomputed offline only once. The other operations required to assemble and solve system (3.26) are performed online.

3.7 Approximation with RB

The linear system (3.25) has dimension $\mathcal{N}\times\mathcal{N}$ and we would like to reduce its dimension by adopting the RB approach. We do that by computing a set of basis by means of a POD or greedy approach (see Sections 2.4.3 and 2.4.2, respectively) to obtain reduced matrices and right hand side vectors as in (2.32). In this way, we obtain the following reduced system

$$\left(\sum_{q=1}^{Q}\sum_{m=1}^{M^{\nu^q}}\boldsymbol{\Theta}_m^{\nu^q}(\boldsymbol{\mu})\boldsymbol{A}_N^{qm}\right)\boldsymbol{u}_{N,M}(\boldsymbol{\mu})=\sum_{m=1}^{M^f}\boldsymbol{\Theta}_m^f(\boldsymbol{\mu})\boldsymbol{L}_N^m, \tag{3.28}$$

that can be rewritten in compact form as

$$\boldsymbol{A}_{N,M}(\boldsymbol{\mu})\boldsymbol{u}_{N,M}=\boldsymbol{L}_{N,M}(\boldsymbol{\mu}), \tag{3.29}$$

that is a system of dimension $N\times N$ with $N\ll\mathcal{N}$. Once we have computed the RB solution, we can recover its full-order expansion as

$$\boldsymbol{u}_{N,M}^{\mathcal{N}}(\boldsymbol{\mu})=\boldsymbol{\mathcal{Z}}\boldsymbol{u}_{N,M}(\boldsymbol{\mu}). \tag{3.30}$$

being $\boldsymbol{\mathcal{Z}}\in\mathbb{R}^{\mathcal{N}\times N}$ the basis matrix defined in (2.20) or in (2.25), if a greedy or a POD sampling strategy has been adopted respectively.

3.8 Numerical results

In this section we present the numerical results obtained for the solution of problem (3.3), by performing first an EIM approximation and then a RB reduction. In particular, in the first example we introduce only one geometric parameter, while in the second one we consider two geometric parameters.

Table 3.1: EIM data, one parameter case.

EIM samples	50
EIM tolerance tol_{eim}	10^{-5}
Affine matrix components $\boldsymbol{A}^{qm}$	$Q_a = 16$
Affine rhs components $\boldsymbol{L}^m$	$Q_f = 2$

3.8.1 One parameter case

To solve the problem we take $\delta = 1$, $f_o(\boldsymbol{\mu}) = 1$, and consider homogeneous Dirichlet boundary conditions. As parametrized computational domain we choose the one built in Example 1.6. We recall that in that case we parametrized the coordinates of two of the control points with respect to a parameter $\boldsymbol{\mu} = \mu \in \mathcal{D} = [0.5, 1]$. Moreover we recall that the NURBS weights can assume any fixed value and in this case we chose them to be all equal to one. In Fig. 3.3 we have a representation of both the original domain we are considering and of the reference domain we build in order to trace back the problem on it. For the solution of the problem we use NURBS basis functions of degree 2 (in the following we refer to this case as to P2) meaning that we realize a p-refinement such that $p = q = 2$ for both the original Ω_o and reference domain Ω, and realize a same h-refinement for the two domains in order to have a 25×60 grid yielding $\mathcal{N} = 1500$ degrees of freedom (considering only inner nodes).

We start by presenting the results for EIM, used to approximate the parameter dependent terms in (3.13) by means of a series of affine terms. To perform EIM we choose a tolerance $tol_{EIM} = 10^{-5}$ and a train set of 50 parameter values. We obtain $M^{\nu^q} = 4$ for $1 \leq q \leq Q$, $Q = 4$, and $M^f = 2$, and so offline we will finally assemble $Q_a = Q \times M^{\nu^q} = 16$ matrices of size $\mathcal{N} \times \mathcal{N}$ and $Q_f = 2$ right-hand side vectors of size $\mathcal{N}$ (that is M^f plus Q_a terms, due to lifting operation that, in this case, do not give any contribution being all null since homogeneous Dirichlet conditions are imposed). Just to give an idea of which are the *magic points* selected by EIM for the terms ν_q, for $1 \leq q \leq 4$ and f, we report in Fig. 3.4 their location on the original configuration. We observe that the points selected to approximate ν_2 and ν_3 are the same, since $\boldsymbol{\nu}$ is a symmetric tensor.
We define the average relative error between the IGA solution $\boldsymbol{u}_o(\boldsymbol{\mu})$ (computed on the original domain) and the solution $\boldsymbol{u}_M(\boldsymbol{\mu})$ of the system (3.26) approximated with EIM, defined as

$$\epsilon_{ave,rel}^{IGA,EIM} = \text{average}_{\boldsymbol{\mu} \in \Xi_{test}} \frac{\|\boldsymbol{u}_o(\boldsymbol{\mu}) - \boldsymbol{u}_M(\boldsymbol{\mu})\|_V}{\|\boldsymbol{u}_o(\boldsymbol{\mu})\|_V}. \tag{3.31}$$

This error, computed over a set Ξ_{test} of 20 samples, is of the order of 10^{-6}. This result confirms that the EIM procedure provides a good approximation of the original nonaffine system. In Fig. 3.5 we report the IGA solutions for three different parameters $\boldsymbol{\mu}$ and the correspondent EIM approximations.

We now present the numerical results obtained through the RB approach by considering first the ones obtained by applying the POD (Section 2.4.3), then the ones obtained by means of the greedy approach (Section 2.4.2). We refer to these two different approaches as to the POD-RB and the greedy-RB methods, respectively.

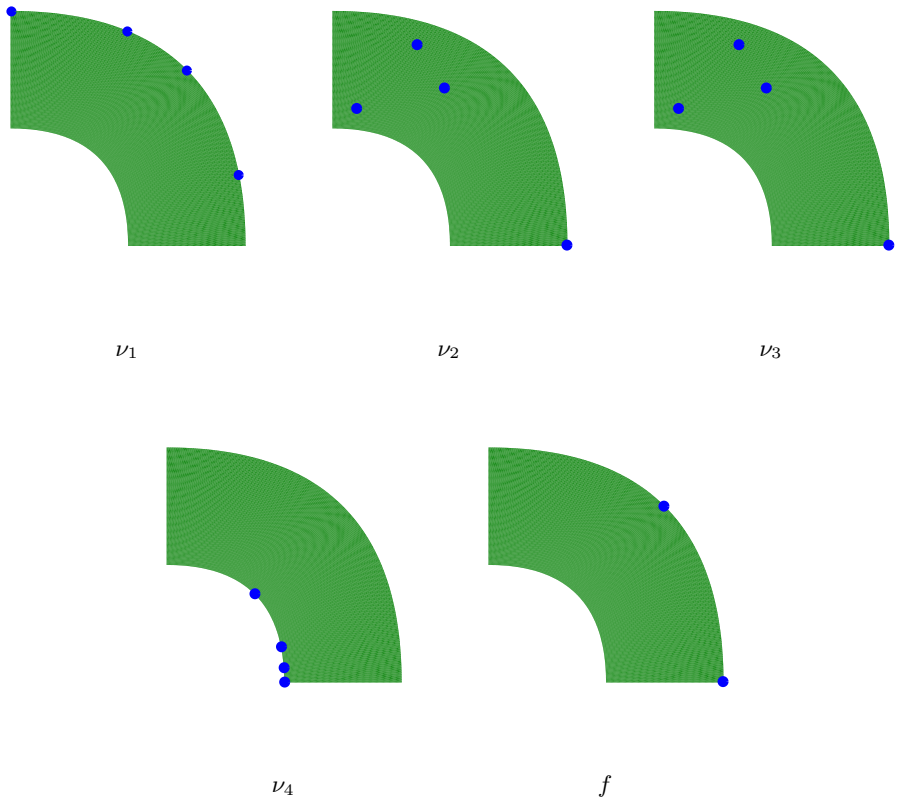


Figure 3.4: Images of the *magic points* selected by EIM to approximate the terms ν_1, ν_2, ν_3, ν_4, ad f, respectively.

Concerning the POD-RB method, we recall that during the offline stage we compute a large set of snapshots and then we transform them into POD modes and the first of these are the ones that retain most of the energy present in the original variables. Through this procedure, we choose the dimensionality of the reduced basis N and during the online stage we solve the reduced system, thanks to a Galerkin projection. For the problem at hand we took a set of 150 snapshots and applied the POD algorithm by considering a tolerance of 10^{-6}. In Fig. 3.6 we report the resulting singular values and conclude that the first 8 POD modes retain most of the energy of the system. Therefore, during the online evaluation, a reduced system of dimension $N_{max} = 8$ has to be solved. Then, in order to compare the results, we repeat the procedure for the cases P3 and P4. To do that we consider again the original and reference domain in Fig. 3.3 and realize on both of them a p-refinement such that $p = q = 3$ and $p = q = 4$, and them perform the same h-refinement to obtain a 25×60 grid. In this way we increase the number of quadrature nodes on which EIM is performed. The result is that EIM returns the same number of affine terms of the case P2 ($M^{\nu^q} = 4$ for $1 \leq q \leq Q$, $Q = 4$, and $M^f = 2$). This results can be justified by the fact that the chosen grid (25×60) is already fine enough in the P2 case and by increasing the polynomial degree, the quadrature nodes are so dense and close among each other that the selection of the indexes and basis realized by the greedy procedure performs more or less in the same way. We then applied the

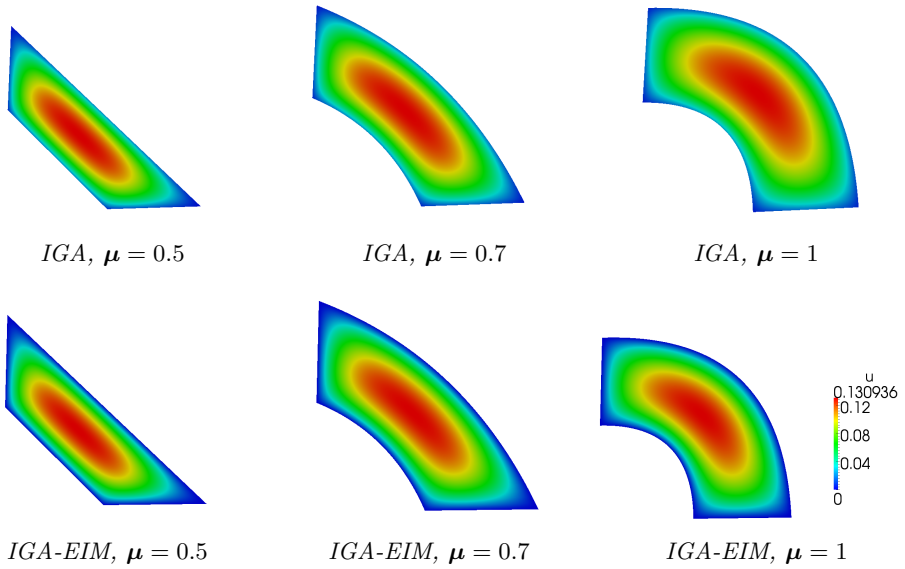


Figure 3.5: IGA solutions $u^{\mathcal{N}}$ (top) and corresponding IGA-EIM solutions u_M (bottom) for the parameters $\boldsymbol{\mu} = 0.5$, $\boldsymbol{\mu} = 0.7$, and $\boldsymbol{\mu} = 1$.

POD algorithm, set with the same parameters (150 precomputed snapshots and a tolerance of 10^{-6}), to the cases P3 and P4. In Fig. 3.7 we report and compare the resulting singular values. We observe that the singular values do not change by varying the polynomial degree. As a matter of fact, the POD procedure works on the operators discretized in the parameters space and by increasing the polynomial orders the geometry and the parametrization are preserved (Section 1.3.2) and, as already said, the quadrature nodes are enough close to not produce a different result. As consequence, since we use the same parameters set Ξ_{train} for the three cases, when we assemble the matrices having as columns the computed snapshots these are strictly correlated among each other and the SVD gives as output the same singular values.

We now aim to evaluate the error between the solution $\boldsymbol{u}_o(\boldsymbol{\mu})$ of the IGA problem on the original domain, and the solution $\boldsymbol{u}^{\mathcal{N}}_{N,M}(\boldsymbol{\mu})$ of problem (3.28), i.e. of the problem approximated by means of EIM and then reduced by means of POD (EIM-POD-RB problem). We consider the case of polynomials degree 2. In Fig. 3.8 we show $\epsilon^{IGA,EIM-POD-RB}_{ave,rel}$ defined as

$$\epsilon^{IGA,EIM-POD-RB}_{ave,rel} = \text{average}_{\boldsymbol{\mu}\in\Xi_{test}} \frac{\|\boldsymbol{u}_o(\boldsymbol{\mu}) - \boldsymbol{\mathcal{Z}}\boldsymbol{u}^{POD}_{N,M}(\boldsymbol{\mu})\|_V}{\|\boldsymbol{u}_o(\boldsymbol{\mu})\|_V}, \tag{3.32}$$

that is the average relative error between $\boldsymbol{u}_o(\boldsymbol{\mu})$ and $\boldsymbol{u}^{\mathcal{N}}_{N,M}(\boldsymbol{\mu})$, computed on a set of 20 parameters samples. The error is shown as a function of the number of basis function N and by considering an increasing number of affine terms Q_a, while Q_f is kept fixed, being $Q_f = 2$. We observe that, for fixed values of Q_a the error initially decreases with increasing N, and then becomes constant for N large enough. Moreover, increasing Q_a tends to reduce the error.

We now consider the RB-greedy strategy. We refer to the resulting problem as the EIM-greedy-RB problem. We consider a set of 150 parameter samples Ξ_{train} and a tolerance

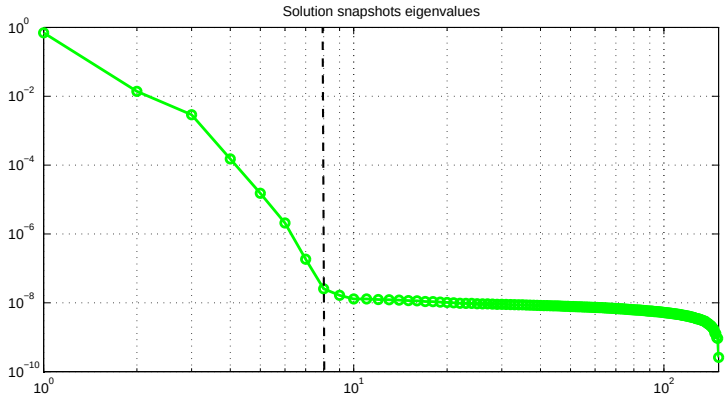


Figure 3.6: Singular values of the snapshots matrix for polynomial degree 2. The plot shows that the first $N_{max} = 8$ POD modes retain most of the energy of the system.

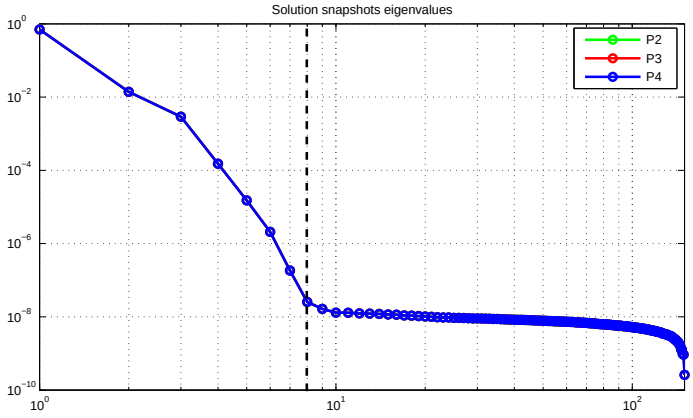


Figure 3.7: Singular values of the snapshots matrix for polynomial degrees 2, 3, 4. The singular values do not change by varying the polynomial degree.

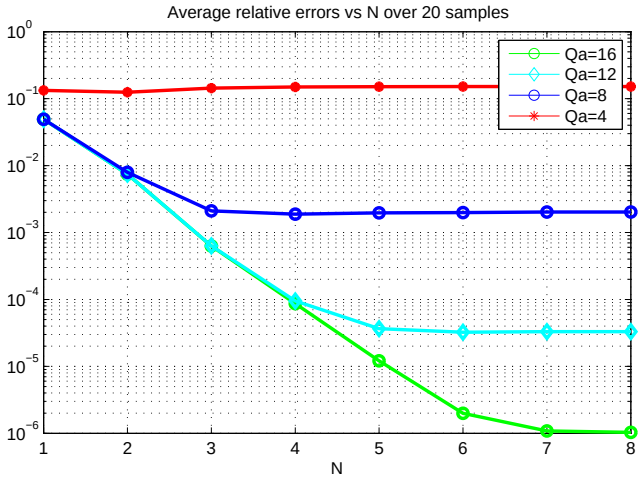


Figure 3.8: Average relative errors $\epsilon_{ave,rel}^{IGA,EIM-POD-RB}$ defined in (3.32), between IGA and EIM-POD-RB solutions vs N for different values of Q_a, computed on a 20 samples parameter vector Ξ_{test}.

of 10^{-6}. The algorithm selects $N_{max} = 9$ basis functions. To test the convergence of the approximated solutions, we take the same set Ξ_{test} of 20 samples already used in the POD case. In Fig. 3.9 we report the error $\epsilon_{ave,rel}^{IGA,EIM-greedy-RB}$ defined as

$$\epsilon_{ave,rel}^{IGA,EIM-greedy-RB} = \text{average}_{\boldsymbol{\mu}\in\Xi_{test}} \frac{\|\boldsymbol{u}_o(\boldsymbol{\mu}) - \boldsymbol{\mathcal{Z}}\boldsymbol{u}_{N,M}^{greedy}(\boldsymbol{\mu})\|_V}{\|\boldsymbol{u}_o(\boldsymbol{\mu})\|_V}, \tag{3.33}$$

that is the average relative error between $\boldsymbol{u}_o(\boldsymbol{\mu})$ and $\boldsymbol{u}_{N,M}^{\mathcal{N}}(\boldsymbol{\mu})$, where $\boldsymbol{u}_{N,M}^{\mathcal{N}}(\boldsymbol{\mu})$ is the solution of problem (3.28), approximated by means of EIM, and reduced by means of greedy (EIM-greedy-RB problem). The error is shown as a function of the number of basis functions N and by considering an increasing number of affine terms Q_a, while Q_f is kept fixed, being $Q_f = 2$. We observe that, for fixed values of Q_a the error initially decreases with increasing N, and then becomes constant for N large enough. Moreover, increasing Q_a the error decreases.

In Fig. 3.10 we compare the error between IGA and the reduced problems obtained through either the POD procedure or the greedy procedure. We do not notice any relevant differences in their accuracy and rapidity in the prediction of the resulting high-fidelity IGA solution in the Online stage. Moreover, the two methods lead to a large computational saving for each online evaluation, which is also similar ($N_{max} = 8$ when using POD and $N_{max} = 9$ when using greedy). These are relatively small dimensions compared to the original high-fidelity problem with $\mathcal{N} = 1500$ degrees of freedom for the case P2, $\mathcal{N} = 1584$ for P3, and $\mathcal{N} = 1674$ for P3. This evident dimensional reduction, allows to reduce the times requested to assemble and solve the IGA system for each parametric evaluation. In Table 3.2 we report the computational times requested to assemble and solve the IGA problem on the original domain (3.4), and the ones requested to assemble and solve the EIM-RB problem (3.28) online, for a single parameter evaluation. The times refer to the 25×60 grid for the cases P2, P3, and P4. In particular in this table we just want to highlight the online computational advantage allowed by the EIM-RB procedure, but do not refer to the times requested offline to assemble and store the involved structures. In Section 3.8.2 we will compare the POD and greedy performances for a specific problem. For this reason, we now report the times just for the

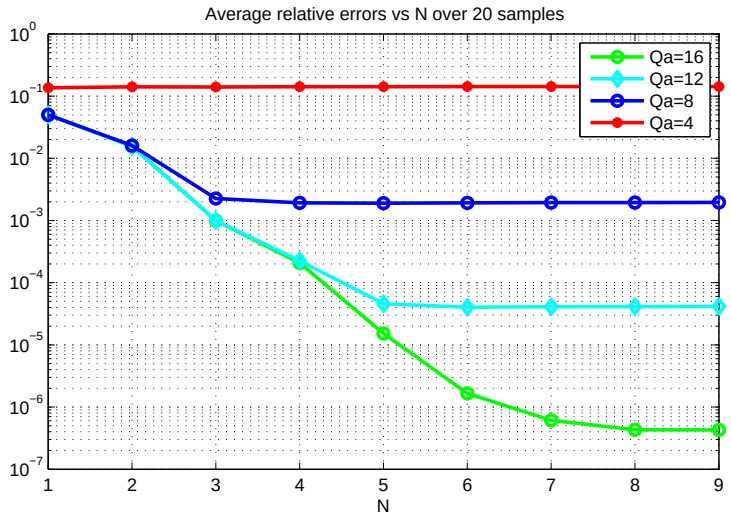


Figure 3.9: Average relative errors $\epsilon_{ave,rel}^{IGA,EIM-greedy-RB}$ defined in (4.20), between IGA and EIM-greedy-RB solutions vs N for different values of Q_a, computed on a 20 samples parameter vector Ξ_{test}.

EIM-POD-RB problem for $N = N_{max} = 8$, but the ones relative to the online stage of the EIM-greedy-RB problem would be the same taking $N = 8$ in that case too. We notice that, in all the cases, we have a computational saving and that this is larger when increasing the polynomial degree, that is when increasing the number of degrees of freedom. As a matter of fact, since EIM selects the same number of points, basis, and thus of affine terms for the three cases, and moreover we are assuming as reduced dimension $N = 8$ for all the cases, the time requested online to evaluate the terms $\boldsymbol{\Theta}(\boldsymbol{\mu})$ is almost the same (it is slightly higher for higher degrees because of the higher time requested to modify the geometry as shown in table 1.3). On the other hand, the time requested to assemble and solve the full systems is obviously higher when increasing the polynomial degree since the number of degrees of freedoms increases as well. For these reasons, we finally conclude that we have the higher computational saving for the case P4.

3.8.2 Two parameters case

In this section we present the numerical results obtained for the resolution of the same problem previously studied, but considering that in this case the control point B_{11} is parametrized with respect to a parameter μ_1 and the control point B_{22} with respect to another parameter μ_2. The vector parameter $\boldsymbol{\mu} = [\mu_1, \mu_2]$ can assume values in the parameter space $\mathcal{D} = [0.5, 1] \times [1, 1.5]$. To solve the problem we still consider $f_o(\boldsymbol{\mu}) = 1$, homogeneous Dirichlet boundary conditions, NURBS basis functions of degree 2, and carry out a h-refinement of the domain in order to obtain a 25×60 grid.

We start by presenting the results for EIM. We choose a tolerance $tol_{EIM} = 10^{-6}$ and a train set of 600 parameter values. We obtain $M^{\nu^1} = 17$, $M^{\nu^2} = 15$, $M^{\nu^3} = 15$, $M^{\nu^4} = 14$, and $M^f = 3$, and so offline we will finally assemble $Q_a = 61$ matrices of size $\mathcal{N} \times \mathcal{N}$ and $Q_f = 3$ right-hand side vectors of size $\mathcal{N}$. In Table 3.3 we summarize these data, while in Fig.

Table 3.2: *Computational times t_{IGA}, requested to assemble and solve the IGA problem on the original domain* (3.4)*, and t_{EIM-RB}, requested to assemble and solve the EIM-RB problem* (3.28)*, for each parameter evaluation.*

Degree	P2	P3	P4
Number of parameters	1	1	1
EIM samples number	50	50	50
EIM tolerance tol_{eim}	10^{-5}	10^{-5}	10^{-5}
Affine matrix components $\boldsymbol{A}^{qm}$	16	16	16
Affine rhs components $\boldsymbol{L}^{m}$	2	2	2
$\mathcal{N}$	1500	1586	1674
N	8	8	8
t_{IGA}	3.681 s	6.620 s	12.154 s
t_{EIM-RB}	0.316 s	0.476 s	0.685 s
t_{IGA}/t_{EIM-RB}	11.65	13.91	17.74

Table 3.3: Empirical interpolation method data, 2 parameters case.

EIM samples	600
EIM tolerance tol_{eim}	10^{-6}
Affine matrix components $\boldsymbol{A}^{qm}$	$Q_a = 61$
Affine rhs components $\boldsymbol{L}^{qm}$	$Q_f = 3$

3.11 we report the images of the *magic points* selected by EIM for the terms ν_q, for $1 \leq q \leq 4$ and f. Then, we compute the average error between the solutions of the IGA system and the one approximated by EIM on a set of 20 parameters samples, as previously done in (3.31). This error is of the order of 10^{-6}, accordingly to the chosen tolerance $\text{tol}_{EIM} = 10^{-5}$. In Fig. 3.12 we report, for three different parameters vectors, the numerical solutions of IGA and the corresponding solutions of (3.26).

We now present the numerical results obtained through the RB approach by considering first the results obtained by applying the POD, then the ones obtained by means of the greedy algorithm.

We took a set of 600 snapshots and applied the POD algorithm for the case of polynomial degree 2. In Fig. 3.13 we report the resulting singular values and conclude that the first 27 POD modes retain most of the energy of the system. So, online a reduced system of dimension $N_{max} = 27$ has to be solved.

Now, we evaluate the average relative error $\epsilon_{ave,rel}^{IGA,EIM-POD-RB}$ between the solution of the

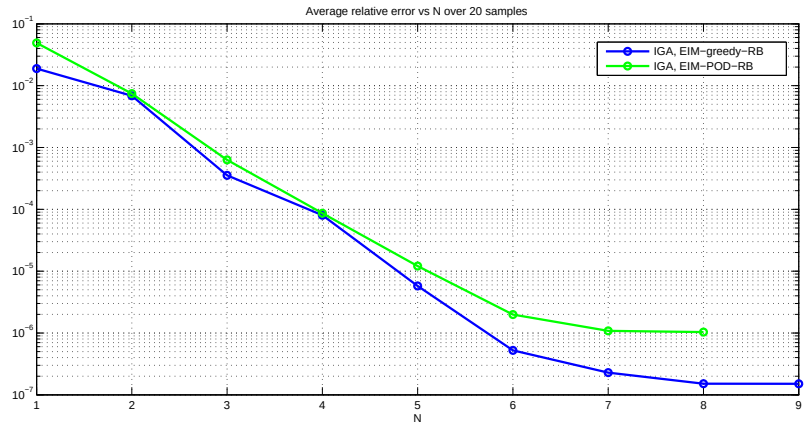


Figure 3.10: Average relative error $\epsilon_{ave,rel}^{IGA,EIM-POD-RB}$ (3.32) and $\epsilon_{ave,rel}^{IGA,EIM-greedy-RB}$ (4.20) vs N for $Qa = 16$.

