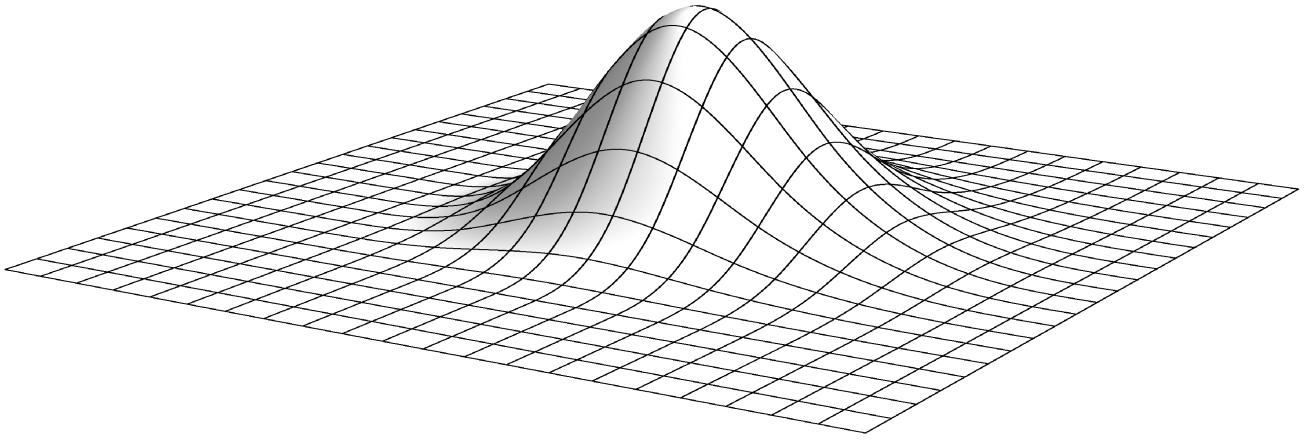
DOCTORAL DISSERTATION

# Construction of polynomial spline spaces over T-meshes for its application in Isogeometric Analysis



Marina Brovka

Discretization and Applications Division

Las Palmas de Gran Canaria • June 2016

INSTITUTO UNIVERSITARIO SIANI INGENIERIA COMPUTACIONAL

UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA

UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA
Instituto Universitario de Sistemas Inteligentes
y Aplicaciones Numéricas en Ingeniería

DAVID GREINER SÁNCHEZ, PROFESOR TITULAR DE UNIVERSIDAD Y SECRETARIO DEL INSTITUTO UNIVERSITARIO DE SISTEMAS INTELIGENTES Y APLICACIONES NUMÉRICAS EN INGENIERÍA (SIANI) DE LA UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA,

CERTIFICA

Que el Consejo de Doctores del Instituto Universitario de Sistemas Inteligentes y Aplicaciones Numéricas en Ingeniería (SIANI) en su sesión de fecha 27 de mayo de 2016 tomó el acuerdo de dar el consentimiento para su tramitación a la tesis doctoral titulada "Construction of polynomial spline spaces over T-meshes for its application in Isogeometric Analysis. Construcción de espacios spline polinómicos sobre T-meshes para su aplicación en Análisis Isogeométrico." presentada por la doctoranda **Dña. Marina Brovka** y dirigida por Dr. D. José María Escobar Sánchez y Dr. D. Rafael Montenegro Armas, a la vista de la idoneidad y calidad de su contenido, interés y relevancia del tema a nivel internacional.

Para que así conste, y a los efectos oportunos se expide el correspondiente certificado a treinta y uno de mayo de dos mil dieciséis.

UNIVERSIDAD DE LAS PALMAS
DE GRAN CANARIA

SIANI

# Construction of polynomial spline spaces over T-meshes for its application in Isogeometric Analysis

Programa de doctorado:
Sistemas Inteligentes y Aplicaciones Numéricas en Ingeniería
Instituto Universitario SIANI

Autor: Marina Brovka

Director: José María Escobar Sánchez

Director: Rafael Alejandro Montenegro Armas

Las Palmas de Gran Canaria, June 2016

*To My Parents*

**Acknowledgements**

I would like to express my deepest gratitude to my supervisors Prof. Rafael Montenegro Armas and Prof. José María Escobar Sánchez for their guidance and patience. My sincere thanks for giving me the opportunity to work in their research group, for their trust, continuous support and friendly attitude that contributed a lot to my motivation.