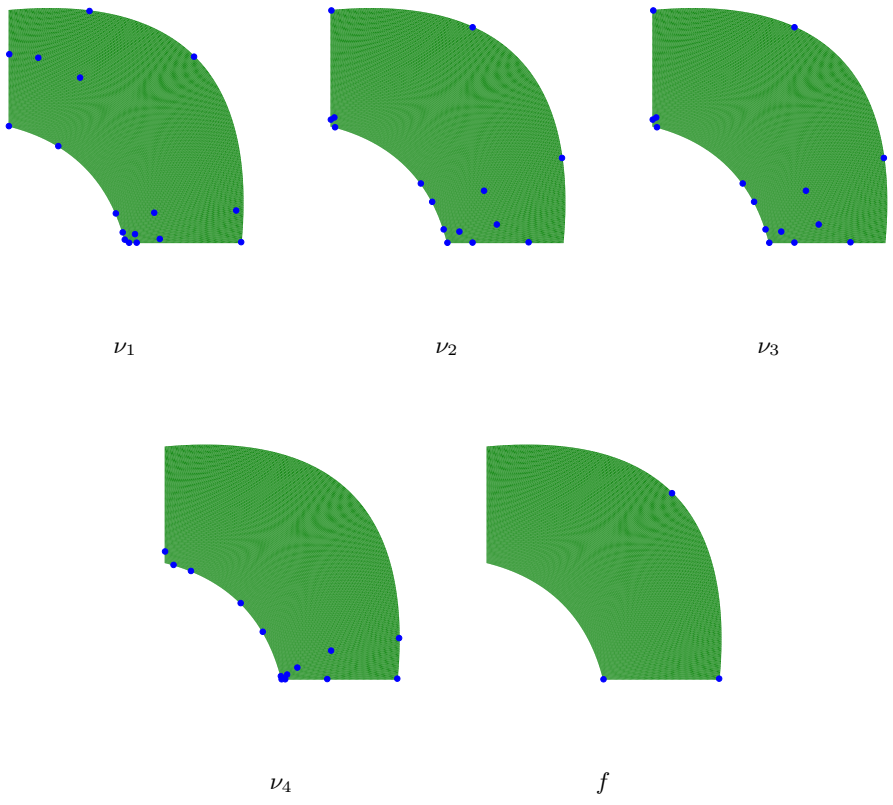
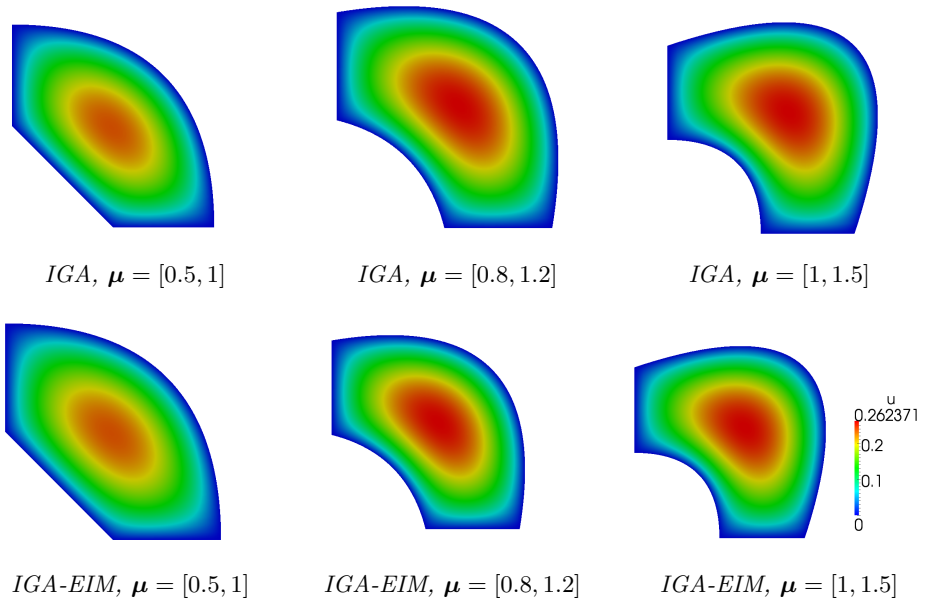


Figure 3.11: Images of the *magic points* selected by EIM to approximate the terms ν_1, ν_2, ν_3, ν_4, ad f, respectively.



IGA, $\boldsymbol{\mu} = [0.5, 1]$ *IGA*, $\boldsymbol{\mu} = [0.8, 1.2]$ *IGA*, $\boldsymbol{\mu} = [1, 1.5]$

IGA-EIM, $\boldsymbol{\mu} = [0.5, 1]$ *IGA-EIM*, $\boldsymbol{\mu} = [0.8, 1.2]$ *IGA-EIM*, $\boldsymbol{\mu} = [1, 1.5]$

Figure 3.12: IGA solutions $u^{\mathcal{N}}$ (top) and corresponding IGA-EIM solutions u_M (bottom) for the parameters vectors $\boldsymbol{\mu} = [0.5, 1]$, $\boldsymbol{\mu} = [0.8, 1.2]$, and $\boldsymbol{\mu} = [1, 1.5]$.

IGA and the solution of the EIM-POD-RB problem, already defined in (3.32). In Fig. 3.14 we plot this error computed on a set of 400 parameters samples. As previously done, we report the error as a function of the number of basis N and by considering an increasing number of affine terms Q_a, while Q_f is kept fixed. For fixed values of Q_a the error initially decreases with increasing N, and then becomes constant for N large enough. Moreover, increasing Q_a the error decreases.

We now go back to the original system approximated by EIM and apply the RB strategy but performing a greedy approach. We consider a set of 600 parameter samples Ξ_{train} and a tolerance of 10^{-6}. Under these conditions, the greedy algorithm selects 29 basis functions. In Fig. 3.16 we report the error $\epsilon_{ave,rel}^{IGA,EIM-greedy-RB}$ defined in (4.20) computed on the same Ξ_{test} of 400 samples used in the POD case. The error is shown as a function of the number of basis N and by considering an increasing number of affine terms Q_a, while Q_f is kept fixed. Also in this case, we observe that, for fixed values of Q_a the error initially decreases with increasing N, and then becomes constant for N large enough, and increasing Q_a tends to reduce the error.

In Fig. 3.17 we compare the error between IGA and the reduced problems obtained through the POD procedure or the greedy procedure. In Table 3.4 we report the computational times requested to assemble and solve the IGA problem on the original domain (3.4), and the ones requested to assemble and solve the EIM-RB problem (3.28), for a single parameter evaluation. We consider as reduced dimension $N = N_{max} = 27$. Comparing these data with the ones in Table 3.2, we notice that the time requested to assemble the IGA system for the P2 case is almost the same, since the grids contain the same number of elements in the two cases. On the other hand, the time requested to assemble the EIM-RB system is higher compared to the one of Table 3.2, since in this case we are considering a higher dimension for

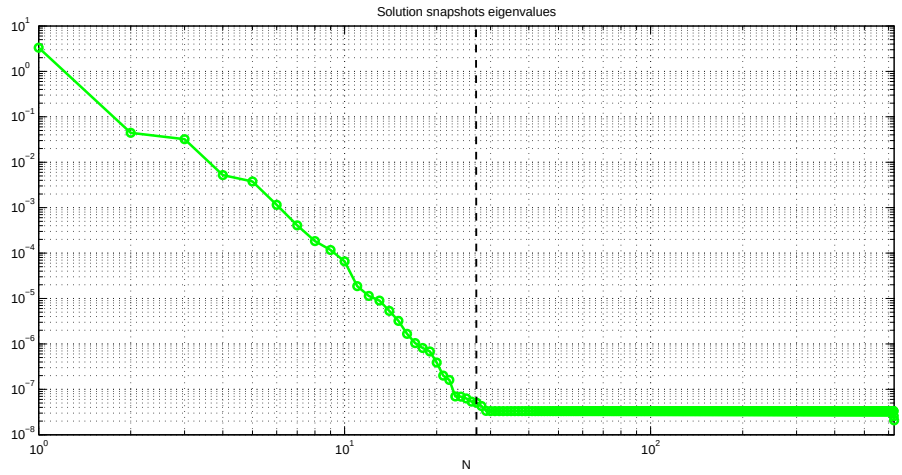


Figure 3.13: Singular values of the snapshots matrix for polynomial degree P2. The plot shows that the first $N_{max} = 27$ POD modes retain most of the energy of the system.

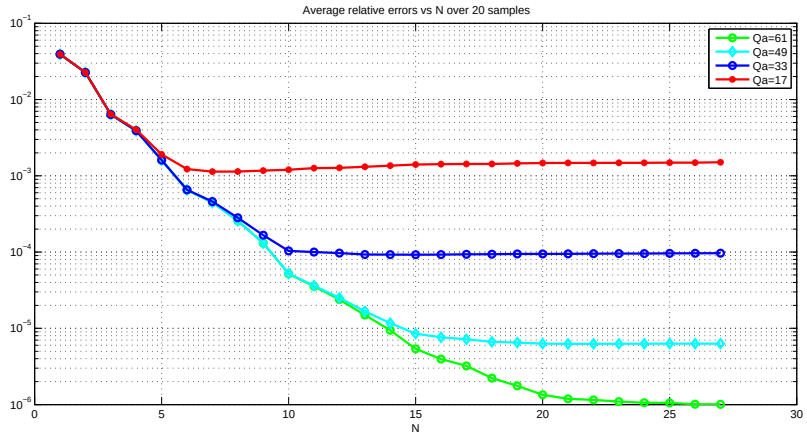


Figure 3.14: Average relative errors $\epsilon_{ave,rel}^{IGA,EIM-POD-RB}$ defined in (3.32), between IGA and EIM-POD-RB solutions vs N for different values of Q_a, computed on a 400 samples parameter vector Ξ_{test}.

the reduced spaces and also because EIM selected a higher number of affine terms. Moreover, from this table we can compare the POD and greedy sampling strategies performances. We notice that both the methods yield a computational saving of about 7 times for each online evaluation, but on the other hand, we notice that the two methods behave differently in the offline stage since the time requested by the greedy is about twice the one requested by the POD algorithm. This difference can be explained by the fact that, as anticipated at the end of Section 2.5, the computational cost for the evaluation of the norm of the residual in the greedy algorithm, can increase dramatically for a high (let us say 50 or more [MSH15]) number of affine terms. So, since in this case we are dealing with 61 affine terms for the stiffness matrix and 3 affine terms for the right-hand side, the use of the greedy-RB strategy is unfavorable for the problem we are studying, but as we will see in Chapter 4, for more complex problems with a lower number of affine terms, the greedy-RB strategy is preferable.

Table 3.4: *Computational times t_{IGA}, requested to assemble and solve the IGA problem on the original domain* (3.4)*, and $t_{EIM-POD-RB}$, $t_{EIM-greedy-RB}$ requested to assemble and solve the EIM-RB problem* (3.28)*, for each parameter evaluation.*

Degree	P2
Number of parameters	2
Affine matrix components $\boldsymbol{A}^{qm}$	61
Affine rhs components $\boldsymbol{L}^m$	61
Sample train (for POD and greedy)	600
$\mathcal{N}$	1500
POD-RB N	27
greedy-RB N	27
t_{IGA}	3.4286 s
POD construction time (offline)	24.27 min
$t_{EIM-POD-RB}$ (online)	0.4868 s
Greedy construction time (offline)	48.49 min
$t_{EIM-greedy-RB}$ (online)	0.4789 s
$t_{IGA}/t_{EIM-POD-RB}$	7.0431
$t_{IGA}/t_{EIM-greedy-RB}$	7.1593

3.9 Conclusions

In this chapter we have considered a class of differential problems defined on surfaces parametrized with respect to the control points coordinates, but not to the NURBS weights. We have considered parametrizations that lead to a non affine parametric dependence in the integrals of the variational form that describes the differential problem and this requires suitable

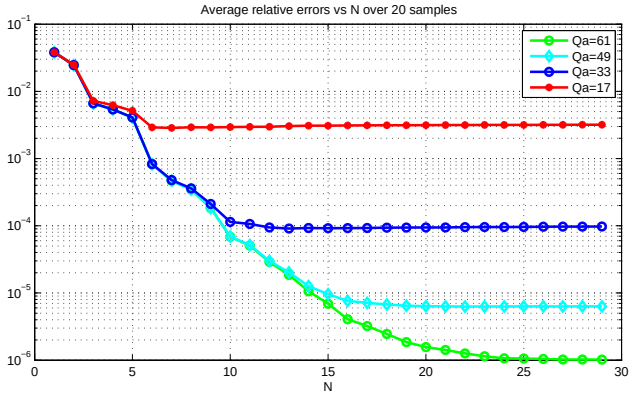


Figure 3.15: Average relative errors $\epsilon_{ave,rel}^{IGA,EIM-greedy-RB}$ defined in (4.20), between IGA and EIM-greedy-RB solutions vs N for different values of Q_a, computed on a 20 samples parameter vector Ξ_{test}.

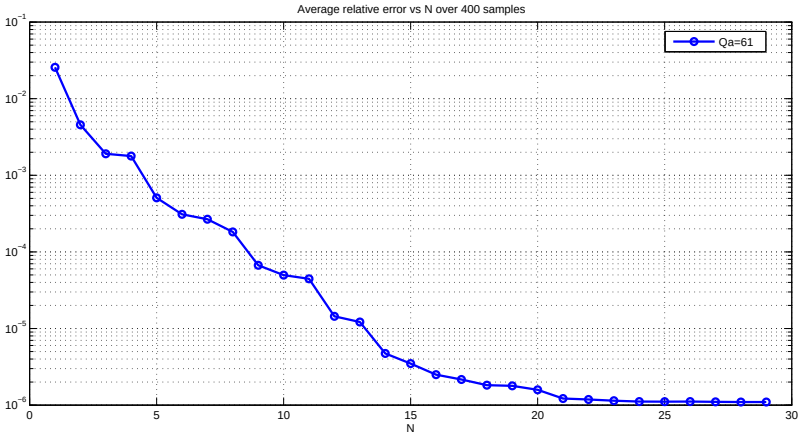


Figure 3.16: Average relative error $\epsilon_{ave,rel}^{IGA,EIM-greedy-RB}$, between IGA and EIM-greedy-RB vs N.

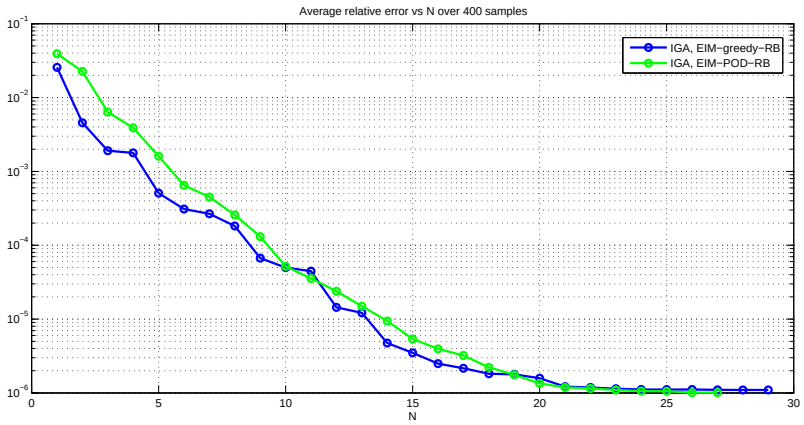


Figure 3.17: Average relative error $\epsilon_{ave,rel}^{IGA,EIM-POD-RB}$ (3.32) and $\epsilon_{ave,rel}^{IGA,EIM-greedy-RB}$ (4.20) vs N for $Qa = 61$.

interpolatory techniques in order to apply the RB method. In particular, we have applied EIM in order to restore the affine dependence on the parameters. Then, we have considered as test applied case the Poisson problem defined on a 2D surface parametrized first with respect to one parameter and then with respect to two parameters. For both the cases we have approximated the nonaffine terms with EIM and then built two different reduced order models by means of the POD and the greedy algorithm. The numerical tests have shown a great accuracy between the reduced models and the full-order one. Moreover, although the construction of the reduced spaces using POD or greedy is different, we did not remark relevant differences in their accuracy and rapidity in the prediction of the results of the problem at hand during the Online stage.

In the next chapter we will see that when the NURBS weights are parametrized, independently on the fact that the control points coordinates assume fixed values or are parameter dependent, EIM is not usable anymore since we are unable to define a reference configuration through the same NURBS functions used to build the original configuration, and to separate, in the integrals of the problem formulation, the $\boldsymbol{\mu}$-dependent terms, from the $\boldsymbol{\mu}$-independent terms. For this reason, we will appeal to a new technique, the Matrix Discrete Empirical Interpolation Method (MDEIM) [NMA15], that represents a variant of EIM. We remark anyway that MDEIM is not affected by the kind of parametrization considered and so could be used also when the parametrization affects only the control points coordinates. In this case, we will see that compared to EIM, it allows to reduce the computational costs associated with the assembling of the IGA structures.

Chapter 4

Reduced Basis for IGA: Parametrized NURBS weights

In this chapter, similarly to Chapter 3, we consider a class of problems described by parametrized PDEs defined on parametrized geometries, in order to solve the PDE for different choices of parametrization of the computational domain. We consider domains built by means of NURBS basis functions and characterized by geometrical parameters $\boldsymbol{\mu}$. In particular, we start by considering the case in which only the NURBS weights are parameter dependent, but not the control points coordinates which are fixed. Then, we generalize the discussion to the case in which both the control points coordinates and the NURBS weights are parametrized. In order to efficiently solve the problem for each new parameter $\boldsymbol{\mu}$, that is for each new domain configuration, we adopt the RB method and for the spatial discretization of the problem, we consider IGA.

As in Chapter 3, we consider parametrizations that induce nonaffine geometric transformations and in turns a nonaffine dependence on the parameters of the terms involved in the problem formulation. We recall that, in order to maximize the computational saving allowed by the RB method, the problem should satisfy the affinity assumption. This is why in Chapter 3 we appealed to EIM in order to restore the affinity of the problem. However, in this chapter we see that when dealing with parametrized NURBS weights, EIM is not usable anymore since we are unable to define a suitable reference configuration and to identify, in the integrals of the problem formulation, a $\boldsymbol{\mu}$-dependent contribution (actually $\boldsymbol{\mu}$-dependent tensorial coefficients to be approximated with EIM), and a $\boldsymbol{\mu}$-independent one to be computed only once offline. For this reason, we resort to a new technique, the Matrix Discrete Empirical Interpolation Method (MDEIM) [CS10, Ton, NMA15] that represents a matrix version of the discrete variant of EIM. In order to explicitely show how MDEIM works, we choose as test problem the Laplace-Beltrami problem on a surface in 3D. We notice that, although we consider this geometry, the procedure can be used for every problem defined on a geometry parametrized with respect to the NURBS weights, and in particular also for 2D surfaces.

In Section 4.1 we introduce a parametrized problem on a manifold described by NURBS. In Section 4.2 we give a reference formulation of the test problem at hand and build a reference domain Ω by following the same approach described in Section 3.2. However, we show that this approach is not feasible, since building the reference domain by means of the same NURBS basis functions used for the original one, induces a parameter dependence on Ω, that, in this way, cannot represent a reference configuration. Moreover, we show that since

the parametrization affects all the NURBS weights, if we use EIM to restore the affinity of the integral formulation of the problem, the online evaluation stage would depend on the full dimension of the problem, which is precisely what should be avoided to build efficient RB. For these reasons, we exploit an alternative methodology. In Section 4.3 we introduce the Discrete Empirical Interpolation Method, DEIM, a discrete variant of EIM. In Section 4.4 we describe a matrix version of DEIM, MDEIM, and explain how it allows to obtain an affine approximation of the problem at hand. Then, in Section 4.5 we reduce the dimension of this problem through the RB procedure. In Section 4.6 we report the numerical results for the problem at hand. In Section 4.7 we solve the problem introduced in Chapter 3 but adopting the MDEIM approach and compare its computational performances with the EIM ones.

4.1 Problem formulation on a parametrized NURBS manifold

We consider problem (1.32) and introduce a parametrized version of it by assuming that the geometry of the problem of interest is parameter dependent. We assume Ω_o to be the parametric geometry built in Example 1.7 in which the parameter vector $\boldsymbol{\mu} = \mu \in \mathbb{R}$ regards only two of the NURBS weights, in particular the ones associated to the control points $\boldsymbol{B}_{12}$ and $\boldsymbol{B}_{22}$. We refer to this parametrized geometry as to $\Omega_o(\boldsymbol{\mu})$. This geometry $\Omega_o(\boldsymbol{\mu})$ is represented by the NURBS space $\mathcal{B}_{ho}$ defined as

$$\mathcal{B}_{ho}(\boldsymbol{\mu}) = \text{span}\{\mathcal{R}_{oj}(\boldsymbol{\mu})\}_{j=1}^{n} = \text{span}\{R_{oj}(\boldsymbol{\xi};\boldsymbol{\mu}) \circ \boldsymbol{F}^{-1}\}_{j=1}^{n}. \tag{4.1}$$

Here, $R_{oj}(\boldsymbol{\xi};\boldsymbol{\mu})$ represent the NURBS basis functions defined on the parametric domain $\hat{\Omega}$. Similarly to the definition given in (1.17), these NURBS basis functions R_{oj} can be expressed as

$$R_{oj}(\boldsymbol{\xi};\boldsymbol{\mu}) := \frac{N_{oj}(\boldsymbol{\xi})w_j}{\sum_{\hat{j}=1}^{n} w_{\hat{j}} N_{o\hat{j}}(\boldsymbol{\xi})}, \quad j = 1, ..., n \tag{4.2}$$

where N_{oj} are the bivariate B-splines basis functions, defined in (1.6) on $\Omega_o(\boldsymbol{\mu})$ and, according to the chosen parametrization, the weights w_j associated to the control points $\boldsymbol{B}_{12}$ and $\boldsymbol{B}_{22}$ are set to be equal to $\boldsymbol{\mu}$. The mapping $\boldsymbol{F}$, already introduced in (1.18), can be rewritten by highlighting the parameter dependence as follows:

$$\boldsymbol{F} : \hat{\Omega} \times \mathcal{D} \to \Omega_o(\boldsymbol{\mu}) \subset \mathbb{R}^d, \quad (\boldsymbol{\xi}, \boldsymbol{\mu}) \to \boldsymbol{F}(\boldsymbol{\xi};\boldsymbol{\mu}) = \sum_{j=1}^{n} R_{oj}(\boldsymbol{\xi};\boldsymbol{\mu})\boldsymbol{B}_j. \tag{4.3}$$

We notice that, while in the isogeometric map (3.2) the parameter dependence regards the control points, in the map (4.3), since some of the NURBS weights are parametrized with respect to $\boldsymbol{\mu}$, the parameter dependence regards the NURBS basis functions. This is the main difference between the problem studied in the previous chapter and the one we are now considering.

The parametrized version of problem (1.32) then reads: given $\boldsymbol{\mu} \in \mathcal{D}$ find $u_o(\boldsymbol{\mu}) : \Omega_o(\boldsymbol{\mu}) \to \mathbb{R}$ such that

$$-\delta\Delta_{\Omega_o} u_o(\boldsymbol{\mu}) = f_o(\boldsymbol{\mu}) \qquad \text{in } \Omega_o(\boldsymbol{\mu}) \tag{4.4a}$$

$$u_o(\boldsymbol{\mu}) = 0 \qquad \text{on } \partial\Omega_o(\boldsymbol{\mu}) \tag{4.4b}$$

We introduce the weak formulation of the problem, which will be referred to as *original problem*: given $\boldsymbol{\mu} \in \mathcal{D}$ find $u_o(\boldsymbol{\mu}) \in V_o$ such that

$$a_o(u_o(\boldsymbol{\mu}), v; \boldsymbol{\mu}) = L_o(v; \boldsymbol{\mu}), \quad \forall v \in V_o, \tag{4.5}$$

where $V_o = V_o(\boldsymbol{\mu}) = H_0^1(\Omega_o)$ and $\Omega_o = \Omega_o(\boldsymbol{\mu})$. The associated bilinear form $a_o : V_o \times V_o \to \mathbb{R}$ and the linear functional $L_o : V_o \to \mathbb{R}$ can be expressed as

$$a_o(w, v; \boldsymbol{\mu}) = \int_{\Omega_o} (\nabla_{\Omega_o} w)^T \boldsymbol{\nu}_o(\boldsymbol{x}; \boldsymbol{\mu}) \nabla_{\Omega_o} v d\Omega_o \tag{4.6a}$$

$$L_o(v; \boldsymbol{\mu}) = \int_{\Omega_o} f_o(\boldsymbol{x}; \boldsymbol{\mu})\ v\ d\Omega_o, \tag{4.6b}$$

where $\boldsymbol{\nu}_o : \mathbb{R}^d \times \mathcal{D} \to \mathbb{R}^{2\times 2}$, $d = 3$, is a parametrized (symmetric positive definite) diffusivity tensor, in this case $\boldsymbol{\nu}_o(\boldsymbol{x}; \boldsymbol{\mu}) = \delta \boldsymbol{I}$, with $\boldsymbol{I} \in \mathbb{R}^{2\times 2}$ the identity matrix, and $f_o : \mathbb{R}^d \times \mathcal{D} \to \mathbb{R}$ are prescribed coefficients.
The adopted parametrization entails a parametric dependence on the whole problem. For what said in Section 2.3, the RB method requires a parameter independent domain in order to compute and combine the IGA solutions that will be used as basis functions of the RB approximation space. For this reason, in Chapter 3 we properly chose a reference ($\boldsymbol{\mu}$-independent) domain Ω, and a parametric transformation $\boldsymbol{T}$ between Ω and $\Omega_o(\boldsymbol{\mu})$, and recast the original problem onto the reference configuration. In this reference problem, the effect of the geometric variations, that is the parameter dependence, was traced back and limited to its parametrized transformation tensors $\boldsymbol{\nu}(\boldsymbol{x}; \boldsymbol{\mu})$ and $f(\boldsymbol{x}; \boldsymbol{\mu})$, as highlighted in (3.13). Unfortunately, these parametrized tensors were non linear functions of both the spatial coordinate $\boldsymbol{x}$ and the vector parameter $\boldsymbol{\mu}$. Thus, we approximated them with affinely parametric dependent terms through the EIM procedure. This is the usual procedure followed when solving a problem in a RB context, and this is why, also for the case treated in this chapter, we try to adopt the same approach.

4.2 Parametrized formulation on a reference domain

In this section we transform problem (4.5) into a new one defined on a reference domain Ω. As already done in Section 3.2, we introduce the map $\boldsymbol{T}$, between Ω and $\Omega_o(\boldsymbol{\mu})$

$$\boldsymbol{T}(\cdot; \boldsymbol{\mu}) : \Omega \to \Omega_o(\boldsymbol{\mu}),$$

such that $\Omega_o(\boldsymbol{\mu}) = \boldsymbol{T}(\Omega; \boldsymbol{\mu})$. Assuming to have already defined a proper reference domain Ω and a transformation $\boldsymbol{T}$, the weak formulation of the reference problem reads: find $u(\boldsymbol{\mu}) \in V$ such that

$$a(u(\boldsymbol{\mu}), v; \boldsymbol{\mu}) = L(v; \boldsymbol{\mu}), \ \ \forall v \in V,$$

where $V = H_0^1(\Omega)$ and the bilinear form and the linear functional can be expressed as

$$a(w, v; \boldsymbol{\mu}) = \int_{\Omega} (\nabla_{\Omega} w)^T \boldsymbol{\nu}(\boldsymbol{x}; \boldsymbol{\mu}) \nabla_{\Omega} v d\Omega \tag{4.7a}$$

$$L(v; \boldsymbol{\mu}) = \int_{\Omega} f(\boldsymbol{x}; \boldsymbol{\mu})\ v\ d\Omega. \tag{4.7b}$$

We now try to identify a suitable reference domain and a parametric transformation $\boldsymbol{T}$. We recall that in Section 3.4, in addition to the geometrical map $\boldsymbol{F}$ between $\hat{\Omega}$ and $\Omega_o(\boldsymbol{\mu})$, and the map $\boldsymbol{T}$ between Ω and $\Omega_o(\boldsymbol{\mu})$, we introduced the map $\boldsymbol{S}$ between $\hat{\Omega}$ and Ω. The main factors that led us to the final choice of the reference domain, shown in Fig. 3.3, where the following ones:

- Ω must be independent on $\boldsymbol{\mu}$;
- $\hat{\mathcal{B}}_{oh}$ (the NURBS space of the original domain traced back into the parameter domain $\hat{\Omega}$) must coincide with $\hat{\mathcal{B}}_h$ (the NURBS space of the reference domain traced back into the parameter domain). Since in the practice both the integration in Ω_o and Ω are traced back into $\hat{\Omega}$ it would not make sense if these space functions were different in the two cases.

The first strategy we adopt to define the reference domain Ω is the one described in Chapter 3. Thus, as we show in Fig. 4.1, we build Ω as a unit square but using the same NURBS basis functions used to build $\Omega_o(\boldsymbol{\mu})$. To this end, we choose the polynomial degrees for the direction x and y as the ones of the domain $\Omega_o(\boldsymbol{\mu})$ in the directions x_o and y_o; moreover, we assign to the control points P_{12} and P_{22} the same weights associated to the corresponding points B_{12} and B_{22} in the original domain. In this way, $\Omega_o(\boldsymbol{\mu})$ and Ω are built by means of the same basis functions and $\hat{\mathcal{B}}_{oh} = \hat{\mathcal{B}}_h$. With this choice, however, the resulting Ω is not $\boldsymbol{\mu}$-independent. As a matter of fact, when we change the parametric weights on the original domain, although if Ω is always a unit square, we are actually changing the weights associated to P_{12} and P_{22}. It means that, since $\Omega = \Omega(\boldsymbol{\mu})$, it does not represent a reference configuration and does not allow to compare and combine the different snapshots and solutions that instead is what RB actually requires to do. It is now clear why the approach adopted in Chapter 3 is not feasible for parametrizations affecting the NURBS weights. In the following, we try different approaches in order to define a reference domain and use EIM.

We first try to fix the weights of the reference domain to 1, also when changing that of the original domain (see Fig. 4.2). In this way Ω is not parameter dependent. However, the problem is that the basis functions are different between Ω_o and Ω, i.e. $\hat{\mathcal{B}}_{oh}$ and $\hat{\mathcal{B}}_h$ are not the same, and as a consequence, the integration operations would not be comparable between the two domains.

Thus, we try another kind of approach. We consider as reference domain the one in Fig. 4.1. As already said, in this way we have that $\hat{\mathcal{B}}_{oh} = \hat{\mathcal{B}}_h$, but the NURBS basis functions generating Ω are parameter dependent. Anyway, this time the idea is not to directly integrate on Ω the NURBS basis functions defining this geometry, but something that is $\boldsymbol{\mu}$-independent. The idea is to make the expression of the NURBS basis functions generating this domain explicit and try to separate the parameter dependence involved in the NURBS weights and in the weighting function, from the B-spline basis functions, that are parameter independent, to finally integrate them. In this way, we aim to bring back the integration of NURBS on Ω to the integration of B-Spline (parameter independent), and then multiply these integrals for

some parameter dependent coefficients. By considering the bilinear form (4.7a) we find

$$\begin{aligned}
a(R_j, R_i; \boldsymbol{\mu}) &= \int_\Omega \nabla R_j \boldsymbol{\nu}(\boldsymbol{x}; \boldsymbol{\mu}) \nabla R_i d\Omega \\
&= \int_\Omega \frac{\partial R_j}{\partial x_1} \nu_{11} \frac{\partial R_i}{\partial x_1} d\Omega + \int_\Omega \frac{\partial R_j}{\partial x_1} \nu_{12} \frac{\partial R_i}{\partial x_2} d\Omega + \int_\Omega \frac{\partial R_j}{\partial x_2} \nu_{21} \frac{\partial R_i}{\partial x_1} d\Omega + \int_\Omega \frac{\partial R_j}{\partial x_2} \nu_{22} \frac{\partial R_i}{\partial x_2} d\Omega \\
&= \boldsymbol{A}_{ij}^{11}(\boldsymbol{\mu}) + \boldsymbol{A}_{ij}^{12}(\boldsymbol{\mu}) + \boldsymbol{A}_{ij}^{21}(\boldsymbol{\mu}) + \boldsymbol{A}_{ij}^{22}(\boldsymbol{\mu}).
\end{aligned}$$

Let us focus on $\boldsymbol{A}_{ij}^{12}(\boldsymbol{x}; \boldsymbol{\mu})$,

$$\begin{aligned}
\boldsymbol{A}_{ij}^{12}(\boldsymbol{x}; \boldsymbol{\mu}) &= \int_\Omega \frac{\partial R_j}{\partial x_1} \nu_{12} \frac{\partial R_i}{\partial x_2} d\Omega \\
&= \int_\Omega \nu_{12} w_j \frac{W(\boldsymbol{\xi}) \frac{dN_j}{d\xi} - W'_{\boldsymbol{\xi}}(\boldsymbol{\xi}) N_j(\xi)}{(W(\boldsymbol{\xi}))^2} w_i \frac{W(\boldsymbol{\xi}) \frac{dN_i}{d\eta} - W'_\eta(\boldsymbol{\xi}) N_i}{(W(\xi))^2},
\end{aligned}$$

where $\boldsymbol{\xi} = (\xi, \eta)$ are the coordinates in the parameter domain, and $W'_\xi(\boldsymbol{\xi})$ and $W'_\eta(\boldsymbol{\xi})$ represent the derivatives of $W(\boldsymbol{\xi})$ respect to ξ and η, respectively. For the sake of simplicity, in the following we write W instead of $W(\boldsymbol{\xi})$:

$$\begin{aligned}
\boldsymbol{A}_{ij}^{12} &= \int_\Omega \nu_{12} \frac{w_j w_i}{W^2} \frac{dN_j}{d\xi} \frac{dN_i}{d\eta} d\Omega - \int_\Omega \nu_{12} \frac{w_j w_i W'_\eta}{W^3} \frac{dN_j}{d\xi} N_i d\Omega + \\
&- \int_\Omega \nu_{12} \frac{w_j w_i W'_\xi}{W^3} N_j \frac{dN_i}{d\eta} d\Omega + \int_\Omega \nu_{12} \frac{w_j w_i W'_\xi W'_\eta}{W^4} N_j N_i d\Omega \\
&= \Theta^1(\boldsymbol{\mu}) \int_\Omega w_j w_i \frac{dN_j}{d\xi} \frac{dN_i}{d\eta} d\Omega + \Theta^2(\boldsymbol{\mu}) \int_\Omega w_j w_i \frac{dN_j}{d\xi} N_i d\Omega + \\
&+ \Theta^3(\boldsymbol{\mu}) \int_\Omega w_j w_i N_j \frac{dN_i}{d\eta} d\Omega + \Theta^4(\boldsymbol{\mu}) \int_\Omega w_j w_i N_j N_i d\Omega.
\end{aligned}$$

The idea in the last step is to restore an affine dependence on the parameter with four $\boldsymbol{\mu}$-dependent coefficients, eventually approximated by EIM,

$$\begin{aligned}
\Theta^1(\boldsymbol{\mu}) &= \nu_{12} \frac{1}{W^2}, \quad \Theta^2(\boldsymbol{\mu}) = \nu_{12} \frac{W'_\eta}{W^3}, \\
\Theta^3(\boldsymbol{\mu}) &= \nu_{12} \frac{W'_\xi}{W^3}, \quad \Theta^4(\boldsymbol{\mu}) = \nu_{12} \frac{W'_\xi W'_\eta}{W^4}
\end{aligned}$$

and four $\boldsymbol{\mu}$-independent integrals to be assembled Offline. The problem is that, since after a refinement all the NURBS weights w_i for $i = 1, ..., \mathcal{N}$, are affected by the parametrization, if we put them out of the integrals to restore the affinity assumption, the online evaluation stage becomes dependent on $\mathcal{N}$ and this is what we are trying to avoid since the beginning.

It is now clear that, when dealing with parametrized NURBS weights, we are not able to define a reference domain and apply EIM on the nonaffine terms present in the integral formulation. Since we would have the same problems if also the control points coordinates are parametrized, in the rest of this chapter we refer to the more general case in which the parametrization affects both the NURBS weights and the control points coordinates. In this case, it is requested to exploit a new kind of procedure. Since our final goal is to provide an affine approximation of matrices and vectors, in the Section 4.3 we appeal to a methodology that represents a variant of EIM, that is the Matrix Discrete Empirical Interpolation Method (MDEIM). While EIM works on nonlinear functions, MDEIM can be applied directly to the (IGA) algebraic structures on the original domain $\Omega_o(\boldsymbol{\mu})$, in our specific case to the matrix form of problem (4.5). As preliminar to MDEIM, in next section we first describe DEIM.

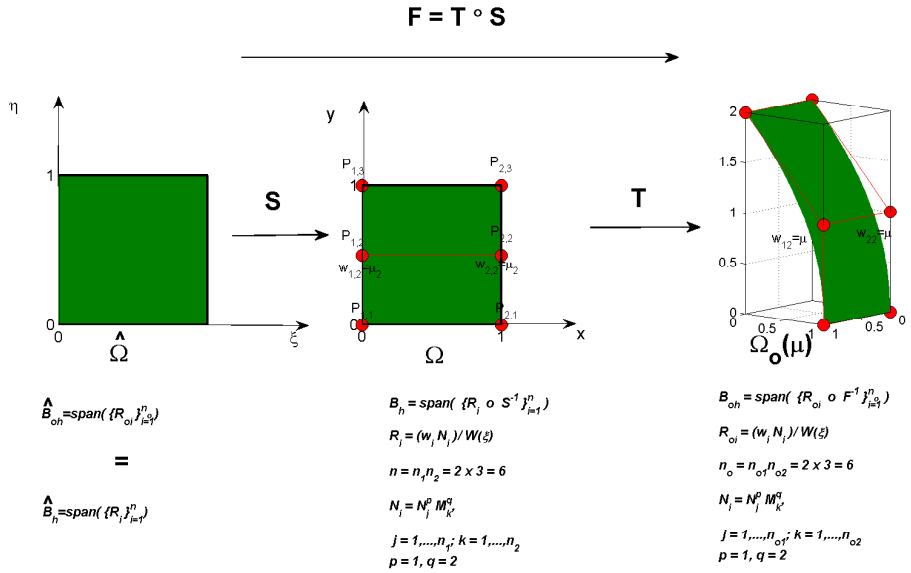


Figure 4.1: We choose Ω such that, the NURBS space in the original domain ($\mathcal{B}_{oh}$) and the one in the reference domain ($\mathcal{B}_h$), if traced back into the parameter domain as $\hat{\mathcal{B}}_{oh}$ and $\hat{\mathcal{B}}_h$ respectively, are coincident. In this way, however, Ω is still $\boldsymbol{\mu}-$*dependent*.

4.3 Discrete Empirical Interpolation Method

The Discrete Empirical Interpolation Method (DEIM) is a variant of EIM, and as EIM, its idea is to approximate a nonlinear function $\boldsymbol{f} : \mathcal{D} \to \boldsymbol{f}(\boldsymbol{\mu}) \in \mathbb{R}^{\mathcal{N}}$ in a low dimensional space as follows [CS10]

$$\boldsymbol{f}(\boldsymbol{\mu}) \approx \boldsymbol{f}_M(\boldsymbol{\mu}) = \boldsymbol{\Phi}\boldsymbol{\Theta}(\boldsymbol{\mu}), \tag{4.8}$$

where $\boldsymbol{\Phi} = [\Phi_1, ..., \Phi_{M_f}] \in \mathbb{R}^{\mathcal{N}\times M_f}$ is a basis of functions and $\boldsymbol{\Theta}(\boldsymbol{\mu}) \in \mathbb{R}^{M_f}$ is the corresponding coefficients vector, with $M_f \ll \mathcal{N}$. As in the EIM case, the basis is computed only once offline, while, for each new parameter $\boldsymbol{\mu}$ the coefficients $\boldsymbol{\Theta}(\boldsymbol{\mu})$ are computed online. The main difference between EIM and DEIM, is that in the EIM algorithm the selection of both the basis and the interpolation points is done following a greedy procedure, while DEIM uses the POD method for the former. We report below the DEIM algorithm:

Input: a set of snapshots $\boldsymbol{\Lambda} = [\boldsymbol{f}(\boldsymbol{\mu}_1), ..., \boldsymbol{f}(\boldsymbol{\mu}_{n_s})] \in \mathbb{R}^{\mathcal{N}\times n_s}$, a tolerance *tol*.
$[\Phi_1, ..., \Phi_{M_f}] = POD(\boldsymbol{\Lambda}, tol)$
$\mathcal{F} = \arg\max|\Phi_1|$
for $k = 2 : M_f$
  compute residual $\boldsymbol{r} = \Phi_k - \boldsymbol{\Phi}\boldsymbol{\Phi}_{\mathcal{F}}^{-1}\Phi_{k,\mathcal{F}}$
  $i = \arg\max|\boldsymbol{r}|$
  $\mathcal{F} = \mathcal{F} \cup i$
  $\boldsymbol{\Phi} = [\boldsymbol{\Phi}\ , \Phi_k]$

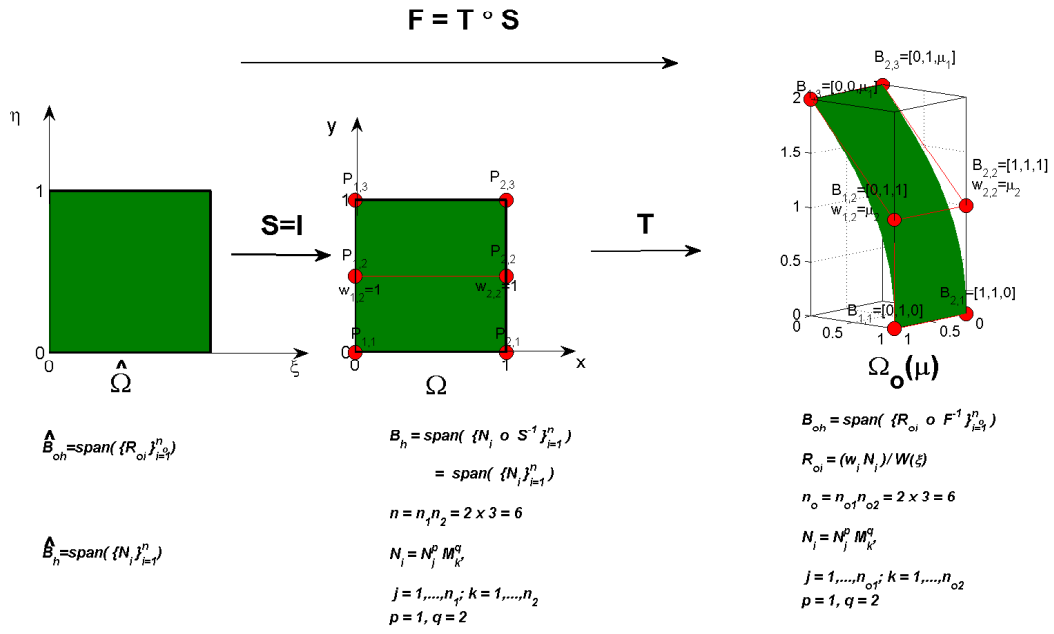


Figure 4.2: The only way to make Ω $\boldsymbol{\mu}$-independent is to assign fixed values to all the NURBS weights, also to the ones corresponding to parametrized weights in the original domain. In this way, the drawback is that the NURBS space in the original domain ($\mathcal{B}_{oh}$) and the one in the reference domain ($\mathcal{B}_h$), if traced back into the parameter domain as $\hat{\mathcal{B}}_{oh}$ and $\hat{\mathcal{B}}_h$ respectively, do not coincide anymore.

end
Output: a basis $\boldsymbol{\Phi} \in \mathbb{R}^{\mathcal{N} \times M_f}$, a set of indexes $\mathcal{F} \in \mathbb{R}^{M_f}$.

The DEIM procedure starts by choosing a set of parameter samples Ξ_{train} of cardinality n_s and by computing a set of snapshots by sampling $\boldsymbol{f}(\boldsymbol{\mu})$ at the values $\boldsymbol{\mu} \in \Xi_{train}$. Then, DEIM applies POD (see Section 2.4.3) to extract a basis from the set of snapshots $\boldsymbol{\Lambda}$. In the following lines of the algorithm, DEIM constructs a set of indexes $\mathcal{F}$, such that $\mathcal{F} \subset \{1, ..., \mathcal{N}\}$, and $|\mathcal{F}| = M_f$. At each step, the selected index is the one corresponding to the entry of the residual $\boldsymbol{r}$ where its maximum occurs. As output of the algorithm, we will have a basis $\boldsymbol{\Phi} \in \mathcal{R}^{\mathcal{N} \times M_f}$ and the set of indexes $\mathcal{F} \subset \{1, ..., \mathcal{N}\}$, $|\mathcal{F}| = M_f$. This set of indexes $\mathcal{F}$ is used in order to compute $\boldsymbol{\Theta}(\boldsymbol{\mu})$ that appears in (4.8). To this end, we impose an interpolation condition in these properly selected entries of the vector $\boldsymbol{f}(\boldsymbol{\mu})$ in the following way: given a new parameter $\boldsymbol{\mu}$ we compute

$$\boldsymbol{\Phi}_{\mathcal{F}} \boldsymbol{\Theta}(\boldsymbol{\mu}) = \boldsymbol{f}_{\mathcal{F}}(\boldsymbol{\mu}), \tag{4.9}$$

where $\boldsymbol{\Phi}_{\mathcal{F}} \in \mathbb{R}^{M_f \times M_f}$, is the matrix formed by the $\mathcal{F}$ rows of $\boldsymbol{\Phi}$ and $\boldsymbol{f}_{\mathcal{F}}(\boldsymbol{\mu}) \in \mathbb{R}^{M_f}$ is the vector formed by the $\mathcal{F}$ rows of $\boldsymbol{f}$. As result,

$$\boldsymbol{f}_M(\boldsymbol{\mu}) = \boldsymbol{\Phi} \boldsymbol{\Phi}_{\mathcal{F}}^{-1} \boldsymbol{f}_{\mathcal{F}}(\boldsymbol{\mu}). \tag{4.10}$$

Until now we have just applied DEIM to a generic function. In the next section, we will see how to directly apply DEIM to the nonaffine parametrized vectors and matrices of problem (4.5), in order to restore their affine dependence. Since we are not considering the reference domain anymore, in the following we are going to omit the subscript o, thus implying that we are always working on physical (original) configurations.

4.4 Matrix DEIM application

By following a procedure similar to the one adopted in Section 1.4.4, we can rewrite problem (4.5) as a linear system in the following way:

$$\boldsymbol{A}(\boldsymbol{\mu}) \boldsymbol{u}(\boldsymbol{\mu}) = \boldsymbol{L}(\boldsymbol{\mu}). \tag{4.11}$$

For what said in the previous section, we can use DEIM directly on the matrix $\boldsymbol{A}(\boldsymbol{\mu})$ (the case of the right hand side vector $\boldsymbol{L}(\boldsymbol{\mu})$ is treated in the same way) to obtain its MDEIM affine approximation that can be expressed as [NMA15]

$$\boldsymbol{A}(\boldsymbol{\mu}) \approx \boldsymbol{A}_M(\boldsymbol{\mu}) = \sum_{m=1}^{M_a} \Theta_m(\boldsymbol{\mu}) \boldsymbol{A}^m. \tag{4.12}$$

Thus, we are interested in finding $M \ll \mathcal{N}$ functions $\Theta_m : \mathcal{D} \to \mathbb{R}$ and parameter independent matrices $\boldsymbol{A}^m \in \mathbb{R}^{\mathcal{N} \times \mathcal{N}}$, $1 \leq m \leq M_a$, such that (4.12) holds. As already said, these functions and matrices are computed in the offline stage as we will see later. We start by vectorizing the matrix $\boldsymbol{A}(\boldsymbol{\mu})$ as $\boldsymbol{a}(\boldsymbol{\mu}) = \text{vec}(\boldsymbol{A}(\boldsymbol{\mu})) \in \mathbb{R}^{\mathcal{N}^2}$; then, (4.12) can be reformulated as: find $\{\boldsymbol{\Phi}, \boldsymbol{\Theta}(\boldsymbol{\mu})\}$ such that

$$\boldsymbol{a}(\boldsymbol{\mu}) \approx \boldsymbol{a}_M(\boldsymbol{\mu}) = \boldsymbol{\Phi} \boldsymbol{\Theta}(\boldsymbol{\mu}), \tag{4.13}$$

where $\boldsymbol{\Phi} \in \mathbb{R}^{\mathcal{N}^2 \times M_a}$ is a parameter-independent basis, and $\boldsymbol{\Theta}(\boldsymbol{\mu}) \in \mathbb{R}^{M_a}$ the coefficients vector:

$$\boldsymbol{\Phi} = [\text{vec}(\boldsymbol{A}_1), ..., \text{vec}(\boldsymbol{A}_{M_a})], \quad \boldsymbol{\Theta}(\boldsymbol{\mu}) = [\Theta_1(\boldsymbol{\mu}), ..., \Theta_{M_a}(\boldsymbol{\mu})]^T.$$

Then, we apply MDEIM to obtain the basis $\boldsymbol{\Phi}$ and the interpolation indexes $\mathcal{F} \subset \{1, ..., \mathcal{N}^2\}$. Online, given a $\boldsymbol{\mu} \in \mathcal{D}$ we compute $\boldsymbol{a}_m(\boldsymbol{\mu})$ as

$$\boldsymbol{a}_M(\boldsymbol{\mu}) = \boldsymbol{\Phi}\boldsymbol{\Theta}(\boldsymbol{\mu}), \quad \text{with } \boldsymbol{\Theta}(\boldsymbol{\mu}) = \boldsymbol{\Phi}_{\mathcal{F}}^{-1}\boldsymbol{a}_{\mathcal{F}}(\boldsymbol{\mu}).$$

Finally, by reversing the vec operation, we obtain the approximated matrix $\boldsymbol{A}_M(\boldsymbol{\mu})$. We remark that, with $\boldsymbol{a}_{\mathcal{F}}(\boldsymbol{\mu})$ we mean the evaluation of the vectorized matrix $\boldsymbol{a}(\boldsymbol{\mu})$ in the selected entries $\mathcal{F}$. This operation requires the following offline steps:

- detect the *reduced dofs*: each index $i \in \mathcal{F}$ in the vector format corresponds to a couple of indexes (row-column indexes in the matrix form) $(l, j) \in I \times J$, with $I, J \subset \{1, ..., \mathcal{N}\}$. The reduced dofs consists in the union of the sets I and J;
- define the *reduced nodes* of the mesh as the set of nodes associated to the reduced dofs. In IGA context, each of these nodes represents a control point and thus is associated to a NURBS basis function.
- detect the *reduced elements* of the mesh: for each NURBS basis function associated to a reduced node, we take the elements on which it does not vanish (i.e. where it possesses support). We recall that the support of NURBS basis functions built by means of bivariate B-spline basis functions of degree p in one direction and q in the other one is always $(p + 1)(q + 1)$ elements.

At the online stage, we just have to assemble the matrix $\boldsymbol{A}(\boldsymbol{\mu})$ in the reduced elements. To this end, we can use the usual assembler passing as input only the reduced elements, while keeping the same global and local numbering for the elements and for the basis functions. The resulting matrix has still dimension $\mathcal{N} \times \mathcal{N}$ but is extremely sparse, since only the elements associated to the reduced elements are actually nonzero. Finally

$$\boldsymbol{\Theta}^a(\boldsymbol{\mu}) = \boldsymbol{\Phi}_{\mathcal{F}}^{-1}\boldsymbol{a}_{\mathcal{F}}(\boldsymbol{\mu}), \tag{4.14}$$

where $\boldsymbol{a}_{\mathcal{F}}(\boldsymbol{\mu}) = [\text{vec}(\boldsymbol{A}(\boldsymbol{\mu}))]_{\mathcal{F}}$ and analogously

$$\boldsymbol{\Theta}^f(\boldsymbol{\mu}) = \boldsymbol{\Phi}_{\mathcal{F}}^{-1}\boldsymbol{f}_{\mathcal{F}}(\boldsymbol{\mu}). \tag{4.15}$$

In this way, we obtain the system approximated by MDEIM

$$\boldsymbol{A}_M(\boldsymbol{\mu})\boldsymbol{u}_M(\boldsymbol{\mu}) = \boldsymbol{L}_M(\boldsymbol{\mu}), \tag{4.16}$$

which is the analogue of the EIM one (see (3.25)). More in detail,

$$\left(\sum_{m=1}^{M_a} \Theta_m^a(\boldsymbol{\mu})\boldsymbol{A}_m\right)\boldsymbol{u}_M(\boldsymbol{\mu}) = \sum_{m=1}^{M_f} \Theta_m^f(\boldsymbol{\mu})\boldsymbol{L}_m. \tag{4.17}$$

4.5 Approximation with RB

We can now combine MDEIM with the reduced basis technique. Indeed, system (4.16) is of dimension $\mathcal{N} \times \mathcal{N}$ and we would like to reduce its dimension by adopting the RB approach. We do that by computing a set of basis by means of a POD or greedy approach (see Sections 2.4.3 and 2.4.2) and so obtain reduced matrices and right hand side vectors as in (2.32). In this way we obtain the following reduced system:

$$\boldsymbol{A}_{N,M}(\boldsymbol{\mu})\boldsymbol{u}_{N,M}(\boldsymbol{\mu}) = \boldsymbol{L}_{N,M}(\boldsymbol{\mu}), \tag{4.18}$$

that can be written explicitely as

$$\left(\sum_{m=1}^{M_A} \Theta_m^a(\boldsymbol{\mu})\boldsymbol{A}_N^m\right)\boldsymbol{u}_{N,M}(\boldsymbol{\mu}) = \sum_{m=1}^{M_f} \Theta_m^f(\boldsymbol{\mu})\boldsymbol{L}_N^m,$$

where the $\boldsymbol{A}_N^m \in \mathbb{R}^{N\times N}$ are the precomputable matrices of small size, while the weights $\boldsymbol{\Theta}^a(\boldsymbol{\mu}) = [\Theta_1^a(\boldsymbol{\mu})...\Theta_{M_a}^a(\boldsymbol{\mu})]$ and $\boldsymbol{\Theta}^f(\boldsymbol{\mu}) = [\Theta_1^f(\boldsymbol{\mu})...\Theta_{M_f}^f(\boldsymbol{\mu})]$ have been defined in (4.14) and (4.15).

Once we have computed the RB solution, we can recover the full-order solution as

$$\boldsymbol{u}_{N,M}^{\mathcal{N}}(\boldsymbol{\mu}) = \boldsymbol{\mathcal{Z}}\boldsymbol{u}_{N,M}(\boldsymbol{\mu}), \tag{4.19}$$

being $\boldsymbol{\mathcal{Z}} \in \mathbb{R}^{\mathcal{N}\times N}$ the basis matrix defined in (2.20) or in (2.25), if a greedy or a POD sampling strategy has been adopted, respectively.

4.6 Numerical results

In this section, we present the numerical results obtained for problem (4.4) defined on the geometry built in Example 1.8, that is a surface parametrized with respect to both NURBS weights and control points coordinates. To solve the problem we perform first a MDEIM approximation and then a RB reduction. We take $f_o(\boldsymbol{\mu}) = \sin(y)$, consider homogeneous Dirichlet boundary conditions, NURBS basis functions of degree 2, and realize a h-refinement of the domain in order to have a 25×60 grid for a total of $\mathcal{N} = 1500$ degrees of freedom (considering only inner nodes). As anticipated, we consider two geometrical parameters, namely μ_1 for the horizontal and vertical displacement of the control points $\boldsymbol{B}_{12}$ and $\boldsymbol{B}_{22}$, and μ_2 for the variation of the weights associated to these control points. The parameter domain is given by $\mathcal{D} = [1, 2] \times [0.5, 1.2]$.

We start by presenting the results for MDEIM, used to approximate the parametrized matrix and right hand side vector of the problem written in the form (4.16). We define a set of 400 parameter samples Ξ_{train} and compute the set of the corresponding 400 matrix and right hand side vectors snapshots. To do that, we have to modify the NURBS geometry for each parameter sample and assemble the current stiffness matrix $\boldsymbol{A}(\boldsymbol{\mu})$ and right hand side $\boldsymbol{L}(\boldsymbol{\mu})$. As explained in Section 4.4 we vectorize each of these matrices and reformulate the problem in the form (4.13). Then, we apply MDEIM to obtain the basis $\boldsymbol{\Phi}$ and the interpolation indexes $\mathcal{F} \subset \{1, ..., \mathcal{N}^2\}$. To do that, we apply the POD algorithm to the snapshots matrices of the vectorized stiffness matrices and right-hand side vectors, respectively. In Fig. 4.3 we report the resulting singular values (just for the first 20 samples for a better visualization) and conclude that the first $M_a = 15$ singular values of the snapshots matrix of the vectorized

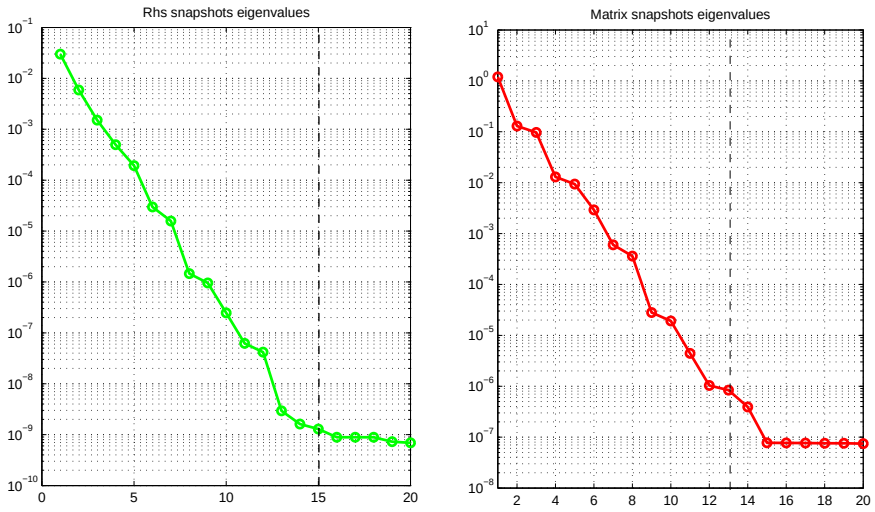


Figure 4.3: Singular values of the snapshots matrix for the rhs (left) and for the stiffness matrix (right) for polynomial degree 2.

matrices and the first $M_f = 13$ singular values of the snapshots matrix of the right-hand side vectors, retain most of the energy of the system.