I am very thankful to Prof. José Manuel Cascón Barbero for his help, collaboration and numerous advices and suggestions.

I am very happy and grateful for having my co-worker José Iván López González, for our fruitful collaboration and for being a team, which made possible our research.

Of cause, I would like to express my gratitude and appreciation for all the help, support and nice working atmosphere to all members of our research group: Dr. Albert Oliver Serra, Guillermo Socorro, Dr. Eduardo Rodríguez Barrera, Jabel Ramírez Naranjo and Prof. Gustavo Montero García.

I would like also to acknowledge The University of Las Palmas de Gran Canaria for their financial support ("Programa de FPI propio de la ULPGC") and all the staff of the university institute SIANI for giving opportunities and facilitating the incorporation to scientific research.

**Construction of polynomial spline spaces over T-meshes for its application in Isogeometric Analysis**

## Abstract

This dissertation addresses the issue of construction of appropriate spline spaces over T-meshes for its application in Isogeometric Analysis. Our work was motivated by the inability of the B-splines to perform local refinement, which complicates their use for analysis. In this thesis we propose a new strategy for defining cubic tensor product spline blending functions that span spaces suitable for analysis. The technique is designed for hierarchical T-meshes with a quad- and octree subdivision scheme. This type of meshes can be efficiently implemented with tree data structures which are frequently used in engineering. Due to the elevated complexity of all current strategies, the main goal we pursue here is the simplicity of the implementation, both in 2D and 3D. For that, we have to assume a restriction on the T-mesh. Namely, the T-mesh should fulfil the requirement of being a strongly balanced mesh. Assuming this reasonable and frequently used restriction over the T-mesh, we can define easily cubic spline functions with desirable properties: linear independence, $C^2$-continuity, cubic polynomial reproduction property, nesting behaviour of the spaces and a straightforward implementation. The proposed strategy includes simple rules for inferring local knot vectors to define blending functions for a given T-mesh. A detailed description of the method is given and the approximation properties of the constructed spline spaces are tested in different type of problems, where an adaptive refinement is necessary to obtain an accurate numerical solution.

# Contents

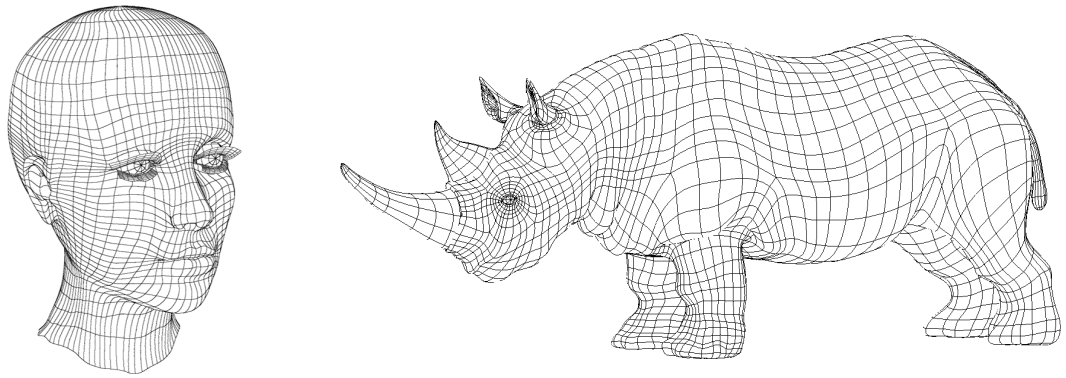
# CHAPTER 1

# Introduction

## 1.1 Isogeometric Analysis concept

Computer-aided design (CAD) is the process of creating a design using computer software. After CAD has emerged (1960-1970) all engineering drawings, done manually using pencil and paper, was replaced by files describing the physical model. That entailed a huge advance in all engineering industry. Nowadays CAD is a widely used technology with numerous applications in automotive, shipbuilding, aerospace industries, industrial and architectural design, computer animation for special effects in cinema. The most commonly used mathematical tool for representing curves and surfaces in CAD are B-splines and NURBS (Non-Uniform Rational B-splines) [1, 2]. They offer a convenient free-form surface modelling and can exactly represent all conic sections. Efficient algorithms to handle spline object are available. Besides, B-splines possess some additional useful properties as higher smoothness and convex hull property.

On the other hand, another important computational technique that has been under intensive development since its emerging (1950-1960) is Finite Element Method (FEM). This numerical method for solving partial differential equations has received a big amount of research effort and became a standard tool in engineering industry. The basic components of the method are the variational formulation of the physical problem and the discrete space used to approximate its solution. The approximation space is defined by its basis functions (Lagrange or Hermite interpolating polynomials). Each basis function is defined locally on its element. To perform the analysis, the computational domain of the problem should be decomposed in non-overlapping simple elements (triangles, quadrilaterals, tetrahedra, hexahedra). This step is referred to as mesh generation process. Taking into account that nowadays the geometry comes from CAD, there is a necessity to generate an analysis-suitable mesh of the object from its CAD representation. It is not a trivial task due to increasing complexity of engineering design. Finite Element Analysis (FEA) and CAD technologies have evolved separately and they use different geometrical tool. That impedes the communication between them



(a) (b)

**Figure 1.1:** *(a) Head NURBS model, (b) Rhinocero NURBS model. Images taken from* `http://www.3drender.com/`

and make the process of data transferring very time-consuming. Creation of the analysis-suitable model and mesh generation process consumes the major part of all analysis time. Tight communication between CAD model and FEA geometry is also crucial to make possible design optimization. In addition, finite element mesh does not reproduce the exact model designed by CAD. For some problems geometry imprecision can lead to important error in numerical solution. To approximate a curved boundary with a desired precision an adaptive mesh refinement is required. This is possible only with an automatic interaction between exact CAD geometry and FEM mesh. With all these inconveniences it became evident the necessity to unify the design and the analysis in a unique engineering process. CAD has to provide directly a geometric model suitable for analysis, or it is necessary to have an automatic method to obtain analysis suitable model from its CAD representation. For that, the analysis process should be changed and adapted for its application on CAD geometries. The thought was proposed in 2005 by Tom Hughes and coworkers in [3]. The concept received the name *Isogeometric Analysis* (IGA). The idea is to use for the analysis the same basis functions (originally NURBS) that have been used for construction of the CAD model. Isogeometric Analysis can be seen as a generalization of FEM that uses basis functions with higher regularity.

Since its introduction IGA attracted a lot of attention in research community and it has been an object of numerous investigation works in the last years. At present it is considered a promising tool for bridging the gap between CAD and FEA industries. Besides, it can offer some additional beneficial properties and possibilities compared with classical Finite Element Method. Here are some of them:

- Working with the exact geometry leads to superior accuracy in the problems sensitive to geometry approximation (contact problems, boundary layer prob-

lems in aerodynamic and hydrodynamic). Even if the exact geometry is not achieved, smooth boundary approximation is necessary for some problems that require curvature continuity of the boundary.

- Better accuracy of numerical solution due to higher regularity of the basis functions [4].
- Higher degree of continuity also allows the use of IGA for solving partial differential equations of order greater than two without the necessity to change variational formulation. For example, Cahn-Hilliard phase-field equation [5, 6, 7, 8].
- Possibility of collocation methods using strong formulation [9, 10, 11, 12].

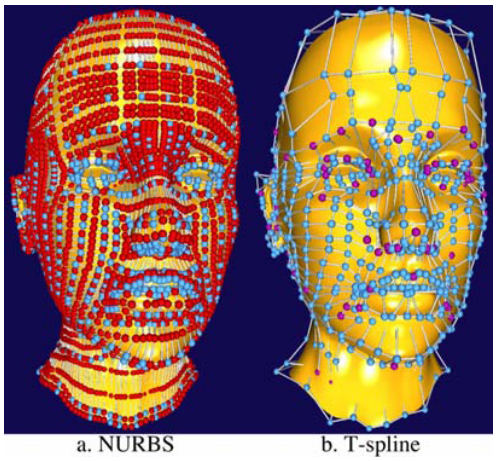
However, to accomplish the ambitious idea of Isogeometric Analysis and make it work in practice, some issues should be resolved first. This question is revised in the next section.