The MDEIM algorithm returns also the interpolation indexes associated to the selected bases. By means of these indexes we identify the reduced nodes of the mesh. We already know that each node corresponds to a NURBS basis function. Finally, for each basis function, we look for its support. These are the reduced elements. In this way from the selected control points (reduced dofs) we find the reduced elements and we proceed by assembling $\boldsymbol{A}$ and $\boldsymbol{L}$ only on these elements. To achieve this goal we employ the usual assembler adopted for the operators defined on the full mesh, but passing as input only the reduced elements. As a consequence, the assembling operation requires a very low computational cost since the number of elements involved is reduced. The resulting matrices and vectors are still of size $\mathcal{N} \times \mathcal{N}$ and $\mathcal{N}$ respectively, but they are extremely sparse since only the entries associated to the reduced elements are actually non zero. The procedure selects 200 elements (out of 1500). The reduced mesh is shown in Fig. 4.4. It is evident that the reduced elements are selected by groups of 9; as a matter of fact, for each reduced node, as already said, we identify the associated basis function, that in this case is originated by bivariate B-spline basis functions of degree $p = 2$ in ξ-direction and $q = 2$ in η-direction, so its support consists of $(p + 1)(q + 1) = 9$ elements. We notice that near the boundary, since repeated knots are present (we recall that we always use open knot vectors, Section 1.1.1) some elements have zero measure, and the reduced elements appear as groups of 6 elements.

Then, we proceed with the RB approximation by applying the greedy algorithm. In this case, we do not impose to the greedy algorithm to satisfy tolerance, but rather set a maximum number of basis $N_{max} = 30$ to be selected, and then verify that they are enough to get the convergence of the procedure. In Fig. 4.5 we report the following errors

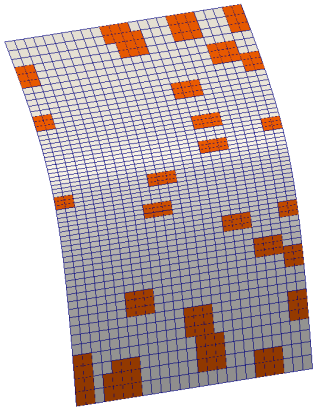


Figure 4.4: Full mesh (1500 elements) in gray and reduced mesh (200 elements) in orange, for polynomial degree 2.

$$\epsilon_{ave,rel}^{IGA,DEIM-greedy-RB} = mean_{\boldsymbol{\mu}\in\Xi_{test}} \frac{\|\boldsymbol{u}(\boldsymbol{\mu}) - \boldsymbol{\mathcal{Z}}\boldsymbol{u}_{N,M}(\boldsymbol{\mu})\|_V}{\|\boldsymbol{u}(\boldsymbol{\mu})\|_V}, \tag{4.20a}$$

$$\epsilon_{ave,rel}^{DEIM,DEIM-greedy-RB} = mean_{\boldsymbol{\mu}\in\Xi_{test}} \frac{\|\boldsymbol{u}_M(\boldsymbol{\mu}) - \boldsymbol{\mathcal{Z}}\boldsymbol{u}_{N,M}(\boldsymbol{\mu})\|_V}{\|\boldsymbol{u}_M(\boldsymbol{\mu})\|_V}. \tag{4.20b}$$

We recall that (4.20a) is the average relative error between $\boldsymbol{u}(\boldsymbol{\mu})$, solution of system (4.11), and $\boldsymbol{u}_{N,M}^{\mathcal{N}}(\boldsymbol{\mu})$, solution of problem (4.18), approximated by means of MDEIM, reduced by means of greedy (DEIM-greedy-RB problem), and reported to the full-order dimension. Similarly, (4.20b) represents the average relative error between $\boldsymbol{u}_M(\boldsymbol{\mu})$, solution of system (4.16) approximated by means of MDEIM, and $\boldsymbol{u}_{N,M}^{\mathcal{N}}(\boldsymbol{\mu})$. In both the cases, the error, computed on the set Ξ_{test}, is shown as a function of the number of basis N.

In order to better appreciate the large computational savings allowed by the RB methodology, we now report the numerical results obtained for the same problem solved on a finer computational mesh. In particular, we consider a 50×125 grid, for a total of 6250 elements. We also compare the results obtained on this grid for different polynomial degrees. In particular, we consider NURBS generated by bivariate B-spline basis functions of degrees 2, 3, and 4 in both directions (in the following we refer to these three cases as P2, P3, P4, respectively). To this end, starting from the non refined geometry (see Fig. 1.16), we perform a suitable p-refinement to increase the polynomial degree in the interested directions and a h-refinements to obtain the new mesh. The number of degrees of freedom increases to 6250 in the first case, to 6426 in the second case, and to 6604 in the third one. For all the considered situations, we apply MDEIM to obtain the basis $\boldsymbol{\Phi}$ and the interpolation indexes $\mathcal{F} \subset \{1, ..., \mathcal{N}^2\}$. To this end, we vectorize the matrices and right-hand side vectors snapshots and apply the POD. In Fig. 4.6 we compare the results for the three polynomial degrees on the same mesh grid (the one with 6250 elements). We observe that the singular values do not change by varying the polynomial degree. As a matter of fact, the POD procedure operates on the operators discretized in the parameters space; in particular, as seen in Section 2.4.3, it operates on a matrix having as columns n_{train} snapshots computed on n_{train} geometrical parameters. As a consequence, if the geometric parametrization of the computational domain does not change and we repeat the procedure on the same set of samples, the POD result does not change,

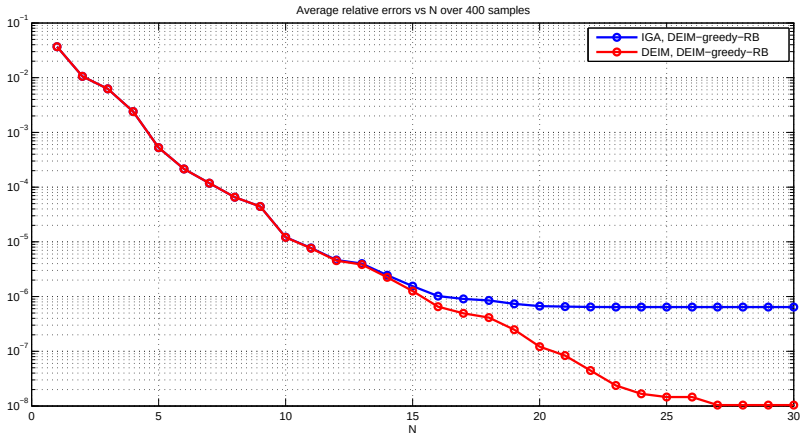


Figure 4.5: Average relative errors: $\epsilon_{ave,rel}^{IGA,DEIM-greedy-RB}$ between IGA and DEIM-greedy-RB vs N (blue) and $\epsilon_{ave,rel}^{DEIM,DEIM-greedy-RB}$ between DEIM and DEIM-greedy-RB vs N (red).

and this is the case of p-refinement. As a matter of fact, in IGA context, by performing p-refinement the parametrization and the geometry are preserved (see Section 1.3.2), and the resulting singular values do not vary. Also in this case, we conclude that $M_a = 13$ and $M_f = 15$ singular vales for the snapshots matrix of $\boldsymbol{A}$ and $\boldsymbol{L}$ retain most of the energy of the system.

Since we are considering $M_a = 13$ and $M_f = 15$ matrix and vector bases for all the three cases considered, the number of the reduced dofs, associated to the indexes selected by MDEIM, is the same. However, B-spline basis functions of degree p in ξ-direction and q in η-direction have support on $(p+1)(q+1)$ elements. As a consequence, higher-degree functions have support over much larger portions of the domain. Therefore, the procedure selects 210, 324, 449 elements (out of 6250) for the P2, P3, P4 case respectively, as shown in Fig.4.7.

Then, we proceed with the RB approximation by applying the greedy algorithm. Also in this case, we set a maximum number of basis $N_{max} = 30$ to be selected, as done before, and then verify that they are enough to ensure the convergence of the procedure. In Fig. 4.8, 4.9, 4.10 we report the errors $\epsilon_{ave,rel}^{IGA,DEIM-greedy-RB}$ and $\epsilon_{ave,rel}^{DEIM,DEIM-greedy-RB}$ defined as in (4.20), for P2, P3, and P4. All these plots show that the behavior of the errors is basically the same up to $N = 15$, so that the effect of system approximation by means of MDEIM is negligible in the reduced model. For $N > 15$, instead, the error between IGA and MDEIM affects the error of the reduced model. In Fig. 4.11 we compare $\epsilon_{ave,rel}^{IGA,DEIM-greedy-RB}$ for the different polynomial degrees and conclude that P3 and P4 have slightly better convergence properties. However, because of the larger number of reduced elements selected, they require at the same time higher computational costs.

In order to summarize the obtained results and conclude about the computational advantages and disadvantages of adopting a higher polynomial degree or a finer mesh, in Table 4.1 we report some data concerning 9 different cases for meshes of 1500, 6250, and 10075 elements, and polynomial degrees 2, 3, 4. In particular in this table we report the computational times requested to assemble and solve the IGA problem for each parameter evaluation and the ones requested to solve online the DEIM-greedy-RB problem. In this context we want to remark

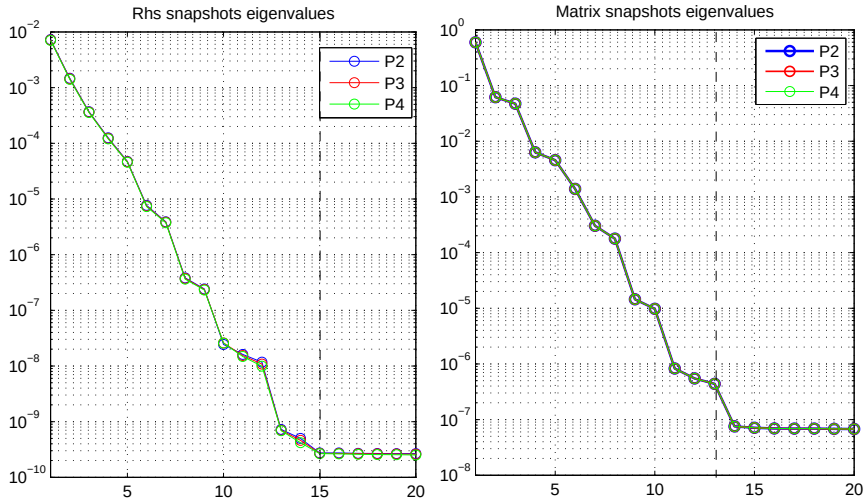


Figure 4.6: Singular values of the snapshots matrices of the vectorized stiffness matrix (right) and rights hand side vectors (left) for polynomial degrees 2, 3, 4. The singular values do not change by varying the polynomial degree.

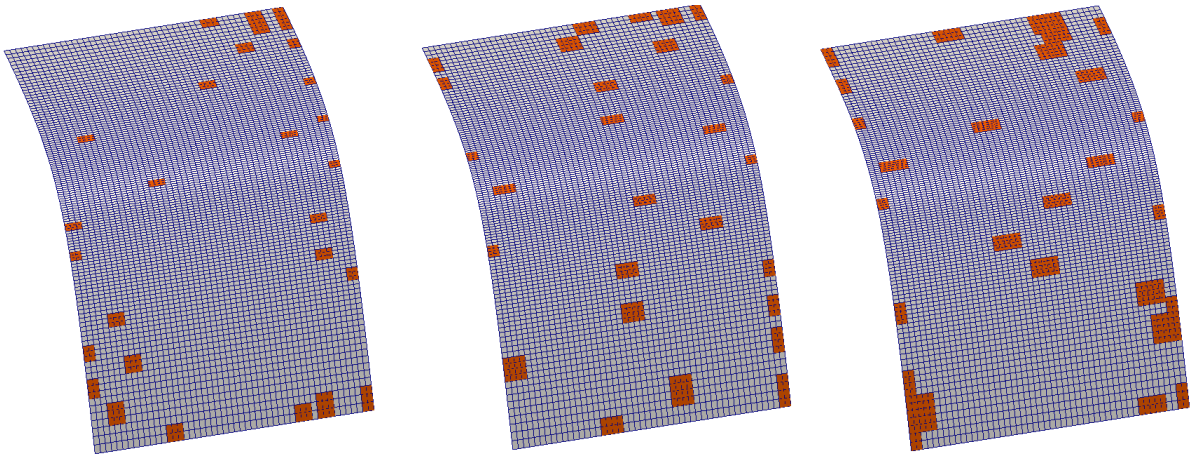


Figure 4.7: Full mesh (6250 elements) in gray and reduced mesh in orange: 210 elements for P2, 324 elements for P3, 449 elements for P4.

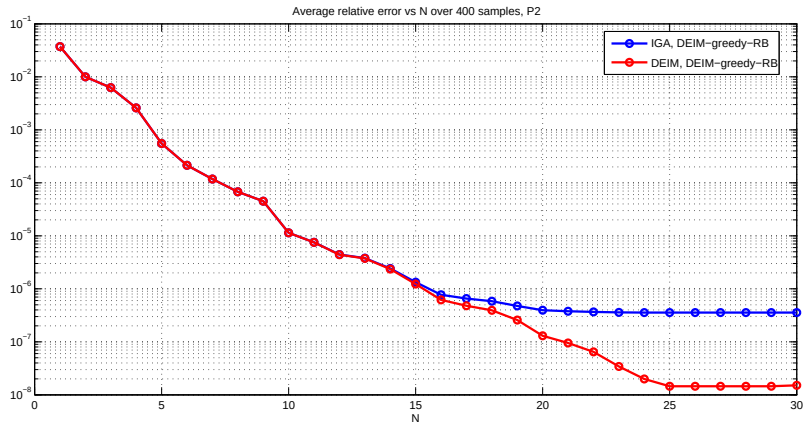


Figure 4.8: Average relative errors for P2 case: $\epsilon_{ave,rel}^{IGA,DEIM-greedy-RB}$ between IGA and DEIM-greedy-RB vs N (blue) and $\epsilon_{ave,rel}^{DEIM,DEIM-greedy-RB}$ between DEIM and DEIM-greedy-RB vs N (red).

an important difference between FE and IGA. When solving a parametrized PDE with a FE method, for each parameter evaluation, it is not requested to modify the basis functions, since they are not parameter dependent. On the other hand, since in IGA the basis functions used for the analysis are the same used to build the parametric dependent geometry, for each parameter evaluation, it could be requested to recompute the basis functions. In particular this happens when the geometry is built by means of NURBS basis functions and the parametrization affects the NURBS weights. It means that online, for each new parameter, we have first to modify the NURBS basis functions, and then we can assemble and solve the reduced system. To keep the computational complexity independent on $\mathcal{N}$, we modify only the NURBS having support on the reduced elements.

From the data reported in Table 4.1 we conclude that the case that allows the largest computational saving is the P2 one on the grid made of 6250 elements. So, for this case, we also perform an analysis about its convergence properties varying the number of terms selected by MDEIM. In Fig. 4.12 we plot the average relative error between $\boldsymbol{u}(\boldsymbol{\mu})$ and $\boldsymbol{u}_{N,M}^{\mathcal{N}}(\boldsymbol{\mu})$, computed on a set of 400 parameters samples, as a function of the number of basis N and by considering an increasing number of affine terms Q_a and Q_f. We observe that, for a higher number of affine terms considered, the error decreases, and that for a fixed couple Q_a, Q_f, the error decreases by increasing N.

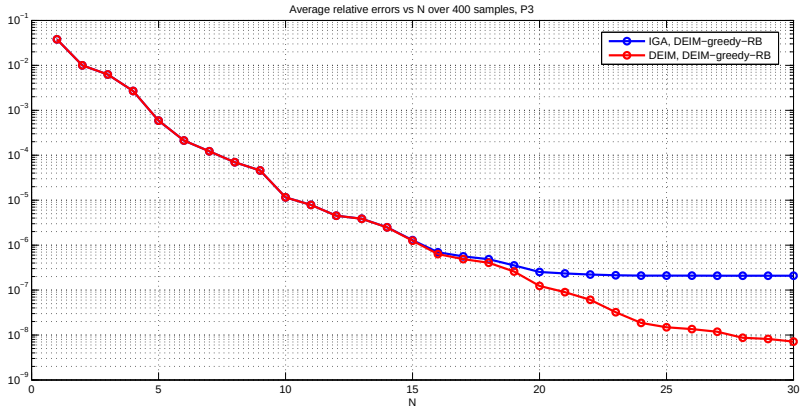


Figure 4.9: Average relative errors for P3 case: $\epsilon_{ave,rel}^{IGA,DEIM-greedy-RB}$ between IGA and DEIM-greedy-RB vs N (blue) and $\epsilon_{ave,rel}^{DEIM,DEIM-greedy-RB}$ between DEIM and DEIM-greedy-RB vs N (red).

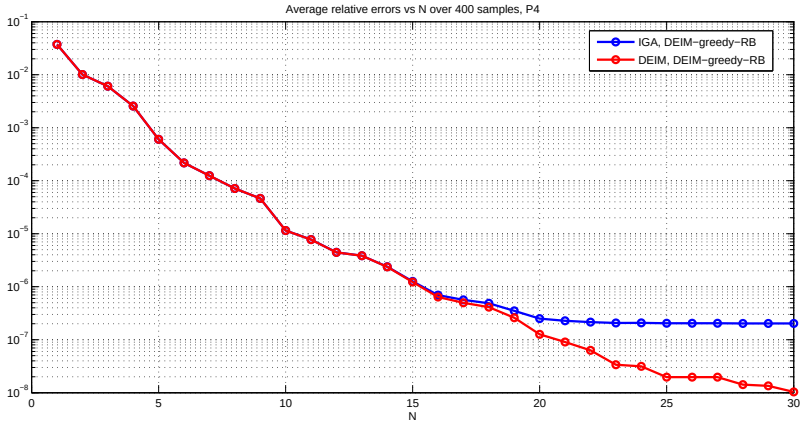


Figure 4.10: Average relative errors for P4: $\epsilon_{ave,rel}^{IGA,DEIM-greedy-RB}$ between IGA and DEIM-greedy-RB vs N (blue) and $\epsilon_{ave,rel}^{DEIM,DEIM-greedy-RB}$ between DEIM and DEIM-greedy-RB vs N (red).

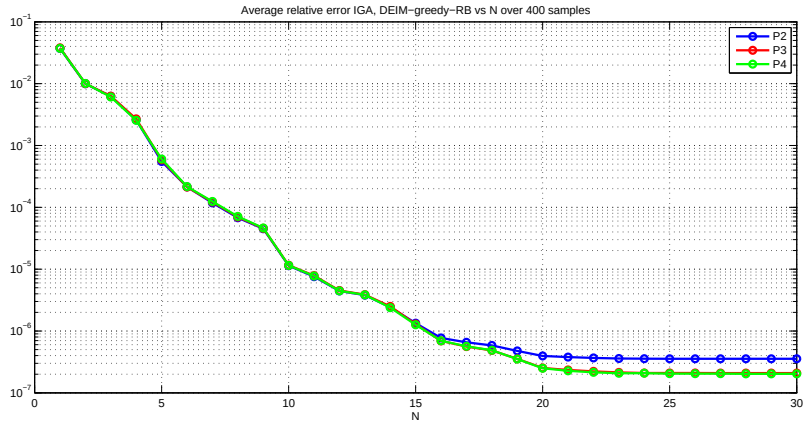


Figure 4.11: Comparison of the average relative error $\epsilon_{ave,rel}^{IGA,DEIM-greedy-RB}$ between P2, P3, and P4

Table 4.1: *Computational times t_{IGA} and t_{RB}, requested to assemble and solve the IGA problem and the DEIM-greedy-RB problem respectively, for P2, P3, and P4 cases, and for meshes made of 1500, 6250, and 10075 elements.*

Degree	Elements	$\mathcal{N}$	Reduced Elements	M_a	M_f	N	t_{IGA} [s]	t_{RB} [s]	$\frac{t_{IGA}}{t_{RB}}$
P2	1500	1500	200	13	15	30	3.43	0.57	6.02
	6250	6250	210	13	15	30	11.0	0.67	16.5
	10075	10075	213	13	15	30	17.9	0.74	24.1
P3	1500	1586	293	13	15	30	6.12	1.73	3.53
	6250	6426	324	13	15	30	22.6	2.06	11.0
	10075	10519	350	13	15	30	35.0	3.01	11.6
P4	1500	1674	431	13	15	30	11.5	4.96	2.31
	6250	6604	449	13	15	30	42.0	5.96	7.05
	10075	10519	478	13	15	30	66.9	6.63	10.1

4.7 MDEIM for parametrized control points: comparison with EIM

In this section, we want to highlight the fact that, since MDEIM directly operates on every kind of matrix or vector, it can also be used in the first case discussed, where only the control points coordinates were parameter dependent.

For this reason, we now consider again theparametrized Poisson problem (3.3) introduced in Section 3.1, defined on the parametrized quarter of annulus of Example 1.6. For the resolution of the problem, we use the same data reported in Section 3.8.1: $\delta = 1$, $f_o(\boldsymbol{\mu}) = 1$, homogeneous Dirichlet boundary conditions, and $\boldsymbol{\mu} = \mu \in \mathcal{D} = [0.5, 1]$. Moreover, we use

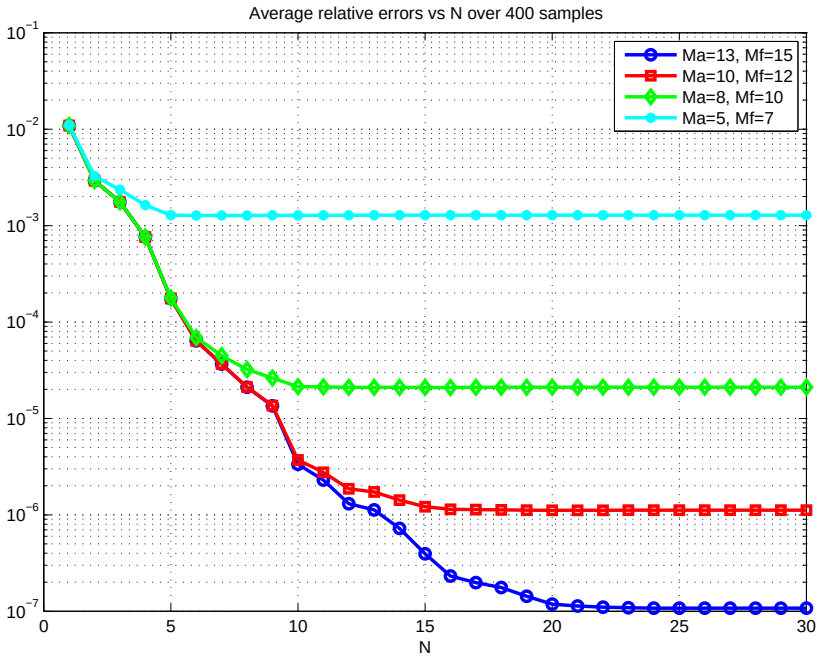


Figure 4.12: Average relative errors $\epsilon_{ave,rel}^{IGA,MDEIM-POD-RB}$ defined in (3.32), between IGA and MDEIM-POD-RB solutions vs N for different values of M_a and M_f, computed on a 400 samples parameter set Ξ_{test}.

NURBS basis functions of degree 2 (in the following we will refer to this case as to P2) meaning that we realize a p-refinement of the original domain such that $p = q = 2$. Finally, we realize an h-refinement in order to have a 25×60 grid yielding $\mathcal{N} = 1500$ degrees of freedom (considering only inner nodes), exactly as we did in Section 3.8.1.

As already said, in this section we aim to compare the EIM and MDEIM performances on the problem at hand. So now, instead of computing the affine approximation of the problem by using EIM, we directly apply MDEIM to the parametrized matrices and vectors appearing in the problem formulation. To perform MDEIM we used the same tolerance used for EIM, that is $\text{tol}_{MDEIM} = 10^{-5}$, and the same train set of 50 parameters samples. The result is that MDEIM entails less affine terms than EIM, in particular 3 for the stiffness matrix and 2 for the right-hand side vector, allowing a larger computational saving in the offline stage. In Table 4.2 we compare the results provided by EIM and MDEIM for the problem at hand. Then, since the number of affine terms is small in both cases, we apply the greedy algorithm with tolerance $\text{tol} = 10^{-6}$ over a 150 parameters samples set. The greedy algorithm provides as output a reduced space of dimension $N_{max} = 10$.

In Table 4.2 we report the time required to solve online the reduced problem when using both EIM and MDEIM. Moreover, we compare the speed-up allowed by the two methods; in particular, the EIM-RB online resolution is about 11.5 times faster than the IGA resolution, while the MDEIM-RB online resolution is more than 26 times faster than IGA. As expected, since MDEIM entails less affine terms than EIM, it provides a larger computational saving, both in the offline and in the online stages, and that is why, in the next chapter we will always resort to MDEIM to make the problems of interest affine.

Table 4.2: *Data of problem (3.3), EIM-RB and MDEIM-RB results, and computational times required offline and online.*

	EIM	MDEIM
Number of parameters	1	1
Number of samples	50	50
tol	10^{-5}	10^{-5}
Affine matrix components $\boldsymbol{A}^m$	16	3
Affine rhs components $\boldsymbol{L}^m$	2	2
$\mathcal{N}$	1500	1500
Greedy tolerance	10^{-6}	10^{-6}
Sample train (greedy)	150	150
Greedy space dimension N	9	10
t_{IGA}	3.68 s	3.68 s
Greedy construction time (offline)	2.43 min	1.09 min
$t_{greedy-RB}$ (online)	0.32 s	0.14 s
$t_{IGA}/t_{greedy-RB}$	11.5	26.28

Chapter 5

Reduced Basis method for Isogeometric Kirchoff-Love shells

In this chapter, we consider the RB method applied to Isogeometric shell analysis with Kirchoff-Love elements.

The main objective of shell theory is to predict the stress and the displacement fields arising in an elastic shell in response to given forces. In classical shell theory one has to distinguish between thick shells ($R/h < 20$) and thin shells ($20 \leq R/h$), where R is the shell radius and h is the shell thickness ([KBLW09]). The appropriate theory to describe thick shells is the Reissner-Mindlin theory where transverse shear deformations are taken into account ([SFR89]). However, for these shells also the Kirchoff-Love theory is applicable, which assumes that transverse shear deformations are negligible. For very thin shells ($1000 < R/h$) the deformations usually cannot be described by geometrically linear behavior and a geometrically nonlinear description is necessary. For a comprehensive exposition of the theory of elastic shells, we refer the reader to ([CM08, Cia00]).

In this chapter, we deal with a Kirchoff-Love shell model for which the problem is approximated by IGA ([KBLW09, MRA13]). The Kirchoff-Love theory of thin shells is characterized by energy functionals which depend on curvature; consequently, they contain second order derivatives of displacement and the resulting equilibrium equations in turn take the form of fourth order PDEs. In this context, Kirchoff-Love model requires C^1 continuity between the elements if spatially approximated with the standard Galerkin method. Indeed, in order to ensure that the bending energy is finite, the trial and test functions should belong to H^2, that is square-integrable functions whose first and second order derivatives are themselves square-integrable. In this context, it is not possible to achieve C^1 continuity when using Lagrange polynomial basis functions, as for the Finite Element method. On the other hand, NURBS as basis functions for the analysis have the significant advantage that the necessary continuities between elements are easily achieved. As a matter of fact, NURBS, being smooth higher order functions, allow great geometric flexibility and high order continuities at the same time, resulting ideally suited as basis functions for Kirchoff-Love shell elements ([BBHH10]).

Since we are interested in solving efficiently parametrized Kirchoff-Love shells, we study problems defined on parametrized computational domains and evaluate, by means of the MDEIM-RB strategy, the displacement fields changes under such parametrizations. We apply MDEIM to restore the affinity of the IGA linear system because, as written in Chapter 4, it can be used both for parametrized NURBS control points and weights and also because,

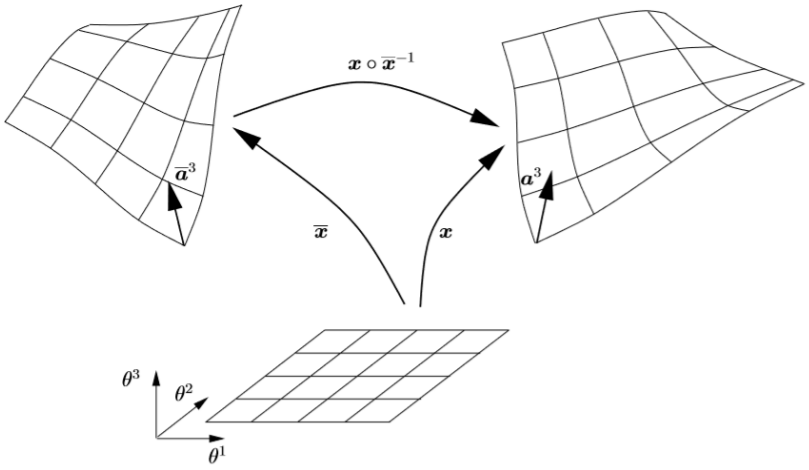


Figure 5.1: Shell geometry in the deformed and reference configuration. Picture taken from [COS00].

compared to EIM, it allows a larger computational saving in both the offline and the online stages.

In Section 5.1 we begin by summarizing the relevant equations of the Kirchoff-Love theory of shells. In particular, we focus on the linear theory of shells under static loading. In Section 5.2 we proceed with the IGA discretization of the Kirchoff-Love thin-shell theory. In Section 5.3, we are interested in solving the problem previously introduced on a parameter dependent domain, so we consider a parametrized version of the Kirchoff-Love model and provide a MDEIM-RB approximation of it. Finally, in Section 5.4 we test the method on a benchmark class of problems.

5.1 Thin-Shell boundary value problem

We consider a shell, whose thickness h is uniform, in an undeformed configuration characterized by a middle surface of domain $\bar{\Omega}$ and boundary $\bar{\Gamma} = \partial\bar{\Omega}$ ([COS00]). We apply a load to the shell that adopts a deformed configuration characterized by a middle surface of domain Ω and boundary $\Gamma = \partial\Omega$. In the Kirchoff-Love shell theory transverse shear deformation is neglected and the director, i.e. the vector normal to the middle surface, remains normal to the middle surface in the deformed configuration. Therefore, the description of the shell can be reduced to the description of its middle surface.

For the sake of simplicity, from now on, we consider the Einstein's convention for which repeated indexes are summed. In particular, Greek letters refers to indexes assuming values in the set $\{1, 2\}$, whereas Latin indexes in $\{1, 2, 3\}$. Let us introduce a system of curvilinear coordinates $\{\theta^1, \theta^2, \theta^3\}$, and the functions $\bar{\boldsymbol{x}}(\theta^1, \theta^2)$ and $\boldsymbol{x}(\theta^1, \theta^2)$ providing a parametric representation of the middle surface of the shell in the reference and deformed configuration, respectively (Fig. 5.1). Given these functions, we define the corresponding surface basis vectors as

$$\bar{\boldsymbol{a}}_\alpha = \frac{\partial \bar{\boldsymbol{x}}}{\partial \theta^\alpha}, \quad \boldsymbol{a}_\alpha = \frac{\partial \boldsymbol{x}}{\partial \theta^\alpha}, \quad \bar{\boldsymbol{a}}_3 = \frac{\bar{\boldsymbol{a}}_1 \times \bar{\boldsymbol{a}}_2}{|\bar{\boldsymbol{a}}_1 \times \bar{\boldsymbol{a}}_2|}, \quad \boldsymbol{a}_3 = \frac{\boldsymbol{a}_1 \times \boldsymbol{a}_2}{|\boldsymbol{a}_1 \times \boldsymbol{a}_2|} \tag{5.1}$$

The two vectors $\bar{\boldsymbol{a}}_1(\theta^1, \theta^2)$ and $\bar{\boldsymbol{a}}_2(\theta^1, \theta^2)$ define the tangent plane to the surface $\bar{\boldsymbol{x}}$ in (θ^1, θ^2),

while $\bar{\boldsymbol{a}}_3(\theta^1,\theta^2)$ in the reference configuration coincides with the normal to the undeformed middle surface of the shell in (θ^1,θ^2) and it results that

$$\bar{\boldsymbol{a}}_\alpha \cdot \bar{\boldsymbol{a}}_3 = 0, \quad |\bar{\boldsymbol{a}}_3| = 1; \tag{5.2}$$

the same considerations are valid for $\boldsymbol{a}_3$ in the deformed configuration.

Then, we define the position vectors $\bar{\boldsymbol{r}}$ and $\boldsymbol{r}$ of a material point in the reference and deformed configurations as

$$\bar{\boldsymbol{r}}(\theta^1,\theta^2,\theta^3) = \bar{\boldsymbol{x}}(\theta^1,\theta^2) + \theta^3\bar{\boldsymbol{a}}_3(\theta^1,\theta^2), \quad -\frac{h}{2} \le \theta^3 \le \frac{h}{2}, \tag{5.3a}$$

$$\boldsymbol{r}(\theta^1,\theta^2,\theta^3) = \boldsymbol{x}(\theta^1,\theta^2) + \theta^3\boldsymbol{a}_3(\theta^1,\theta^2), \quad -\frac{h}{2} \le \theta^3 \le \frac{h}{2}. \tag{5.3b}$$

In this context we also have to introduce the covariant components of the surface metric tensors defined as

$$\bar{a}_{\alpha\beta} = \bar{\boldsymbol{a}}_\alpha \cdot \bar{\boldsymbol{a}}_\beta, \quad a_{\alpha\beta} = \boldsymbol{a}_\alpha \cdot \boldsymbol{a}_\beta, \tag{5.4}$$

and the controvariant components of the undeformed and deformed surface metric tensors, $\bar{a}_{\alpha\beta}$ and $a_{\alpha\beta}$, respectively, as

$$\bar{a}^{\alpha\gamma}\bar{a}_{\gamma\beta} = \delta^\alpha_\beta, \quad a^{\alpha\gamma}a_{\gamma\beta} = \delta^\alpha_\beta; \tag{5.5}$$

we also note that

$$d\bar{\Omega} = \sqrt{\bar{a}}d\theta^1 d\theta^2. \tag{5.6}$$

By means of these elements, we can define the covariant base vectors in the reference and the current configurations as

$$\bar{\boldsymbol{g}}_\alpha = \frac{\partial\bar{\boldsymbol{r}}}{\partial\theta^\alpha} = \bar{\boldsymbol{a}}_\alpha + \theta^3\frac{\partial\bar{\boldsymbol{a}}_3}{\partial\theta^\alpha}, \quad \bar{\boldsymbol{g}}_3 = \frac{\partial\bar{\boldsymbol{r}}}{\partial\theta^3} = \bar{\boldsymbol{a}}_3 \tag{5.7a}$$

$$\boldsymbol{g}_\alpha = \frac{\partial\boldsymbol{r}}{\partial\theta^\alpha} = \boldsymbol{a}_\alpha + \theta^3\frac{\partial\boldsymbol{a}_3}{\partial\theta^\alpha}, \quad \boldsymbol{g}_3 = \frac{\partial\boldsymbol{r}}{\partial\theta^3} = \boldsymbol{a}_3; \tag{5.7b}$$

therefore, the corresponding covariant components of the metric tensors in both the configurations read as

$$\bar{g}_{ij} = \bar{\boldsymbol{g}}_i \cdot \bar{\boldsymbol{g}}_j, \quad g_{ij} = \boldsymbol{g}_i \cdot \boldsymbol{g}_j. \tag{5.8}$$

These metric tensors are used to define the Green-Lagrange strain tensor that reads

$$E_{ij} = \frac{1}{2}(g_{ij} - \bar{g}_{ij}) \tag{5.9}$$

which, by using Eqs. (5.7) and (5.8), can be rewritten as

$$E_{ij} = \alpha_{ij} + \theta^3\beta_{ij}, \tag{5.10}$$

where α_{ij} and β_{ij}, that are the non-zero components of the tensors, are related to the deformation of the shell as

$$\alpha_{ij} = \frac{1}{2}(\boldsymbol{a}_i \cdot \boldsymbol{a}_j - \bar{\boldsymbol{a}}_i \cdot \bar{\boldsymbol{a}}_j), \tag{5.11}$$

$$\beta_{\alpha\beta} = \boldsymbol{a}_\alpha \cdot \frac{\partial \boldsymbol{a}_3}{\partial \theta^\beta} - \bar{\boldsymbol{a}}_\alpha \cdot \frac{\partial \bar{\boldsymbol{a}}_3}{\partial \theta^\beta}. \tag{5.12}$$

The terms $\alpha_{\alpha\beta}$, the membrane strains, measure the strains of the surface, while the components $\alpha_{\alpha 3}$ measure the shearing of the director $\bar{\boldsymbol{a}}_3$, the component α_{33} measures the stretching of the director, and the components $\beta_{\alpha\beta}$, the bending strains, measure the bending or change in curvature of the shell. We remind that, according to the Kirchoff-Love theory of thin shells, the director $\boldsymbol{a}_3$ is constrained to coincide with the unit normal to the deformed middle surface of the shell ($\boldsymbol{a}_\alpha \cdot \boldsymbol{a}_3 = 0$, $|\boldsymbol{a}_3|$), and as consequence the shear strain $\alpha_{\alpha 3}$ vanish identically and the bending strains becomes

$$\beta_{\alpha\beta} = \frac{\partial \bar{\boldsymbol{a}}_\alpha}{\partial \theta^\beta} \cdot \bar{\boldsymbol{a}}_3 - \frac{\partial \boldsymbol{a}_\alpha}{\partial \theta^\beta} \cdot \boldsymbol{a}_3. \tag{5.13}$$

As already said, we confine our attention to linear theory of shell, and thus we begin by writing

$$\boldsymbol{x}(\theta^1, \theta^2) = \bar{\boldsymbol{x}}(\theta^1, \theta^2) + \boldsymbol{u}(\theta^1, \theta^2), \tag{5.14}$$

where $\boldsymbol{u}(\theta^1, \theta^2)$ is the displacement field of the middle surface of the shell. The membrane and bending strains then follow as

$$\alpha_{\alpha\beta} = \frac{1}{2}(\bar{\boldsymbol{a}}_\alpha \cdot \frac{\partial \boldsymbol{u}}{\partial \theta^\beta} + \frac{\partial \boldsymbol{u}}{\partial \theta^\alpha} \cdot \bar{\boldsymbol{a}}_\beta) \tag{5.15}$$

$$\beta_{\alpha\beta} = -\frac{\partial \boldsymbol{u}}{\partial \theta^\alpha \theta^\beta} \cdot \bar{\boldsymbol{a}}_3 + \frac{1}{\sqrt{\bar{a}}}\left[\frac{\partial \boldsymbol{u}}{\partial \theta^1} \cdot (\frac{\partial \bar{\boldsymbol{a}}_\alpha}{\partial \theta^\beta} \times \bar{\boldsymbol{a}}_2) + \frac{\partial \boldsymbol{u}}{\partial \theta^2} \cdot (\bar{\boldsymbol{a}}_1 \times \frac{\partial \bar{\boldsymbol{a}}_\alpha}{\partial \theta^\beta})\right] \tag{5.16}$$

$$+ \frac{\bar{\boldsymbol{a}}_3 \cdot \frac{\partial \bar{\boldsymbol{a}}_\alpha}{\partial \theta^\beta}}{\sqrt{\bar{a}}}[\frac{\partial \boldsymbol{u}}{\partial \theta^1} \cdot (\bar{\boldsymbol{a}}_2 \times \bar{\boldsymbol{a}}_3) + \frac{\partial \boldsymbol{u}}{\partial \theta^2} \cdot (\bar{\boldsymbol{a}}_3 \times \bar{\boldsymbol{a}}_1)]. \tag{5.17}$$

By means of the expressions (5.15) and (5.16), we conclude that the deformation of the shell is completely described by the displacement $\boldsymbol{u}$ of the middle surface that, as a consequence, in the following we consider as the unknown of the model. Moreover, in the context of the linearized theory, the undeformed and deformed domains are indistinguishable and so from now on we do not do any distinction between them.

5.1.1 Equilibrium configuration of elastic shells

The equilibrium configurations are obtained by applying the principle of virtual work which states that the sum of internal and external work vanish in the state of equilibrium. For the sake of simplicity, we assume that the shell is linear elastic. Thus, by introducing the Young's modulus E and the Poisson ratio ν, we define its strain energy density per unit area as ([KBLW09])

$$W(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{1}{2}\frac{Eh}{1-\nu^2} H^{\alpha\beta\gamma\delta}\alpha_{\alpha\beta}\alpha_{\gamma\delta} + \frac{1}{2}\frac{Eh^3}{12(1-\nu^2)} H^{\alpha\beta\gamma\delta}\beta_{\alpha\beta}\beta_{\gamma\delta} \tag{5.18}$$

where

$$H^{\alpha\beta\gamma\delta} = \nu \bar{a}^{\alpha\beta}\bar{a}^{\gamma\delta} + \frac{1}{2}(1-\nu)(\bar{a}^{\alpha\gamma}\bar{a}^{\beta\delta} + \bar{a}^{\alpha\delta}\bar{a}^{\beta\gamma}). \tag{5.19}$$

In (5.19), the first term is the membrane strain energy density and the second term is the bending strain energy density. The membrane stress resultant and the bending stress resultant are then defined as

$$n^{\alpha\beta} = \frac{\partial W}{\partial \alpha_{\alpha\beta}} = \frac{Eh}{1-\nu^2} H^{\alpha\beta\gamma\delta} \alpha_{\gamma\delta}, \tag{5.20}$$

$$m^{\alpha\beta} = \partial W \partial \beta_{\alpha\beta} = \frac{Eh^3}{12(1-\nu^2)} H^{\alpha\beta\gamma\delta} \beta_{\gamma\delta}, \tag{5.21}$$

respectively. The internal potential energy is a function of the middle-surface configuration, which can be written as an integral over the middle surface

$$\Pi_{int}[\boldsymbol{u}] = \int_\Omega W(\boldsymbol{\alpha}, \boldsymbol{\beta}) d\Omega \tag{5.22}$$

and the external potential energy results

$$\Pi_{ext}[\boldsymbol{u}] = -\int_\Omega \boldsymbol{q} \cdot \boldsymbol{u} d\Omega - \int_\Gamma \boldsymbol{N} \cdot \boldsymbol{u} ds, \tag{5.23}$$

where $\boldsymbol{q}$ are the distributed loads per unit area of Ω and $\boldsymbol{N}$ are the axial forces per unit length of Γ; in this manner, the total potential energy of the shell results

$$\Pi[\boldsymbol{u}] = \Pi_{int}[\boldsymbol{u}] + \Pi_{ext}[\boldsymbol{u}]. \tag{5.24}$$

The stable equilibrium configurations of the shell minimize the total potential energy:

$$\Pi[\boldsymbol{u}] = \inf_{\boldsymbol{v} \in V} \Pi[\boldsymbol{v}], \tag{5.25}$$

where V is the space of solutions consisting of all trial displacements fields $\boldsymbol{v}$ with finite energy $\Pi[\boldsymbol{v}]$. It is clear from the form of the potential energy, that V may be identified with the Sobolev space of functions in H^2.

The equilibrium condition must be fulfilled for any arbitrary variation of the displacement variables $\delta\boldsymbol{u}$, that in weak form is expressed as

$$\langle D\Pi[\boldsymbol{u}], \delta\boldsymbol{u} \rangle = \langle D\Pi_{int}[\boldsymbol{u}], \delta\boldsymbol{u} \rangle + \langle D\Pi_{ext}[\boldsymbol{u}], \delta\boldsymbol{u} \rangle = 0, \tag{5.26}$$

where $\langle D\Pi[\boldsymbol{u}], \delta\boldsymbol{u} \rangle$ denotes the first variation of Π at $\boldsymbol{u}$ in the direction of the virtual displacement $\delta\boldsymbol{u}$,

$$\langle D\Pi_{int}[\boldsymbol{u}], \delta\boldsymbol{u} \rangle = \int_\Omega [n^{\alpha\beta} \delta\alpha_{\alpha\beta} + m^{\alpha\beta} \delta\beta_{\alpha\beta}] d\Omega \tag{5.27}$$

is the internal work and

$$\langle D\Pi_{ext}[\boldsymbol{u}], \delta\boldsymbol{u} \rangle = -\int_\Omega \boldsymbol{q} \cdot \delta\boldsymbol{u} d\Omega - \int_\Gamma \boldsymbol{N} \cdot \delta\boldsymbol{u} ds \tag{5.28}$$

is the external virtual work.

5.2 Isogeometric discretization

We now proceed with the spatial discretization of the shell potential energy (5.24) by means of IGA ([KBLW09, COS00]) . By using of the Voigt's notation, we map the second order symmetric tensors into arrays

$$\boldsymbol{n} = \begin{pmatrix} n^{11} \\ n^{22} \\ n^{12} \end{pmatrix} \quad \boldsymbol{m} = \begin{pmatrix} m^{11} \\ m^{22} \\ m^{12} \end{pmatrix} \quad \boldsymbol{\alpha} = \begin{pmatrix} \alpha_{11} \\ \alpha_{22} \\ \alpha_{12} \end{pmatrix} \quad \boldsymbol{\beta} = \begin{pmatrix} \beta_{11} \\ \beta_{22} \\ \beta_{12} \end{pmatrix}$$

Eqs. (5.20) and (5.21) are similarly written as

$$\boldsymbol{n} = \frac{Eh}{1-\nu^2}\boldsymbol{H}\boldsymbol{\alpha} \tag{5.29}$$

$$\boldsymbol{m} = \frac{Eh}{12(1-\nu^2)}\boldsymbol{H}\boldsymbol{\beta} \tag{5.30}$$

where

$$\boldsymbol{H} = \begin{pmatrix} (\bar{a}^{11})^2 & \nu\bar{a}^{11} + (1-\nu)(\bar{a}^{12})^2 & \bar{a}^{11}\bar{a}^{12} \\ & (\bar{a}^{22})^2 & \bar{a}^{22}\bar{a}^{12} \\ \text{sym.} & & \frac{1}{2}[(1-\nu)\bar{a}^{11}\bar{a}^{22} + (1+\nu)(\bar{a}^{12})^2] \end{pmatrix},$$

which replaces (5.19) within the Voigt's formalism. Then, the internal virtual work (5.27) can be rewritten as

$$\langle \Pi_{int}[\boldsymbol{u}], \delta\boldsymbol{u} \rangle = \int_\Omega \left[\frac{Eh}{(1-\nu^2)} \delta\boldsymbol{\alpha}^T \boldsymbol{H}\boldsymbol{\alpha} + \frac{Eh^3}{12(1-\nu^2)} \delta\boldsymbol{\beta}^T \boldsymbol{H}\boldsymbol{\beta} \right] d\Omega \tag{5.31}$$

We now introduce a set of mesh elements $\{K_k\}_{k=1}^{N_e}$, that represents a partition of Ω, as already done in (1.46). In a discretized system with $\mathcal{N}$ degrees of freedom, the displacement vector can be expressed in the form

$$\boldsymbol{u}^{\mathcal{N}}(\theta^1, \theta^2) = \sum_{i=1}^{\mathcal{N}} R_i(\theta^1, \theta^2)\boldsymbol{u}_i, \tag{5.32}$$

where $\{\boldsymbol{u}_i,\ i = 1, ..., \mathcal{N}\}$ are the discrete nodal displacement vectors and $\{R_i,\ i = 1, ..., \mathcal{N}\}$ the corresponding NURBS basis functions, define in the parametric domain $\hat{\Omega}$ and ensuring the C^1 continuity between the elements K_k. By applying Eqs. (5.15) and (5.16) to (5.32) we obtain the membrane and bending strains in the forms

$$\boldsymbol{\alpha}_h(\theta^1, \theta^2) = \sum_{i=1}^{\mathcal{N}} \boldsymbol{M}_i(\theta^1, \theta^2)\boldsymbol{u}_i, \tag{5.33}$$

$$\boldsymbol{\beta}_h(\theta^1, \theta^2) = \sum_{i=1}^{\mathcal{N}} \boldsymbol{B}_i(\theta^1, \theta^2)\boldsymbol{u}_i, \tag{5.34}$$

respectively, where the matrices $\boldsymbol{M}_i$ and $\boldsymbol{B}_i$ are expressed as

$$\boldsymbol{M}_i = \begin{pmatrix} R_{i,1}\boldsymbol{a}_1 \cdot \boldsymbol{e}_1 & R_{i,1}\boldsymbol{a}_1 \cdot \boldsymbol{e}_2 & R_{i,1}\boldsymbol{a}_1 \cdot \boldsymbol{e}_3 \\ R_{i,2}\boldsymbol{a}_2 \cdot \boldsymbol{e}_1 & R_{i,2}\boldsymbol{a}_2 \cdot \boldsymbol{e}_2 & R_{i,2}\boldsymbol{a}_2 \cdot \boldsymbol{e}_3 \\ (R_{i,2}\boldsymbol{a}_1 + R_{i,1}\boldsymbol{a}_2) \cdot \boldsymbol{e}_1 & (R_{i,2}\boldsymbol{a}_1 + R_{i,1}\boldsymbol{a}_2) \cdot \boldsymbol{e}_2 & (R_{i,2}\boldsymbol{a}_1 + R_{i,1}\boldsymbol{a}_2) \cdot \boldsymbol{e}_3 \end{pmatrix},$$

$$\boldsymbol{B}_i = \begin{pmatrix} \boldsymbol{B}_{i1} \cdot \boldsymbol{e}_1 & \boldsymbol{B}_{i1} \cdot \boldsymbol{e}_2 & \boldsymbol{B}_{i1} \cdot \boldsymbol{e}_3 \\ \boldsymbol{B}_{i2} \cdot \boldsymbol{e}_1 & \boldsymbol{B}_{i2} \cdot \boldsymbol{e}_2 & \boldsymbol{B}_{i2} \cdot \boldsymbol{e}_3 \\ \boldsymbol{B}_{i3} \cdot \boldsymbol{e}_1 & \boldsymbol{B}_{i3} \cdot \boldsymbol{e}_2 & \boldsymbol{B}_{i3} \cdot \boldsymbol{e}_3 \end{pmatrix},$$

respectively. In the above expressions, $(\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3)$ are the basis vectors of an orthonormal coordinate reference frame, and

$$\boldsymbol{B}_{i1} = -\mathcal{R}_{i,11}\boldsymbol{a}_3 + \frac{1}{\sqrt{a}}[\mathcal{R}_{i,1}\boldsymbol{a}_{1,1} \times \boldsymbol{a}_2 + \mathcal{R}_{i,2}\boldsymbol{a}_1 \times \boldsymbol{a}_{1,1} \tag{5.35}$$

$$+ \boldsymbol{a}_3 \cdot \boldsymbol{a}_{1,1}(\mathcal{R}_{i,1}\boldsymbol{a}_2 \times \boldsymbol{a}_3 + \mathcal{R}_{i,2}\boldsymbol{a}_3 \times \boldsymbol{a}_3)] \tag{5.36}$$

$$\boldsymbol{B}_{i2} = -\mathcal{R}_{i,22}\boldsymbol{a}_3 + \frac{1}{\sqrt{a}}[\mathcal{R}_{i,1}\boldsymbol{a}_{2,2} \times \boldsymbol{a}_2 + \mathcal{R}_{i,2}\boldsymbol{a}_1 \times \boldsymbol{a}_{2,2} \tag{5.37}$$

$$+ \boldsymbol{a}_3 \cdot \boldsymbol{a}_{2,2}(\mathcal{R}_{i,1}\boldsymbol{a}_2 \times \boldsymbol{a}_3 + \mathcal{R}_{i,2}\boldsymbol{a}_3 \times \boldsymbol{a}_3)] \tag{5.38}$$

$$\boldsymbol{B}_{i3} = -\mathcal{R}_{i,12}\boldsymbol{a}_3 + \frac{1}{\sqrt{a}}[\mathcal{R}_{i,1}\boldsymbol{a}_{1,2} \times \boldsymbol{a}_2 + \mathcal{R}_{i,2}\boldsymbol{a}_1 \times \boldsymbol{a}_{1,2} \tag{5.39}$$

$$+ \boldsymbol{a}_3 \cdot \boldsymbol{a}_{1,2}(\mathcal{R}_{i,1}\boldsymbol{a}_2 \times \boldsymbol{a}_3 + \mathcal{R}_{i,2}\boldsymbol{a}_3 \times \boldsymbol{a}_3)] \tag{5.40}$$

where we used the comma to indicate partial differentiation. Finally, by substituting Eqs. (5.32)-(5.34) into the formulation of the principle of virtual work in Eq. (5.26) we obtain the equation of equilibrium for the nodal displacements:

$$\boldsymbol{K}\boldsymbol{u} = \boldsymbol{f}, \tag{5.41}$$

where

$$\boldsymbol{K} = \sum_{k=1}^{N_e} \int_{K_k} \left[\frac{Eh}{1-\nu^2} (\boldsymbol{M}_i)^T \boldsymbol{H} \boldsymbol{M}_j + \frac{Eh^3}{12(1-\nu^2)} (\boldsymbol{B}_i)^T \boldsymbol{H} \boldsymbol{B}_j \right] d\Omega \equiv \sum_{k=1}^{N_e} \boldsymbol{K}_{kij} \tag{5.42}$$

is the stiffness matrix and

$$\boldsymbol{f}_i = \sum_{k=1}^{N_e} \left\{ \int_{K_k} \boldsymbol{q}\mathcal{R}_i d\Omega + \int_{\partial(K_k)\cup\Gamma} \boldsymbol{N}\mathcal{R}_i ds \right\} \equiv \sum_{k=1}^{N_e} \boldsymbol{f}_{ki} \tag{5.43}$$

is the the right hand side that represents the nodal force vector. These integrals may be efficiently evaluated by means of suitable numerical quadrature rules (e.g. Gauss-Legendre, see Section 1.4.4).

5.3 MDEIM and RB approximation of Kirchoff-Love model

In this section we are interested in solving the Kirchoff-Love model defined on parametrized computational domain, i.e. a parametrized shell. In order to efficiently solve the problem

for each parameter $\boldsymbol{\mu}$ of interest, we provide a MDEIM-RB approximation of the problem, following the procedure developed in Chapter 4.

We first introduce a parametric version of system (5.41) as follows

$$\boldsymbol{K}(\boldsymbol{\mu})\boldsymbol{u}(\boldsymbol{\mu}) = \boldsymbol{f}(\boldsymbol{\mu}). \tag{5.44}$$

Then, by following a procedure similar to the one adopted in Section 4.4, we can directly use MDEIM on the matrix $\boldsymbol{K}(\boldsymbol{\mu})$ and right-hand side $\boldsymbol{f}(\boldsymbol{\mu})$ to obtain an affine approximation of (5.44) as

$$\boldsymbol{K}_M(\boldsymbol{\mu})\boldsymbol{u}_M(\boldsymbol{\mu}) = \boldsymbol{f}_M(\boldsymbol{\mu}), \tag{5.45}$$

that can be explicited as

$$\left(\sum_{m=1}^{M_k} \Theta_m^k(\boldsymbol{\mu})\boldsymbol{K}^m\right)\boldsymbol{u}_M(\boldsymbol{\mu}) = \sum_{m=1}^{M_f} \Theta_m^f(\boldsymbol{\mu})\boldsymbol{f}^m. \tag{5.46}$$

Finally, we combine MDEIM with the RB technique as in Section 4.5. With this aim, we compute a set of basis by means of a POD or greedy approach (see Sections 2.4.2 and 2.4.3) and obtain the reduced system

$$\boldsymbol{K}_{N,M}(\boldsymbol{\mu})\boldsymbol{u}_{N,M}(\boldsymbol{\mu}) = \boldsymbol{f}_{N,M}(\boldsymbol{\mu}), \tag{5.47}$$

that can be expressed as

$$\left(\sum_{m=1}^{M_k} \Theta_m^k(\boldsymbol{\mu})\boldsymbol{K}_N^m\right)\boldsymbol{u}_{N,M}(\boldsymbol{\mu}) = \sum_{m=1}^{M_f} \Theta_m^f(\boldsymbol{\mu})\boldsymbol{f}_N^m. \tag{5.48}$$

5.4 Numerical Examples

In the following the proposed method is shown for some standard numerical benchmark tests, namely the three problems of the *shell course* ([BSL$^+$85, BBWR04]): the pinched cylinder, the pinched hemisphere, and the Scordelis-Lo roof. These problems have been extensively discussed and treated in literature for the assessment of shell analysis, both in a FE context ([SFR89, ACdSAF03, BB93]) and in a IGA context ([KBLW09, CHB09]). For each of the example presented, we first compute the numerical solution by means of the IGA discretization, then we introduce a parametrization of the geometries of interest and solve the problems by adopting the RB approach.