## 1.2 State of the art

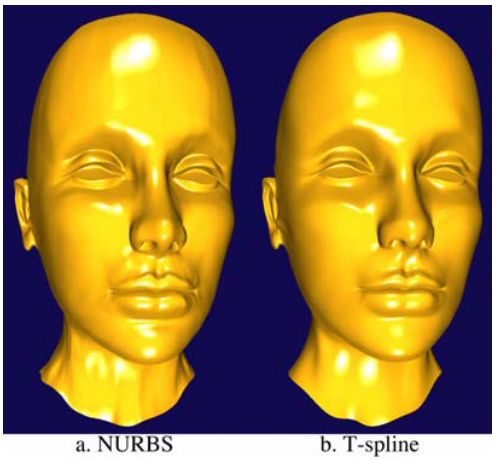
IGA is relatively new method that needs a theoretical framework to be developed and some open problems to be solved. Clearly, a lot of question should be studied: error estimation theory for IGA [13, 14, 15], efficient quadrature rules [16, 17], efficiency of the direct and iterative solvers for the systems arising in the method [18, 19], efficient implementation of the structures and procedures arising in the method, imposing boundary conditions, the influence of parameterization quality on accuracy of the solution [20]. However, the most urgent and important problems of IGA are the local refinement problem and the parameterization of computational domain.

### 1.2.1 Local refinement

Originally Isogeometric Analysis concept was proposed and tested using NURBS basis functions inasmuch as NURBS was the most common tool used by CAD software at that moment. However, it suffers from an important drawback: their tensor product structure does not allow for local refinement. NURBS surface is defined by a set of control points which lie, topologically, in a rectangular grid. A new control point insertion induces the insertion of an entire row of control points extended through the entire domain. That leads to a large number of superfluous control points. The representation of local features is inefficient and requires the use of several NURBS patches joined together. Frequently this join is discontinuous, i.e. it has gaps and overlaps that make the model not suitable for analysis. T-splines were proposed by Sederberg et al. [21] as an alternative to NURBS that permits local refinement. T-splines can be seen as a generalization




(a) *"Head modeled (a) as a NURBS with* 4712 *control points and (b) as a T-spline with* 1109 *control points. The red NURBS control points are superfluous"*




(b) *"NURBS head model, converted to a T-spline"*

**Figure 1.2:** *Images taken from the original work of Sederberg et al. [22]: "T-spline simplification and local refinement".*

of NURBS. T-splines allows the insertion of a partial row of control points that terminates in T-junction, which makes them locally refinable. T-spline plug-in are currently available for two NURBS-based CAD software Maya and Rhino, see `http://www.tsplines.com/products/tsplines-for-rhino.html`. T-splines offer a flexible tool for creating one-patch watertight surfaces with local detailed features and lower number of control points compared with NURBS, see Fig. 1.2. However, it appears that initially introduced T-splines are not suitable for their use in Isogeometric Analysis since they lack some properties essential for analysis and for proper convergence behaviour: linear independence, polynomial reproduction property and nesting behaviour of approximation spaces (this question will be illustrated in more details in Chapter 2. Besides, rational T-splines complicate and increase computational cost of the calculus of the derivatives.

Therefore, it is still an open issue in the context of Isogeometric Analysis to find an alternative to NURBS that overcomes the local refinement problem and can be used in analysis. This issue has been the object of numerous research works in recent years.

Analysis-suitable T-splines, proposed by Scott et al. in [23], are a class of T-splines defined over T-meshes that meet certain topological restrictions formulated in terms of T-junction extensions. Blending functions defined over an extended analysis-suitable T-mesh are linearly independent [24] and possess polynomial reproduction property. The refinement algorithm allows to accomplish highly localized refinements and construct nested T-spline spaces, but it presents an elevated implementation complexity and, as far as we know, the generalization of the

strategy to 3D cases is still an open question.

Another well known approach for local enrichment of the approximation space is the hierarchical refinement, originally introduced by Forsey and Bartels in [25] and later developed in [26]. Recently, hierarchical refinement technique in the context of Isogeometric Analysis was described in [27, 28, 29]. This approach is based on a simple and natural idea to construct multilevel spaces by replacing coarse level functions with finer basis functions. Starting from an initial uniform mesh, hierarchical refinement scheme leads to sequential construction of nested spline spaces with linearly independent basis functions. Relatively simple implementation and straightforward generalization to 3D make it an attractive option for local refinement. However, a shortcoming of this strategy is the impossibility to define a spline space over a given arbitrary T-mesh, as well as the presence of redundant basis functions and excessive support overlapping. An interesting theoretical approach to the latter problem was given in [30]. The truncation technique is applied to redefine the function supports and reduce their overlapping. However, that elevates considerably the computational cost of the strategy.

Other strategies for performing local refinement of spline spaces are $C^1$-continuous PHT-splines [31] and Locally Refined B-splines (LR-splines) [32].