5.4.1 Pinched cylinder with diaphragms

The first problem we consider concerns a cylindrical shell pinched under the action of two diametrically opposite unit loads located within the middle section of the shell ([BSL$^+$85]). We consider the case in which the cylinder is supported by rigid diaphragms at its ends. The length of the cylinder is $L = 600$, the radius is $R = 300$, the thickness is $h = 3$, the Young's modulus is $E = 3\times10^6$, and the Poisson ratio is $\nu = 0.3$. Due to the symmetry of the problem, only one octant of the cylinder is used in the calculation (Fig. 5.2); dimensionless data are considered. The reference solution $u_{ref} = 1.8248 \times 10^{-5}$ is given as the radial displacement at the point of the applied load (red circle in Fig. 5.2) ([BSL$^+$85]).

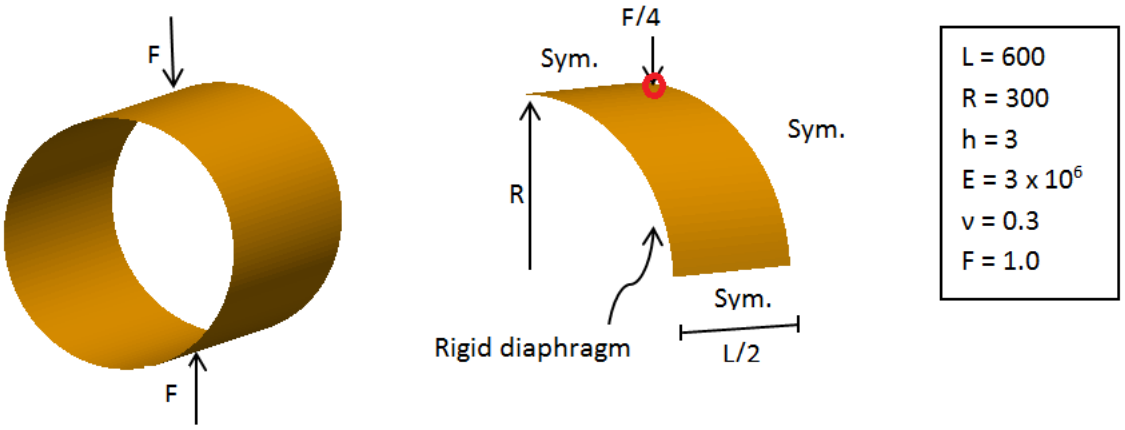


Figure 5.2: Definition of the pinched cylinder problem.

In Fig. 5.3 we report the convergence of the displacement u_{mon} under the applied load, that is our monitored quantity, for different meshes and polynomial degrees. The monitored displacement converges to the reference solution. From now on, in our calculation, we use polynomial degrees $p = 3$ and 35 control points in both directions, yielding $\mathcal{N} = 3675$ degrees of freedom and 1024 elements. In Fig. 5.4 we report the mesh for the octant of the cylinder built with these data and the correspondent numerical solution that is symmetrically extended for visualization purpose.

We are now interested in solving the same problem for different configurations of the cylinder, namely geometrical parametrizations. In particular, we aim to modify the radius of middle section of the shell. To this end, we introduce a geometrical parameter $\mu \in [-100, 100]$ such that the radius of the middle section of the parameter dependent geometry is $\tilde{R} = R + \mu$. In this way, the problem is still symmetric and we can study only one octant of the geometry as previously done. We underline that, in order to ensure the smoothness of the surface when changing the middle radius and the fulfillment of the symmetry conditions we built the parametrized cylinder with degree 2 in both directions and set the control points as in Table 5.1. Then, for the analysis, we performed a k-refinement to obtain degree $p = 3$ and 35 control points per direction. In Fig. 5.5 we report the meshes corresponding to the parameters $\mu = 0, 50, 100, -50, -100$. In Fig. 5.6 we plot the monitored displacement $u_{mon}(\mu)$ obtained for $\mu \in [-100, 100]$. We observe that the monitored displacement is maximum for $\mu = -9.09$, for which is $u_{mon}(\mu) = 1.9228 \cdot 10^{-5}$, and is minimum for $\mu = 100$, for which $u_{mon}(\mu) = 8.6762 \cdot 10^{-6}$. The evaluation of each of these solutions is obtained by solving the IGA system (5.44) and each evaluation requires about 4.62 s.

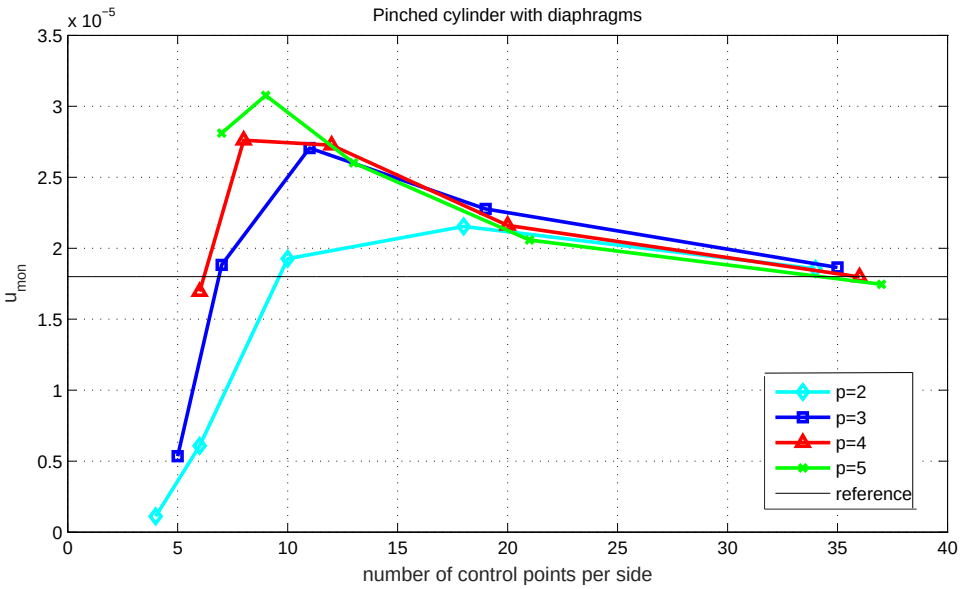


Figure 5.3: Pinched cylinder displacement convergence.

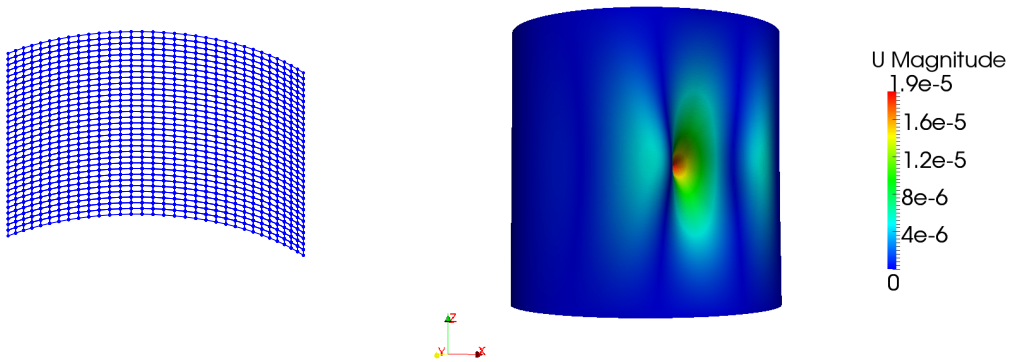
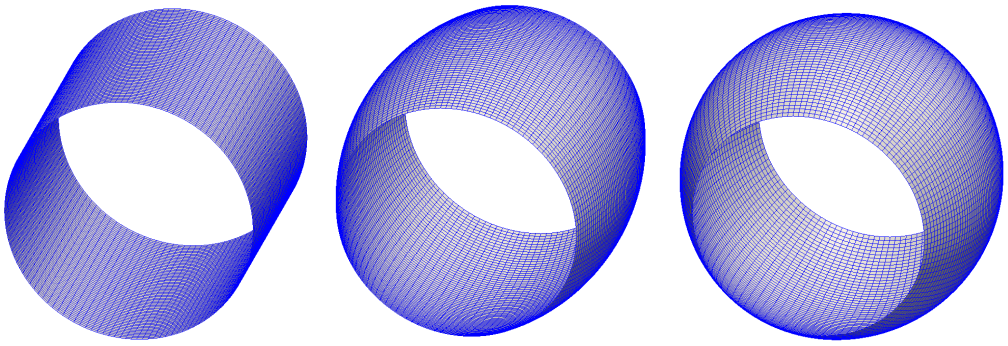


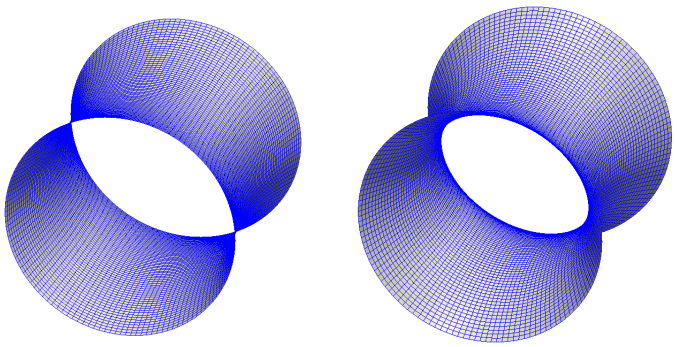
Figure 5.4: Pinched cylinder. One octant of the mesh (left) and numerical solution (right) for polynomial degree $p = 3$ and $\mathcal{N} = 3675$ dofs.

Table 5.1: *Control points of the parametrized pinched cylinder. In the following $z = L/2$.*

$\boldsymbol{B}_{i,j}$	$w_{i,j}$
$\boldsymbol{B}_{1,1} = (R, 0, 0)$	$w_{1,1} = 1$
$\boldsymbol{B}_{1,2} = (R, R, 0)$	$w_{1,2} = \sqrt{2}/2$
$\boldsymbol{B}_{1,3} = (0, R, 0)$	$w_{1,3} = 1$
$\boldsymbol{B}_{2,1} = (R + \mu, 0, z/2)$	$w_{2,1} = 1$
$\boldsymbol{B}_{2,2} = (R + \mu, R + \mu, z/2)$	$w_{2,2} = \sqrt{(2)}/2$
$\boldsymbol{B}_{2,3} = (0, R + \mu, z/2)$	$w_{2,3} = 1$
$\boldsymbol{B}_{3,1} = (R + \mu, 0, z)$	$w_{3,1} = 1$
$\boldsymbol{B}_{3,2} = (R + \mu, R + \mu, z)$	$w_{3,2} = \sqrt{2}/2$
$\boldsymbol{B}_{3,3} = (0, R + \mu, z)$	$w_{3,3} = 1$



$\mu = 0$ $\mu = 50$ $\mu = 100$



$\mu = -50$ $\mu = -100$

Figure 5.5: Pinched cylinder. Meshes for polynomial degree $p = 3$ and 1024 elements for the parameters $\mu = 0, 50, 100, -50, -100$ (the meshes are symmetrically extended for visualization purposes).

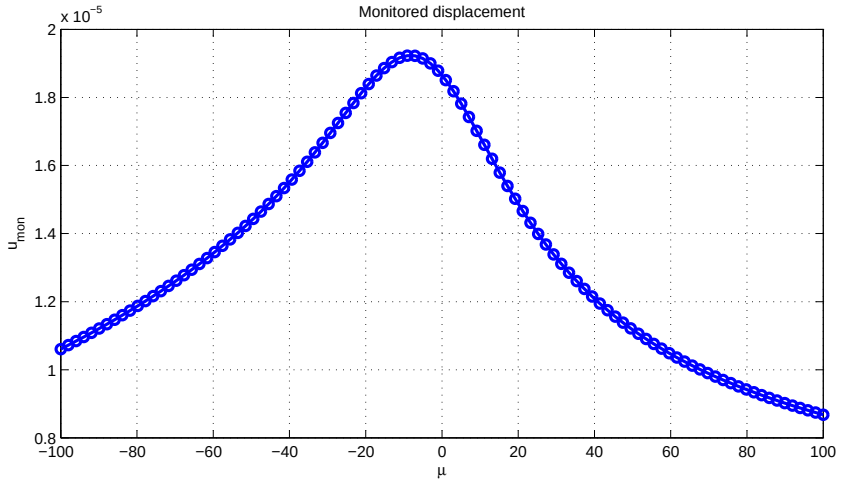


Figure 5.6: Pinched cylinder. Monitored displacement $u_{mon}(\mu)$ for $\mu \in [-100, 100]$.

To solve the problem in a more efficient way, we perform, as anticipated, a MDEIM-RB approximation. To perform MDEIM we use a tolerance $\text{tol}_{MDEIM} = 10^{-6}$ and a set of 100 parameters samples. In Fig. 5.7 we report the resulting singular values and conclude that MDEIM entails $M_k = 10$ terms for the stiffness matrix and $M_f = 1$ term for the right-hand side vector. Moreover, the MDEIM procedure selects 88 elements (out of 1024) for the reduced mesh.

Then, we proceed with the RB approximation. We apply the greedy algorithm with tolerance $\text{tol} = 10^{-6}$ over a 150 parameters samples set. The greedy algorithm provides as output a reduced space of dimension $N_{max} = 20$. In Fig. 5.8 we report the numerical solutions for the IGA, the MDEIM, and the MDEIM-RB problems for the parameters $\mu = 0, 50, 100, -50, -100$. The surfaces are overlapped to the cylinder to highlight the difference between the undeformed and the deformed configurations. In Table 5.2 we report the data of the problem, the results provided by MDEIM and greedy algorithm, and the computational times requested to solve the problem at hand. We conclude that the online evaluation of the solution of the MDEIM-RB problem is about 10 times faster than IGA one. Finally, in Fig. 5.9 we report and compare the average relative errors between the IGA and the MDEIM-RB solutions vs N, for $1 \leq N \leq N_{max}$ for different values of M_k, computed on a 400 samples parameter set Ξ_{test}. It is clear that the RB approximation is more accurate if we consider the maximum number of affine terms selected by MDEIM.

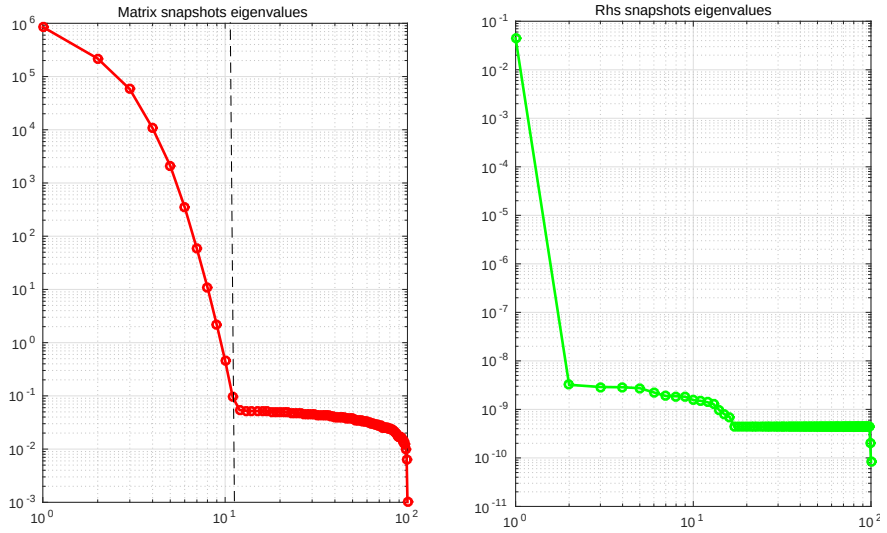
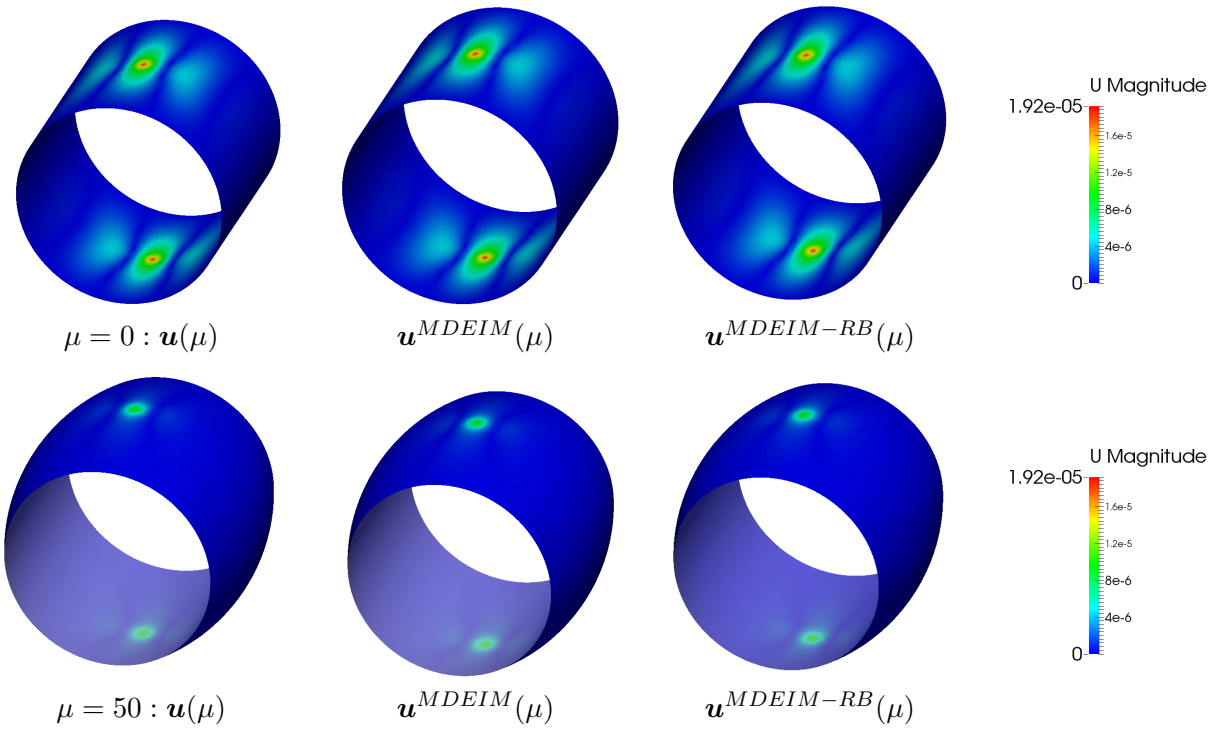
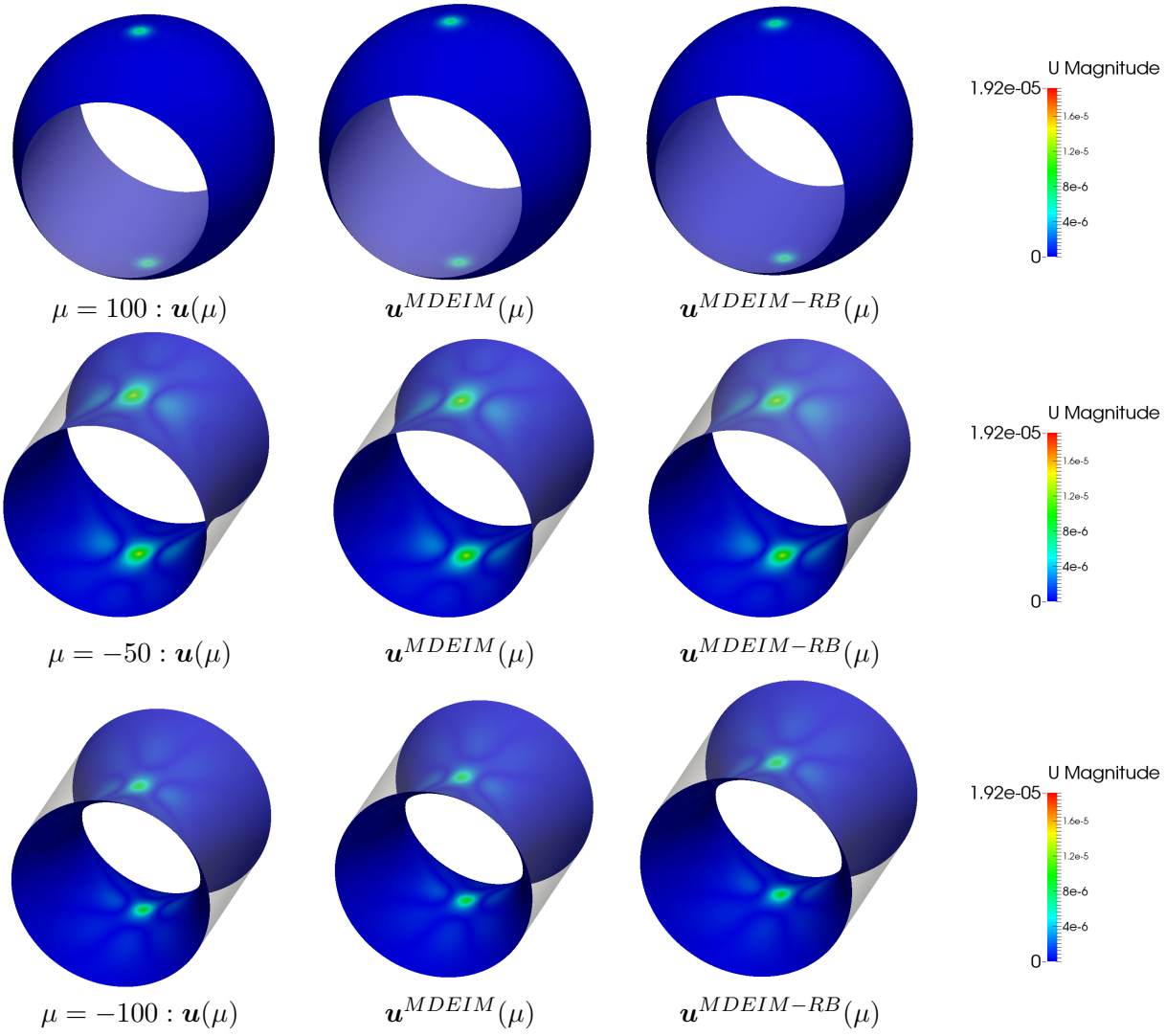


Figure 5.7: Pinched cylinder. Singular values of the snapshots matrices of the vectorized stiffness matrices and right hand side vectors for polynomial degrees 3.

Figure 5.8: Parametrized pinched cylinder for $\mu = 0, 50, 100, -50, -100$.





$\mu = 100 : \boldsymbol{u}(\mu)$ $\boldsymbol{u}^{MDEIM}(\mu)$ $\boldsymbol{u}^{MDEIM-RB}(\mu)$

$\mu = -50 : \boldsymbol{u}(\mu)$ $\boldsymbol{u}^{MDEIM}(\mu)$ $\boldsymbol{u}^{MDEIM-RB}(\mu)$

$\mu = -100 : \boldsymbol{u}(\mu)$ $\boldsymbol{u}^{MDEIM}(\mu)$ $\boldsymbol{u}^{MDEIM-RB}(\mu)$

Table 5.2: *Pinched cylinder. Data of the problems, MDEIM and the greedy algorithm, and computational times requested offline and online.*

IGA	
NURBS degree	3
Elements	1024
$\mathcal{N}$	3675
t_{IGA}	4.62 s
MDEIM	
Number of parameters	1
Number of samples	100
tol	10^{-6}
Affine matrix components $\boldsymbol{K}^m$	11
Affine rhs components $\boldsymbol{f}^m$	1
Reduced elements	88
Greedy-RB	
Greedy tolerance	10^{-6}
Sample train (greedy)	150
Greedy space dimension N	20
Greedy construction time (offline)	9.40 min
$t_{greedy-RB}$ (online)	0.45 s
$t_{IGA}/t_{greedy-RB}$	10.27

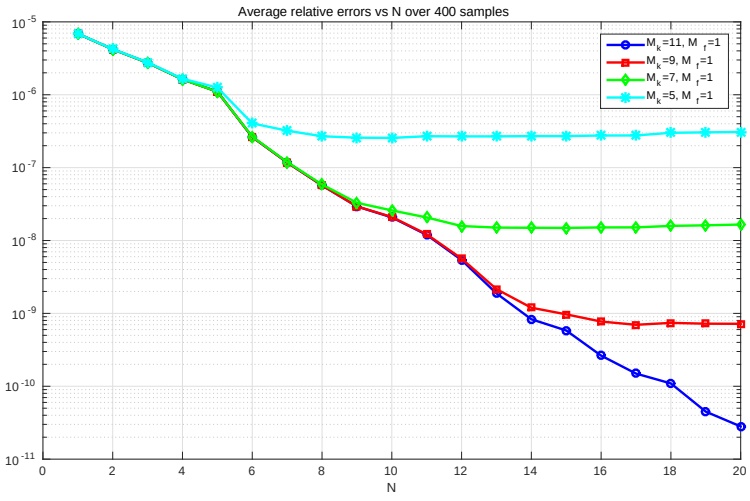


Figure 5.9: Pinched cylinder. Average relative error between IGA and MDEIM-greedy-RB solutions vs N, for $1 \leq N \leq N_{max}$ for different values of M_k, computed on a 400 samples parameter set Ξ_{test}.

5.4.2 Pinched hemisphere

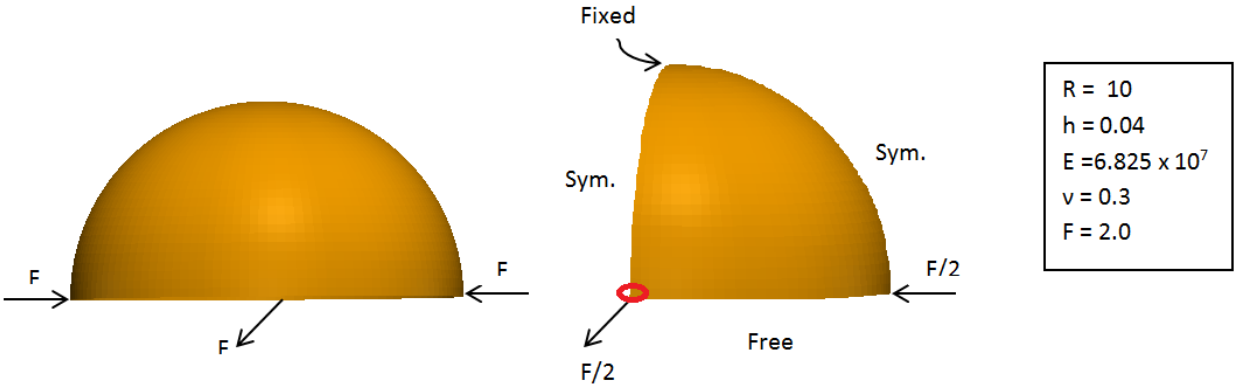


Figure 5.10: Definition of the pinched hemisphere problem.

In the pinched hemisphere, equal and opposite concentrated forces are applied at antipodal points of the equator ([BSL+85]). The equator is otherwise considered to be free. The radius of the sphere is $R = 10$, the thickness is $h = 0.04$, the Young's modulus is $E = 6.825 \times 10^7$, and the Poisson's ratio is $\nu = 0.3$. Due to symmetry, only one quart of the hemispherical shell is used in the calculation (Fig. 5.10). The hemispherical shape is an example of the geometries that we can build simply and exactly with NURBS but not in a classical FE setting.

In this case, as reference solution, we consider the radial displacement under one of the points of the applied loads (red circle in Fig. 5.10) that is equal to $u_{ref} = 0.0924$ ([BSL+85]). Convergence of the monitored displacement u_{mon} at this point is presented in Fig. 5.11. The solutions converge to u_{ref}. From now on, in our calculation, we use polynomial degrees $p = 3$ and 35 control points in both directions, yielding $\mathcal{N} = 3675$ degrees of freedom and 1024 elements. In Fig. 5.12 we report the mesh for the quarter of the hemispherical shell, while the solution is symmetrically extended to the entire shell for visualization purposes.

We now solve the same problem for different configurations of the hemisphere. To this end, we define a parametrized geometry depending on a parameter $\mu \in [-5, 5]$, whose control points and weights are reported in 5.3. This NURBS geometry is built with degrees $p = 2$ for the two curved directions. We observe that for $\mu = 0$ we obtain the quarter of hemisphere. With this parametrization, the problem is still symmetric and we can study only one quadrant of the geometry as previously done. In particular, for the analysis we perform a k-refinement to obtain $p = 3$ and 35 control points in both directions. In Fig. 5.13 we report the mesh built for the parameters $\mu = 0, 5, -5$, while in Fig. 5.14 we plot the monitored displacement for each $\mu \in [-5, 5]$. We observe that the monitored displacement has a monotone behavior and assumes maximum value for $\mu = 5$, being equal to $u_{mon}(\mu) = 0.1164$, and minimum value for $\mu = -5$, for which $u_{mon}(\mu) = 0.032$. The evaluation of each of these solutions is obtained by solving the IGA system (5.44) and each evaluation requires about 4.82 s.

As previously done, in order to efficiently solve the problem for each parameter μ, we reduce its dimension by means of a MDEIM-RB approximation. To perform MDEIM we use a tolerance $\text{tol}_{MDEIM} = 10^{-6}$ and a set of 100 parameters samples. In Fig. 5.15 we report the

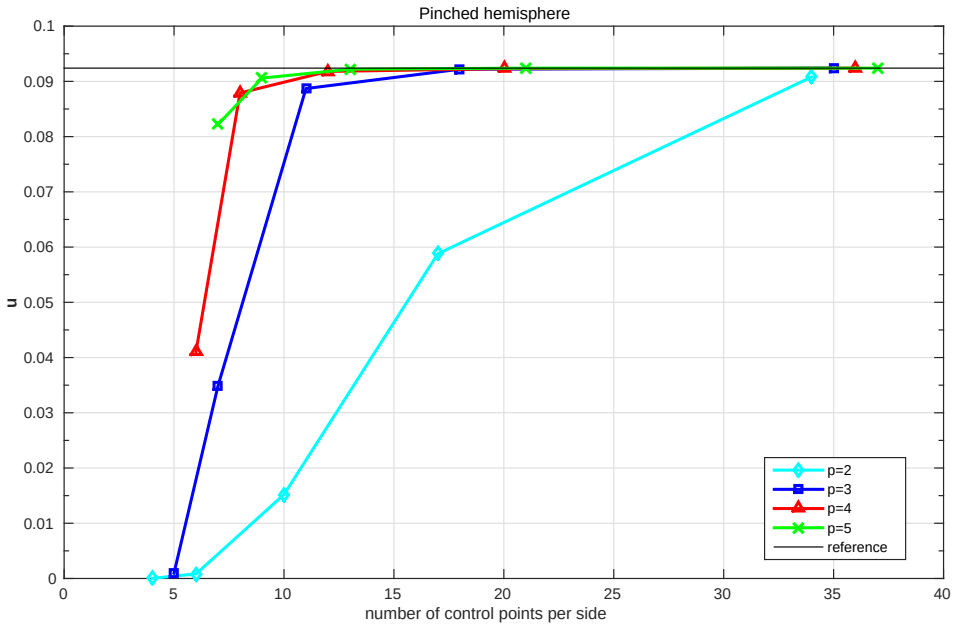


Figure 5.11: Pinched hemisphere displacement convergence.

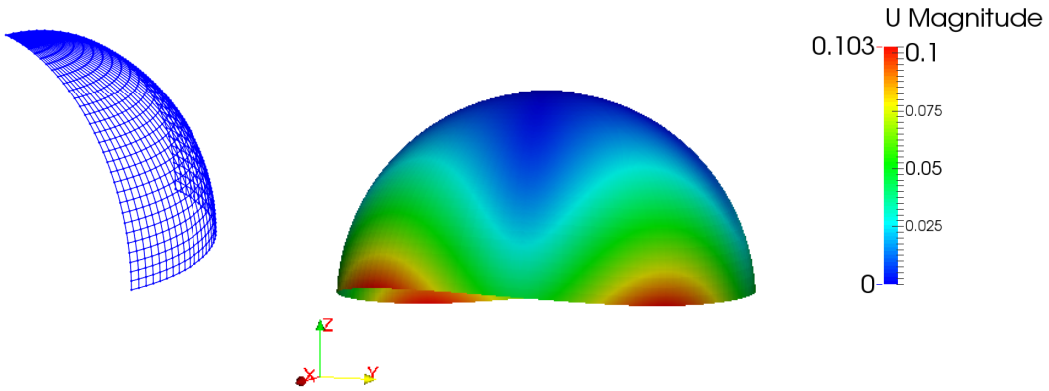


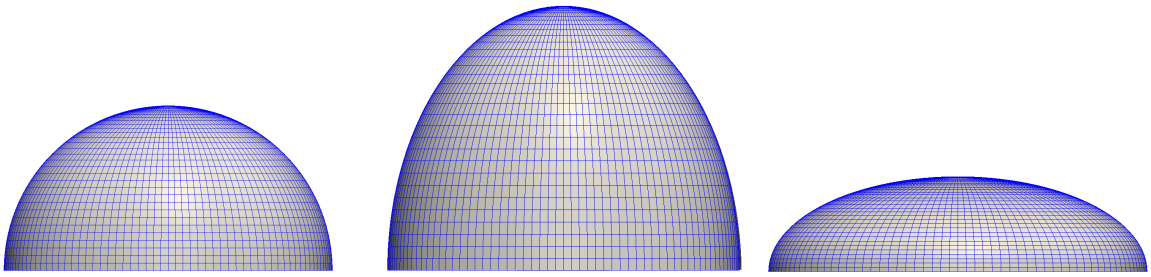
Figure 5.12: Pinched hemisphere. One quadrant of the mesh (left) and numerical solution (right) for polynomial degree $p = 3$ and $\mathcal{N} = 3675$ dofs.

resulting singular values and conclude that MDEIM entails $M_k = 15$ terms for the stiffness matrix and $M_f = 1$ term for the right-hand side vector. Moreover, the MDEIM procedure selects 117 elements (out of 1024) for the reduced mesh.

Then, we proceed with the RB approximation. We apply the greedy algorithm with tolerance tol $= 10^{-6}$ over a 150 parameters samples set. The greedy algorithm provides as output a reduced space of dimension $N_{max} = 9$. In Fig. 5.16 we report the numerical solutions for the IGA, the MDEIM, and the MDEIM-RB problems for the parameters $\mu = 0, 5, -5$.

Table 5.3: *Control points of the parametrized hemispherical shell.*

$\boldsymbol{B}_{i,j}$	$w_{i,j}$
$\boldsymbol{B}_{1,1} = (R, 0, 0)$	$w_{1,1} = 1$
$\boldsymbol{B}_{1,2} = (R, R, 0)$	$w_{1,2} = \sqrt{2}/2$
$\boldsymbol{B}_{1,3} = (0, R, 0)$	$w_{1,3} = 1$
$\boldsymbol{B}_{2,1} = (R, 0, R + \mu)$	$w_{2,1} = \sqrt{2}/2$
$\boldsymbol{B}_{2,2} = (R, R, R + \mu)$	$w_{2,2} = 1/2$
$\boldsymbol{B}_{2,3} = (0, R, R + \mu)$	$w_{2,3} = \sqrt{2}/2$
$\boldsymbol{B}_{3,1} = (0, 0, R + \mu)$	$w_{3,1} = 1$
$\boldsymbol{B}_{3,2} = (0, 0, R + \mu)$	$w_{3,2} = \sqrt{2}/2$
$\boldsymbol{B}_{3,3} = (0, 0, R + \mu)$	$w_{3,3} = 1$



$\mu = 0$ $\mu = 5$ $\mu = -5$

Figure 5.13: Pinched hemisphere. Meshes for polynomial degree $p = 3$ and 1024 elements for the parameters $\mu = 0, 5, -5$ (the meshes are symmetrically extended for visualization purposes).

The solutions are reported on a same scale, assuming values between the minimum and the maximum value of $u_{mon}(\mu)$ for $\mu \in [-5, 5]$, and the surfaces are overlapped to the hemisphere in order to highlight the difference between the undeformed and the deformed configurations. In Table 5.4 we report the data of the problem, the results provided by MDEIM and greedy algorithm, and the computational times requested to solve the problem at hand. We conclude that the online evaluation of the solution of the MDEIM-RB problem is about 9 times faster than IGA one. Finally, in Fig. 5.17 we report and compare the average relative errors between the IGA and the MDEIM-RB solutions vs N, for $1 \leq N \leq N_{max}$ for different values of M_k, computed on a 400 samples parameter set Ξ_{test}. It is clear that the RB approximation is more accurate if we consider the maximum number of affine terms selected by MDEIM.

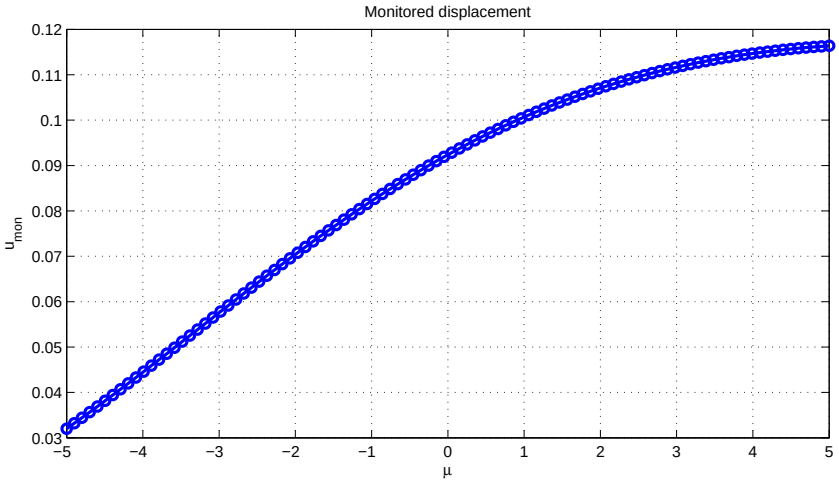


Figure 5.14: Pinched hemisphere. Monitored displacement $u_{mon}(\mu)$ for $\mu \in [-5, 5]$.

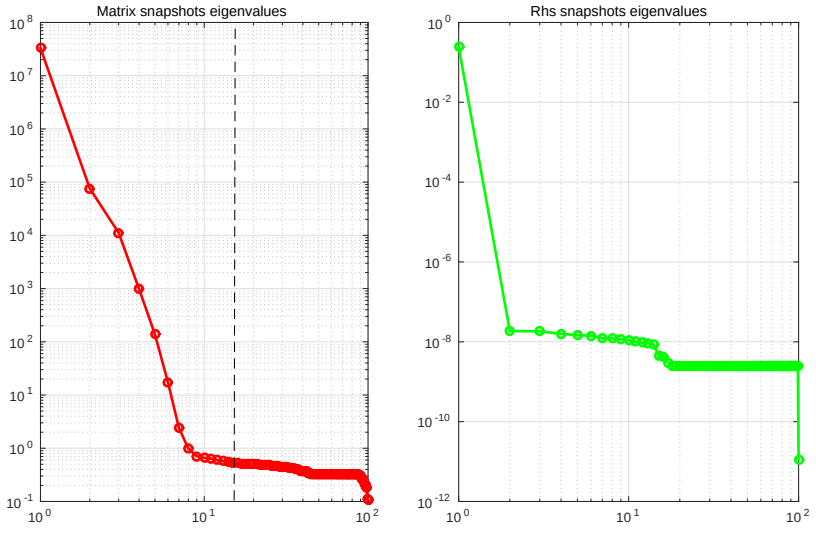


Figure 5.15: Pinched hemisphere. Singular values of the snapshots matrices of the vectorized stiffness matrices and right hand side vectors for polynomial degrees 3.

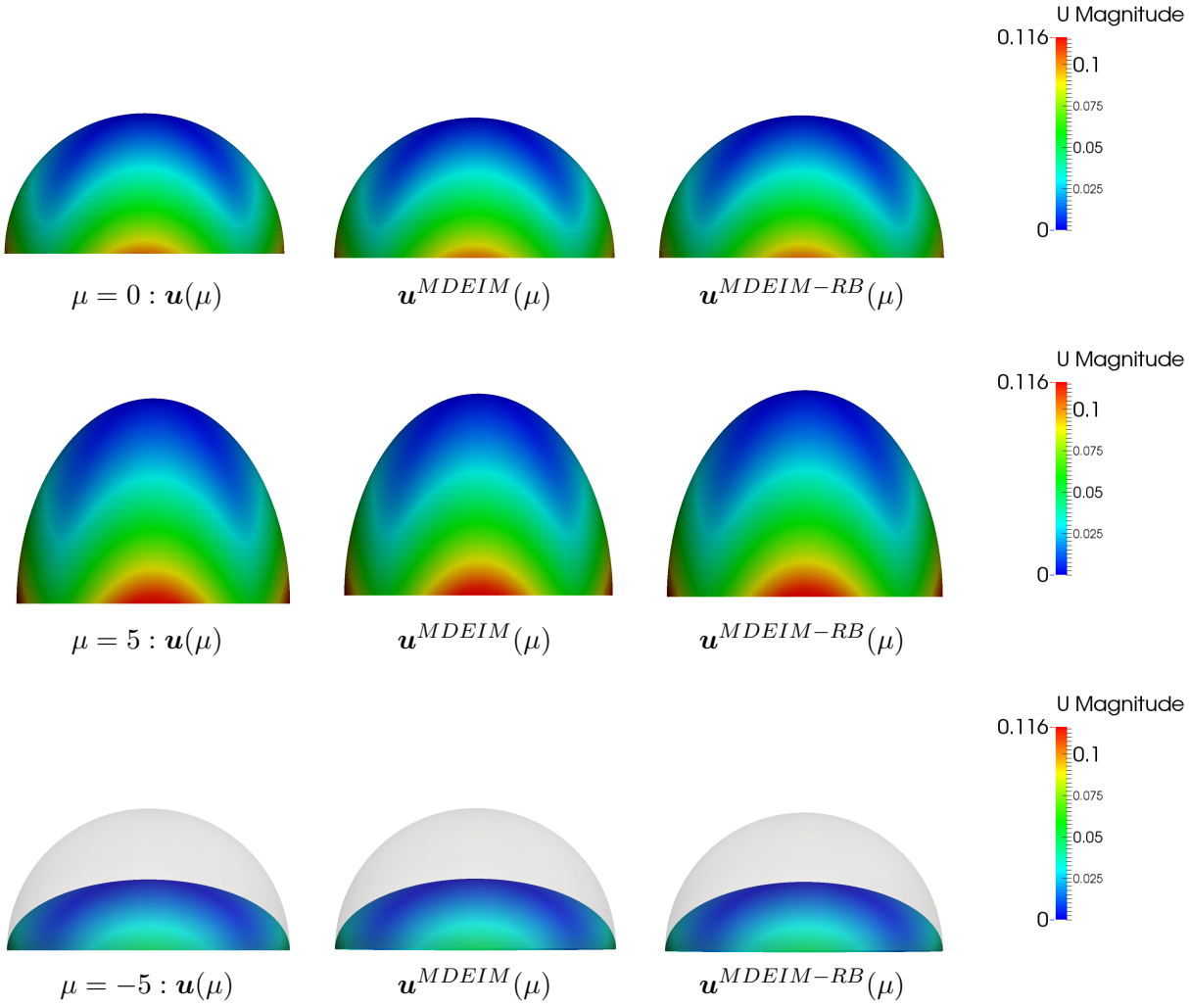


Figure 5.16: Parametrized pinched hemisphere for $\mu = 0, 5, -5$.

Table 5.4: *Pinched hemisphere. Data of the problems, MDEIM and the greedy algorithm, and computational times requested offline and online.*

IGA	
NURBS degree	3
Elements	1024
$\mathcal{N}$	3675
t_{IGA}	4.82 s
MDEIM	
Number of parameters	1
Number of samples	100
tol	10^{-6}
Affine matrix components $\boldsymbol{K}^m$	15
Affine rhs components $\boldsymbol{f}^m$	1
Reduced elements	117
Greedy-RB	
Greedy tolerance	10^{-6}
Sample train (greedy)	150
Greedy space dimension N	9
Greedy construction time (offline)	4.60 min
$t_{greedy-RB}$ (online)	0.54 s
$t_{IGA}/t_{greedy-RB}$	8.92

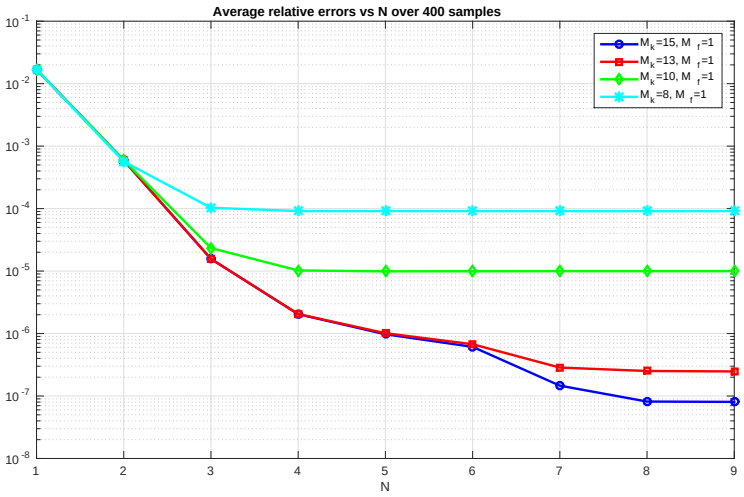


Figure 5.17: Pinched hemisphere. Average relative error between IGA and MDEIM-RB solutions vs N, for $1 \leq N \leq N_{max}$ for different values of M_k, computed on a 400 samples parameter set Ξ_{test}.

5.4.3 Scordelis-Lo roof

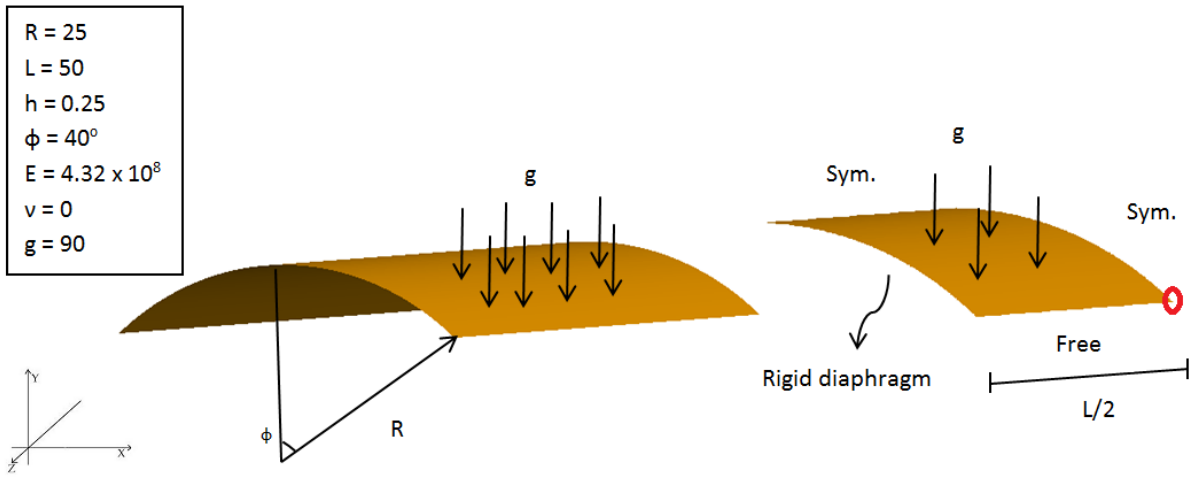


Figure 5.18: Definition of the Scordelis-Lo roof problem.

The Scordelis-Lo Roof is a section of a cylindrical shell. The displacement of the roof is imposed to be null in the x and y directions at its ends, whereas the side edges are free (Fig. 5.18). The shell is subjected to a uniform gravity load ([BSL$^+$85, MH85]). In our calculation, the length of the cylinder is $L = 50$, the radius is $R = 25$, the thickness is $h = 0.25$, and the angle subtended by the roof is $2\Phi = 80^o$; the Young's modulus is $E = 4.32 \times 10^8$ and the Poisson ratio is $\nu = 0$. Due to symmetry, we model only one quarter of the geometry.

In this case, the vertical displacement at the midpoint of the side edge (red circle in Fig. 5.18) is given as the reference solution and the reference value is $u_{ref} = 0.3024$ ([BSL$^+$85, MH85]). Convergence of the displacement at this location is presented in Fig. 5.19. The monitored displacements converge to $u_{mon} = 0.3000$ which is slightly lower than the reference one. In the following calculations we always consider polynomial degree $p = 3$ and 35 control points in both directions yielding $\mathcal{N} = 3675$ degrees of freedom and 1024 elements. In Fig. 5.20 we report the mesh for the quarter of the roof shell, while the solution is symmetrically extended to the entire shell for visualization purposes.

We are now interested in solving the same problem for different configurations of the roof. To this end, we introduce a geometrical parameter $\mu \in [0, 10]$ and impose a rigid displacement of the central section of the cylinder. In this way, the problem is still symmetric and we can study only one quadrant of the geometry as previously done. In particular, the geometry is built by means of NURBS of degree $p = 2$ in both directions and control points and weights reported in Table 5.5. In Fig. 5.21 we report the mesh built by means of polynomials of degrees $p = 3$ and 35 control points, obtained by realizing a suitable k-refinement, in both directions for the parameters $\mu = 0, 5, 10$, while in Fig. 5.22 we plot the monitored displacement for each $\mu \in [0, 10]$. We observe that the monitored displacement has a monotone behavior and is maximum for $\mu = 10$, for which $u_{mon}(\mu) = 4.2128$, and minimum for $\mu = 0$ assuming the value $u_{mon}(\mu) = 0.3$. The evaluation of each of these solutions is obtained by solving the IGA system (5.44) and each evaluation requires about 3.79 s.

As previously done, in order to efficiently solve the problem for each parameter μ, we

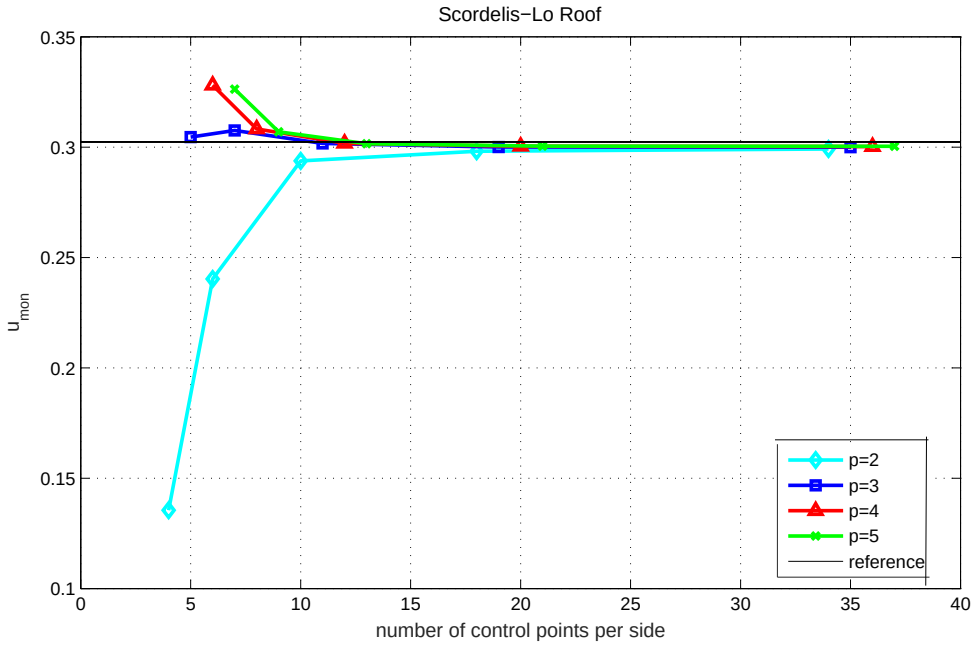


Figure 5.19: Scordelis-Lo roof displacement convergence.

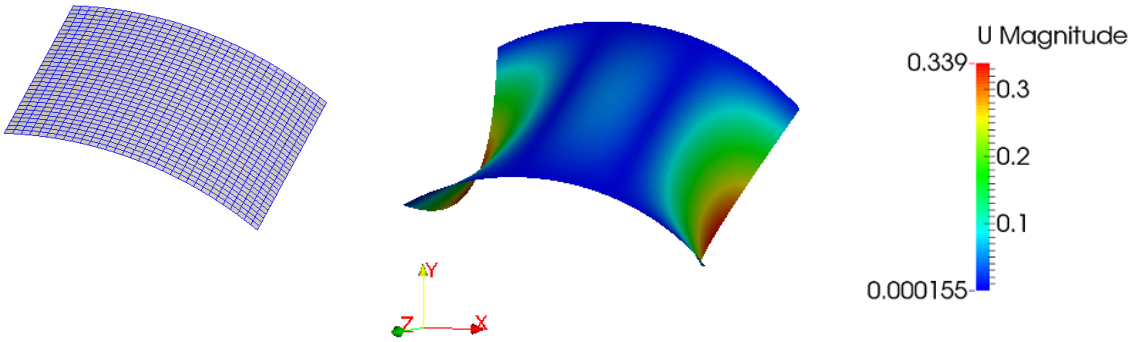


Figure 5.20: Scordelis-Lo roof. One quadrant of the mesh (left) and numerical solution (right) for polynomial degree $p = 3$ and $\mathcal{N} = 3675$ dofs.

reduce its dimension by means of a MDEIM-RB approximation. To perform MDEIM we use a tolerance $\text{tol}_{MDEIM} = 10^{-6}$ and a set of 100 parameters samples. In Fig. 5.23 we report the resulting singular values and conclude that MDEIM entails $M_k = 10$ terms for the stiffness matrix and $M_f = 5$ terms for the right-hand side vector. Moreover, the MDEIM procedure selects 107 elements (out of 1024) for the reduced mesh.

Then, we proceed with the RB approximation. We apply the greedy algorithm with tolerance $\text{tol} = 10^{-6}$ over a 150 parameters samples set. The greedy algorithm provides as output a reduced space of dimension $N_{max} = 10$. In Fig. 5.24 we report the numerical solutions for the IGA, the MDEIM, and the MDEIM-RB problems for the parameters $\mu =$

Table 5.5: *Control points of the parametrized Scordelis-Lo roof. In the following* $b = \sqrt{R^2 + R^2 \tan(\Phi/2)^2}$

$\boldsymbol{B}_{i,j}$	$w_{i,j}$
$\boldsymbol{B}_{1,1} = (R\cos(90-\Phi), R\sin(90-\Phi), 0)$	$w_{1,1} = 1$
$\boldsymbol{B}_{1,2} = (b\cos(70), b\sin(70), 0)$	$w_{1,2} = \cos(\Phi/2)$
$\boldsymbol{B}_{1,3} = (0, R, 0)$	$w_{1,3} = 1$
$\boldsymbol{B}_{2,1} = (R\cos(90-\Phi), R\sin(90-\Phi)+\mu, L/2)$	$w_{2,1} = 1$
$\boldsymbol{B}_{2,2} = (b\cos(70), b\sin(70)+\mu, L/2)$	$w_{2,2} = \cos(\Phi/2)$
$\boldsymbol{B}_{2,3} = (0, R+\mu, L/2)$	$w_{2,3} = 1$
$\boldsymbol{B}_{3,1} = (R\cos(90-\Phi), R\sin(90-\Phi)+\mu, L)$	$w_{3,1} = 1$
$\boldsymbol{B}_{3,2} = (b\cos(70), b\sin(70)+\mu, L)$	$w_{3,2} = \cos(\Phi/2)$
$\boldsymbol{B}_{3,3} = (0, R+\mu, L)$	$w_{3,3} = 1$

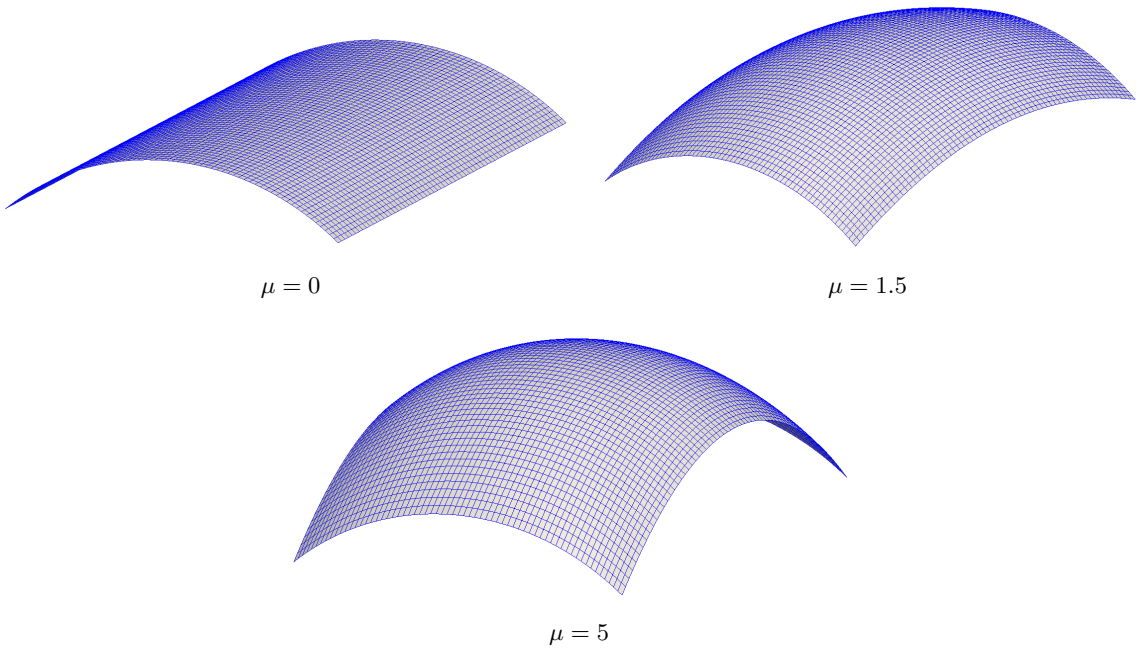


Figure 5.21: Scordelis-Lo roof. Meshes for polynomial degree $p = 3$ and 1024 elements for the parameters $\mu = 0, 5, 10$ (the meshes are symmetrically extended for visualization purposes.

$0, 5, 10$. In this case the scales of values do not refer to a common range, since the values assumed by the solutions for the different parameters differs a lot among them. In Table 5.6 we report the data of the problem, the results provided by MDEIM and greedy algorithm,

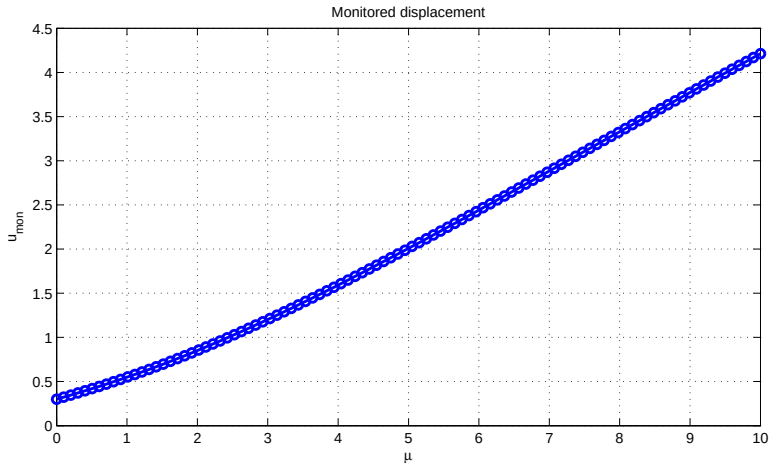


Figure 5.22: Scordelis-Lo roof. Monitored displacement $u_{mon}(\mu)$ for $\mu \in [0, 10]$.

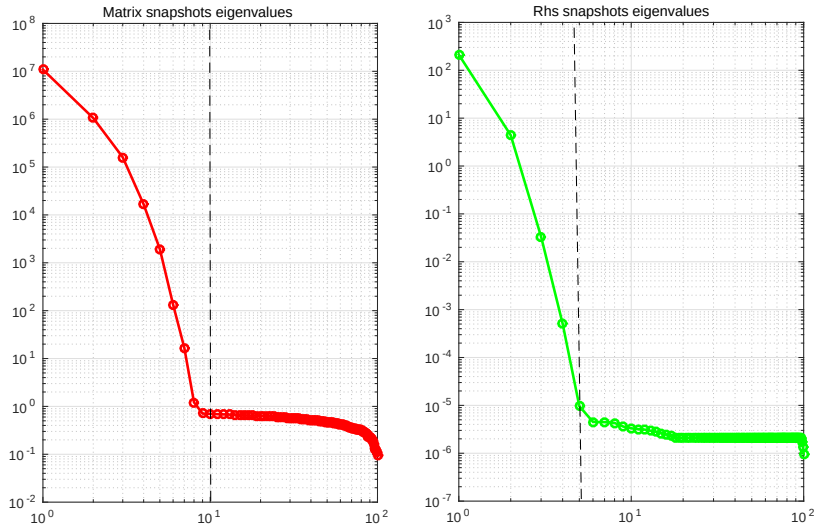


Figure 5.23: Scordelis-Lo roof. Singular values of the snapshots matrices of the vectorized stiffness matrices and right hand side vectors for polynomial degrees 3.

and the computational times requested to solve the problem at hand. We conclude that the online evaluation of the solution of the MDEIM-RB problem is about 11 times faster than IGA one. Finally, in Figs. 5.25 and 5.26 we report and compare the average relative errors between the IGA and the MDEIM-RB solutions vs N, for $1 \leq N \leq N_{max}$ for different values of M_k and M_f, respectively, computed on a 400 samples parameter set Ξ_{test}. In both the cases, the RB approximation is more accurate if we consider the maximum number of affine terms selected by MDEIM.

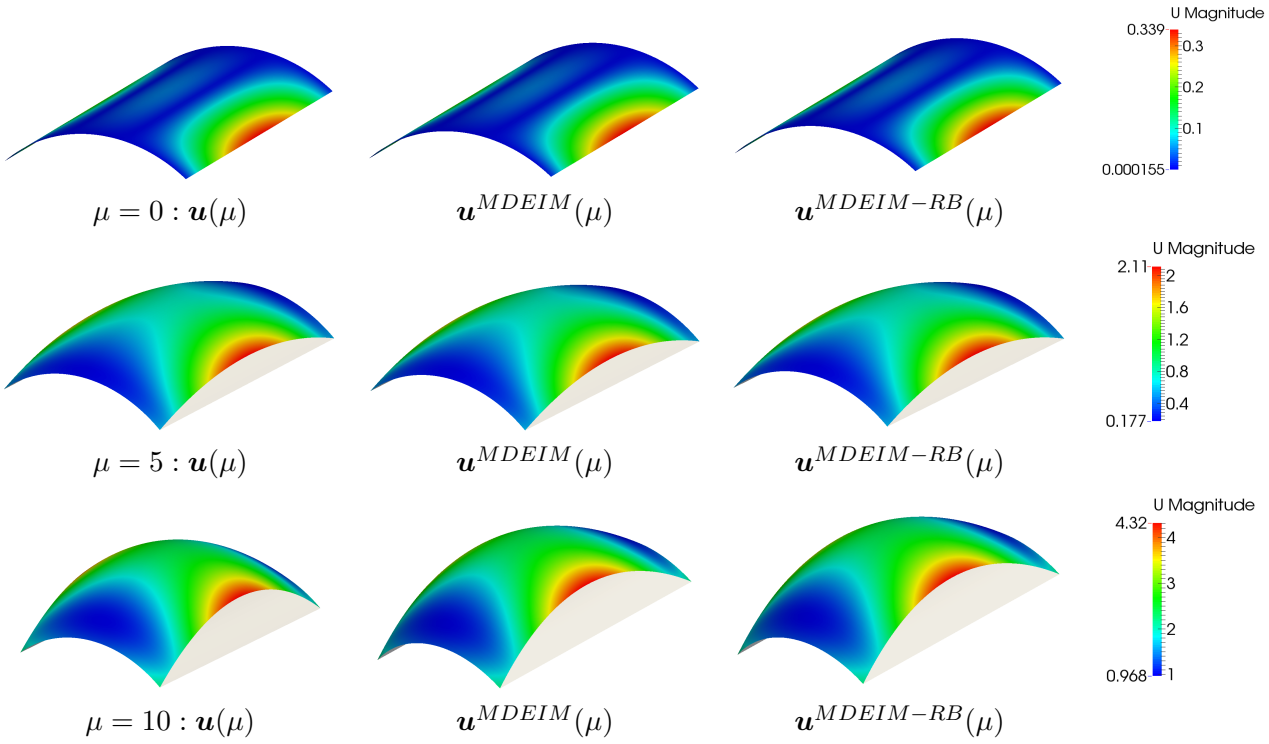


Figure 5.24: Parametrized Scordelis-Lo roof for $\mu = 0, 5, 10$.

Table 5.6: *Scordelis-Lo roof. Data of the problems, MDEIM and the greedy algorithm, and computational times requested offline and online.*

IGA	
NURBS degree	3
Elements	1024
$\mathcal{N}$	3675
t_{IGA}	3.79 s
MDEIM	
Number of parameters	1
Number of samples	100
tol	10^{-6}
Affine matrix components $\boldsymbol{K}^m$	10
Affine rhs components $\boldsymbol{f}^m$	5
Reduced elements	107
Greedy-RB	
Greedy tolerance	10^{-6}
Sample train (greedy)	150
Greedy space dimension N	10
Greedy construction time (offline)	3.86 min
$t_{greedy-RB}$ (online)	0.33 s
$t_{IGA}/t_{greedy-RB}$	11.48

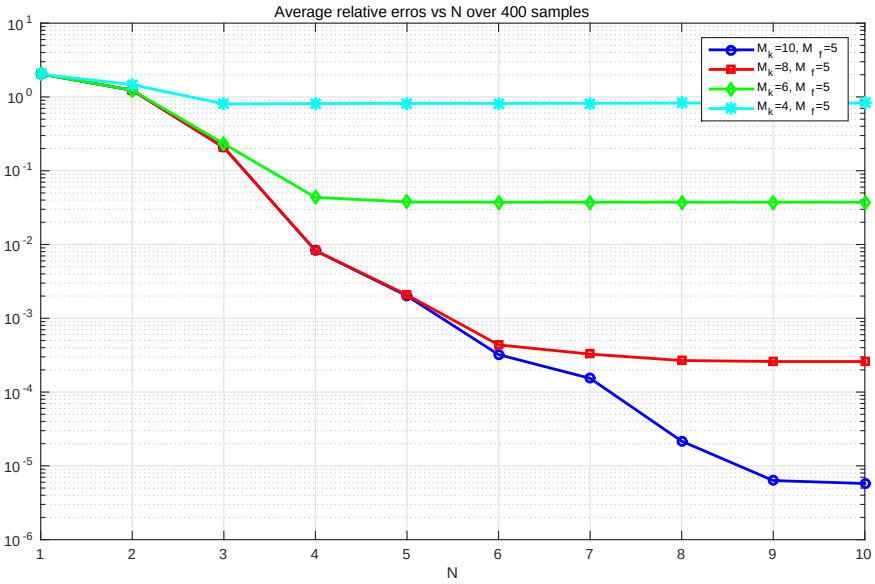


Figure 5.25: Scordelis-Lo roof. Average relative error between IGA and MDEIM-RB solutions vs N, for $1 \leq N \leq N_{max}$ for different values of M_k, computed on a 400 samples parameter set Ξ_{test}.

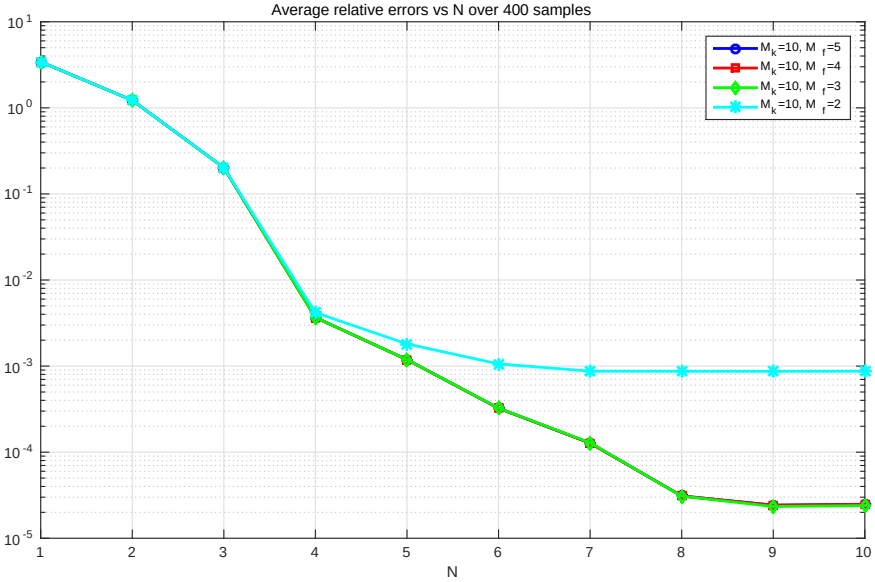


Figure 5.26: Scordelis-Lo roof. Average relative error between IGA and MDEIM-RB solutions vs N, for $1 \leq N \leq N_{max}$ for different values of M_f, computed on a 400 samples parameter set Ξ_{test}.

Conclusions

The main achievement of this work is the application of Reduced Basis (RB) method for the efficient numerical solution of parametrized PDEs approximated with Isogeometric Analysis (IGA), with particular emphasis on geometrical parametrizations.

We considered IGA [CHB09, PT97] in the framework of the Galerkin method. In particular, we focused on the numerical solution of second order PDEs on lower dimensional manifolds, as surfaces in a three dimensional space, and fourth order PDEs, namely Kirchoff-Love shell models, parametrized with respect to the NURBS control points, the weights, or both control points and weights. For all these possible kinds of parametrizations, we analyzed and implemented, for the first time, a suitable strategy to apply RB to IGA for parametrized geometries in a feasible way.

Specifically, we studied a class of problems such that the parametrization of the computational domain induces a nonaffine geometrical transformation, that in turn induces a nonaffine dependence of the problem and its formulation on the parameters and on the spatial coordinates. However, we recall that in order to fully exploit the computational savings of model reduction, the RB method is based on the affinity assumption [GP05, Man12, Qua14, QRM11, SVH$^+$06]. With this aim, we developed at the Offline phase affine approximations of the involved linear operators, finally leading, Online, to inexpensive evaluation of the approximated operators. The recovery of the affine dependence was the crucial point of the RB and IGA interfacing procedure.

We started by considering a class of problems defined on NURBS surfaces parametrized only with respect to the NURBS control points coordinates. In this case, we traced back the problem onto a parameter independent domain, by explaining how to properly choose this reference domain and giving an explicit expression of the parametric transformation that leads from the reference to the original domain. In the problem recast in the reference domain, the effect of the geometric variations, that is the parameter dependence, was traced back and limited to its parametrized transformation tensors. As anticipated, these tensors were non linear functions of both the spatial coordinates and the parameters. Thus, we approximated them with affine parametric dependent terms through the Empirical Interpolation Method (EIM) [BMNP04].

Then, we considered the case of geometries with parametrized NURBS weights. In this case, we realized that the procedure of tracing back the problem in a reference configuration and apply EIM was not usable anymore. As a matter of fact, being the NURBS functions used to build the original domain parametric dependent, it was not immediate to build, by means of the same basis functions, a reference domain. Moreover, we were unable to separate, in the integrals of the problem formulation, parameter dependent terms, to be efficiently approximated with EIM, from parameter independent ones. For this reason, in order to restore the affinity assumption, we resorted to an alternative technique, namely the Matrix Discrete

Empirical Interpolation Method (MDEIM) [NMA15]. Compared to EIM, MDEIM, allows to obtain an affine approximation of parameter dependent problem by directly interpolating the IGA matrices and vectors assembled in the parameter dependent domain. Moreover, we highlighted that MDEIM can be used both for parametrized NURBS control points and weights. We compared the performances of EIM and MDEIM and concluded that MDEIM entails less affine terms than EIM for the approximation of the IGA solutions, thus allowing a larger computational saving both at the offline and the online stages.

Once recovered an affine approximation of the IGA problem, we built two different reduced order models by means of the Proper Orthogonal Decomposition (POD) and the Greedy algorithm. Numerical tests have shown a great accuracy between the reduced models and the full-order ones. Moreover, although the construction of the reduced spaces using POD or the greedy algorithm is in principle different, we did not observe noticeable discrepancies in their accuracy and speedup in the prediction of the results at the Online stage. On the other hand, we remarked some differences in their behavior at the Offline phase and concluded that, since the norm of the residual in the greedy algorithm depends on the number of affine terms, the greedy-RB strategy is preferable to the POD, for problems with a low number of affine terms.

The work carried out shows that the developed RB-IGA method allows large computational savings for problems defined on complex parametrized NURBS geometries. As application we considered a class of structural problems in the context of Kirchoff-Love shell elements [KBLW09, CHB09, BB93]. Kirchhoff-Love shell theories is one of the main shell models used, normally referred as thin shell theory. This model requires at least C^1 continuous basis functions and this severely limits its application in Engineering, since using the standard Lagrange polynomials as basis functions this requirement cannot be fulfilled. Conversely, in IGA, the C^1 continuity of the basis functions is easily achievable. As a matter of fact, NURBS, being smooth, high order functions, allow great geometric flexibility and high order continuities at the same time, resulting ideally suited as basis functions for Kirchoff-Love shell elements. Three benchmarks from shell obstacle courses [CHB09] were used to test our method: the pinched cylinder, the hemisphere, and the Scordelis-Lo roof problems. For these problems we proposed a parametrized NURBS representation and solved them by means of a MDEIM-RB approximation obtaining a significant computational speedup. On a side note, we remark that, for the first time, the RB method has been applied to the numerical solution of parametrized fourth order PDEs.

The developed RB-IGA method is thus interesting from both academic and industrial points of view. As a matter of fact, since IGA is directly interfaced with CAD, a possible future development of the work could be the implementation of a software based on the RB-IGA method, allowing real-time evaluations of structural outputs of interest for different NURBS parametrizations. Moreover, although we focused on structural problems, the methodology outlined is well suited for several other applications, especially in the context of optimal control or shape optimization.

Bibliography

[ACdSAF03] P.M.A. Areias, J.M.A. César de Sá, C.A. António, and A.A. Fernandes. Analysis of 3D problems using a new enhanced strain hexahedral element. *International Journal for Numerical Methods in Engineering*, 58(11):1637–1682, 2003.

[BB93] M. L. Bucalem and K.-J. Bathe. Higher-order MITC general shell elements. *International Journal for Numerical Methods in Engineering*, 36(21):3729–3754, 1993.

[BBHH10] D.J. Benson, Y. Bazilevs, M.C. Hsu, and T.J.R. Hughes. Isogeometric shell analysis: the Reissner–Mindlin shell. *Computer Methods in Applied Mechanics and Engineering*, 199(5):276–289, 2010.

[BBWR04] M Bischoff, K-U Bletzinger, WA Wall, and E Ramm. Models and finite elements for thin-walled structures. *Encyclopedia of computational mechanics*, 2004.

[BCD+11] P. Binev, A. Cohen, W. Dahmen, R. DeVore, G. Petrova, and P. Wojtaszczyk. Convergence rates for greedy algorithms in reduced basis methods. *SIAM Journal on Mathematical Analysis*, 43(3):1457–1472, 2011.

[BDQ15] A. Bartezzaghi, L. Dedè, and A. Quarteroni. Isogeometric analysis of high order partial differential equations on surfaces. Technical report MATHICSE 07.2015, 2015.

[BHL93] G. Berkooz, P. Holmes, and J. L. Lumley. The proper orthogonal decomposition in the analysis of turbulent flows. *Annual review of fluid mechanics*, 25(1):539–575, 1993.

[BMNP04] M. Barrault, Y. Maday, N.C. Nguyen, and A.T. Patera. An empirical interpolation method: application to efficient reduced-basis discretization of partial differential equations. *Comptes Rendus Mathématique*, 339(9):667–672, 2004.

[BMS14] M. Bebendorf, Y. Maday, and B. Stamm. Comparison of some reduced representation approximations. In *Reduced Order Methods for Modeling and Computational Reduction*, pages 67–100. Springer, 2014.

[BSL+85] T. Belytschko, H. Stolarski, W. K. Liu, N. Carpenter, and J. S.J. Ong. Stress projection for membrane and shear locking in shell finite elements. *Computer Methods in Applied Mechanics and Engineering*, 51(1):221–258, 1985.

[CHB05] J.A. Cottrell, T.J.R. Hughes, and Y. Bazilevs. Isogeometric Analysis: CAD, finite elements, nurbs, exact geometry and mesh refinement. *Computer Methods in Applied Mechanics and Engineering*, 194:4135–4195, 2005.

[CHB09] J.A. Cottrell, T.J.R Hughes, and Y. Bazilevs. *Isogeometric analysis: toward integration of CAD and FEA*. John Wiley & Sons, 2009.

[Cia00] P. G. Ciarlet. *Mathematical Elasticity, volume III: Theory of shells*. Studies in Mathematics and its Applications, 29. North-Holland Publishing Co., Amsterdam, 2000.

[CM08] P. G. Ciarlet and C. Mardare. An introduction to shell theory. differential geometry: theory and applications. *Computer Methods in Applied Mechanics and Engineering*, pages 94–184, 2008.

[COS00] F. Cirak, M. Ortiz, and P. Schroder. Subdivision surfaces: a new paradigm for thin-shell finite-element analysis. *International Journal for Numerical Methods in Engineering*, 47(12):2039–2072, 2000.

[CS10] S. Chaturantabut and D.C. Sorensen. Nonlinear model reduction via discrete empirical interpolation. *SIAM Journal on Scientific Computing*, 32(5):2737–2764, 2010.

[CVP05] N.N. Cuong, K. Veroy, and A. T. Patera. Certified real-time solution of parametrized partial differential equations. In *Handbook of Materials Modeling*, pages 1529–1564. Springer, 2005.

[DQ15] L. Dedè and A. Quarteroni. Isogeometric Analyis for second order partial differential equations on surfaces. *Computer Methods in Applied Mechanics and Engineering*, 284:807–834, 2015.

[DR75] J. Davis and P. Rabinovitz. *Methods of numerical integration*. Academic Press, 1975.

[GMNP07] M. A. Grepl, Y. Maday, N. C. Nguyen, and A.T. Patera. Efficient reduced-basis treatment of nonaffine and nonlinear partial differential equations. *ESAIM: Mathematical Modelling and Numerical Analysis*, 41(03):575–605, 2007.

[GP05] M.A. Grepl and A.T. Patera. A posteriori error bounds for reduced-basis approximations of parametrized parabolic partial differential equations. *ESAIM: Mathematical Modelling and Numerical Analysis*, 39(01):157–181, 2005.

[HCB05] T.J.R. Hughes, J.A. Cottrell, and Y. Bazilevs. Isogeometric Analysis: CAD, finite elements, nurbs, exact geometry and mesh refinement. *Computer Methods in Applied Mechanics and Engineering*, 194(39):4135–4195, 2005.

[HO08] B. Haasdonk and M. Ohlberger. Reduced basis method for finite volume approximations of parametrized linear evolution equations. *ESAIM: Mathematical Modelling and Numerical Analysis*, 42(02):277–302, 2008.

[HRSP07] D.B.P. Huynh, G. Rozza, S. Sen, and A.T Patera. A successive constraint linear optimization method for lower bounds of parametric coercivity and inf–sup stability constants. *Comptes Rendus Mathématique*, 345(8):473–478, 2007.

[KBLW09] J. Kiendl, K-U. Bletzinger, J. Linhard, and R. Wüchner. Isogeometric shell analysis with kirchhoff–love elements. *Computer Methods in Applied Mechanics and Engineering*, 198(49):3902–3914, 2009.

[LMR06] A.E. Løvgren, Y. Maday, and E.M. Rønquist. A reduced basis element method for the steady Stokes problem. *ESAIM: Mathematical Modelling and Numerical Analysis*, 40(3):529–552, 2006.

[Man12] A. Manzoni. Reduced models for optimal control, shape optimization and inverse problems in haemodynamics. PhD thesis, EPFL. 2012.

[MH85] R. H. Macneal and R. L. Harder. A proposed standard set of problems to test finite element accuracy. *Finite elements in Analysis and Design*, 1(1):3–20, 1985.

[MRA13] D. Millán, A. Rosolen, and M. Arroyo. Nonlinear manifold learning for meshfree finite deformation thin-shell analysis. *International Journal for Numerical Methods in Engineering*, 93(7):685–713, 2013.

[MSH15] A. Manzoni, F. Salmoiraghi, and L. Heltai. Reduced basis isogeometric methods (RB-IGA) for the real-time simulation of potential flows about parametrized naca airfoils. *Computer Methods in Applied Mechanics and Engineering*, 284:1147–1180, 2015.

[Neg11] F. Negri. Reduced basis method for parametrized optimal control problems governed by PDEs. Technical report MATHICSE, 2011.

[NMA15] F. Negri, A. Manzoni, and D. Amsallem. Efficient model reduction of parametrized systems by Matrix Discrete Empirical Interpolation. Technical report MATHICSE 02.2015, 2015.

[NRP09] N.C. Nguyen, G. Rozza, and A.T. Patera. Reduced basis approximation and a posteriori error estimation for the time-dependent viscous burgers equation. *Calcolo*, 46(3):157–185, 2009.

[PT97] L Piegl and W Tiller. The nurbs book. 2nd. 1997.

[QRM11] 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.

[Qua14] A. Quarteroni. Reduced basis approximation for parametrized partial differential equations. In *Numerical Models for Differential Problems*, pages 585–633. Springer, 2014.

[RHP08] G. Rozza, D.B.P. Huynh, and A.T. Patera. Reduced basis approximation and a posteriori error estimation for affinely parametrized elliptic coercive partial differential equations. *Archives of Computational Methods in Engineering*, 15(3):229–275, 2008.

[SFR89] J.C. Simo, D.D. Fox, and M.S. Rifai. On a stress resultant geometrically exact shell model. Part II: The linear theory; computational aspects. *Computer Methods in Applied Mechanics and Engineering*, 73(1):53–92, 1989.

[SVH+06] S. Sen, K. Veroy, DBP Huynh, S. Deparis, N.C. Nguyen, and A.T. Patera. Natural norm a posteriori error estimators for reduced basis approximations. *Journal of Computational Physics*, 217(1):37–62, 2006.

[TDQ14] A. Tagliabue, L. Dedè, and A. Quarteroni. Isogeometric analysis and error estimates for high order partial differential equations in fluid dynamics. *Computers & Fluids*, 102:277–303, 2014.

[Ton] T. Tonn. Reduced-basis method (rbm) for non-affine elliptic parametrized pdes:(motivated by optimization in hydromechanics). PhD thesis, Universitat Ulm, Diss.

[Vol11] S. Volkwein. Model reduction using proper orthogonal decomposition. *Lecture Notes, Institute of Mathematics and Scientific Computing, University of Graz. http://www. uni-graz. at/imawww/volkwein/POD. pdf*, 2011.

[VP05] K. Veroy and A.T. Patera. Certified real-time solution of the parametrized steady incompressible Navier-Stokes equations: rigorous reduced-basis a posteriori error bounds. *International Journal for Numerical Methods in Fluids*, 47(8-9):773–788, 2005.

[VPP03] K. Veroy, C. Prud'homme, and A.T Patera. Reduced basis approximation of the viscous Burgers' equation: rigorous a posteriori error bounds. *Comptes Rendus Mathématique*, 337(9):619–624, 2003.

[VPRP03] K. Veroy, C. Prudhomme, D.V. Rovas, and A.T. Patera. A posteriori error bounds for reduced-basis approximation of parametrized noncoercive and nonlinear elliptic partial differential equations. In *Proceedings of the 16th AIAA computational fluid dynamics conference*, volume 3847, 2003.

[WWS14] O. Weeger, U. Wever, and B. Simeon. Nonlinear frequency response analysis of structural vibrations. *Computational Mechanics*, 54(6):1477–1495, 2014.

Acknowledgments

Desidero ringraziare il Prof. Alfio Quarteroni e il Dott. Luca Dede', relatori di questa tesi, per la loro costante disponibilità e il supporto datomi durante questo periodo.

Ringrazio in particolare il Prof. Alfio Quarteroni per avermi dato la possibilità di svolgere questa tesi presso l'EPFL e per avermi permesso di lavorare su un argomento così altamente interessante e stimolante.

Un grazie speciale va al Dott. Luca Dede' per aver reso questa esperienza altamente educativa e per i suoi continui incoraggiamenti.

Ringrazio Federico Negri per i suoi prezioni consigli e suggerimenti e per avermi aiutata in qualsiasi momento ne avessi bisogno.

Grazie al Prof. Stefano Berrone per essersi reso disponibile ad essere il mio correlatore del Politecnico di Torino per conseguire la seconda laurea con l'Alta Scuola Politecnica.

Grazie a tutti i componenti del gruppo di Chair of Modelling and Scientific Computing (CMCS) per l'ambiente altamente professionale e gioviale che mi hanno permesso di condividere con loro ogni giorno.