It is worth mentioning another group of options for defining multivariate spline functions that do not have tensor product structure. Namely, bivariate simplex splines over triangulation [33]; simplex splines over Delaunay configurations, as a natural generalization of univariate B-splines, was proposed in [34, 35, 36]; box splines [37]; quadratic $C^1-$continuous splines over Powell-Sabin triangle split [38]. Recent examples of isogeometric analysis using splines over triangulations can be found in [39, 40, 41]. Splines over triangulation could offer a better flexibility and adaptivity to irregular domains compared to more rigid tensor product splines. However, it is not a trivial task to define globally smooth spline spaces over triangulations. Currently available options presents a high complexity, and it seems that tensor product splines are still the most popular and standard option for CAD due to their simplicity.

### 1.2.2 Domain parameterization

CAD models usually provide only the boundary surface of a solid. This spline representation of the boundary can be used directly in isogeometric shell analysis or isogeometric boundary integral method. But the application of Isogeometric Analysis requires a full volumetric representation of the geometry. So, another open problem of Isogeometric Analysis is how to generate a trivariate spline representation of a solid starting from the CAD description of its boundary. As it is pointed by Cottrell et al. in [42], *"the most significant challenge facing Isogeometric Analysis is developing three-dimensional spline parameterizations from surfaces"*. Parameterization is suitable for analysis if it does not have self-intersection, i.e. it is invertible. Moreover, in order to expect a high accuracy in numerical results it

is necessary to obtain a good quality volume parameterization. Orthogonality and uniformity of isoparametric curves are desirable for the tensor-product structured parameterization. It is not trivial task to obtain a good quality smooth global parametric mapping for complex domains and it can be very time-consuming. For the application of IGA it is essential to have an effective method to construct a spline parameterization. Here are some attempts to tackle the problem.

In [43] the parameterization is found by solving a constraint optimization problem for a planar B-spline surface. Constraints are defined by imposing injectivity sufficient conditions in terms of the control points, and the optimization consists in minimizing some energy function in order to reach a good orthogonality and uniformity of the parametric mapping. The idea was extended for 3D in [44]. Another similar technique was proposed by these authors in [45, 46]. They use a harmonic mapping obtained by solving an optimization problem for the control points. Additional term is added to the objective function in order to improve the quality where needed.

The use of harmonic mapping is a common characteristic of several works dealing with 2D and 3D parameterization methods. For example, Li et al. [47] construct a harmonic volumetric mapping through a meshless procedure by using a boundary method. The algorithm can be applied to any genus data, but it is complex and requires placing some source and collocation points on an offset surface. Optimal results of source positions are unknown, and in practice they are chosen in a trial-and-error manner or with the help of human experience.

Martin et al. [48] present a methodology based on discrete harmonic functions to parameterize a solid, where the input data are surface triangulation and a tetrahedral mesh of the solid. They solve several Laplace equations, first on the surface to establish surface parameterization and then on the complete 3D domain using FEM. The two obtained orthogonal harmonic functions are used to construct a structured hexahedral mesh of the solid, which is smoothed to remove irregularities. The user has to make an initial choice of two critical points to establish the surface parameterization and to fix a seed for generating the skeleton.

Zhang et al. proposed in [49] a procedure to construct T-spline representation of a genus-zero solid from its boundary triangulation. First, a parametric mapping between the triangulation and the boundary of the parametric domain, a unit cube, is established using Floater parameterization method. Then an octree subdivision is carried out for the cube until the error between the T-mesh and the input triangle mesh is less than a threshold. During this process, the boundary nodes are mapped to the input triangle mesh and, then, the interior nodes are relocated via T-mesh optimization, which is maximizing the worst Jacobian of the trilinear map for each element. In [50] the method was extended for arbitrary genus topology. A smooth harmonic scalar field over the surface triangulation is computed, and saddle points are extracted to determine the topology of the object and to construct a polycube with the same topology that serves as the parametric domain for the trivariate T-spline.

## 1.3 Goal and outline of the thesis

This thesis is the result of our group research work on Isogeometric Analysis. As was mentioned before, it is relatively new method with many questions to explore. A lot of new interesting things about IGA have been studied and discovered since we have embarked on this journey. Some of these things will be reflected implicitly in the present work. But, mainly, our attention was focused on two open problems of IGA: construction of spline spaces with nice properties suitable for analysis and the problem of volumetric parameterization of computational domain from its boundary representation. Some preliminary results about parameterization method developed in our research are briefly included in this work, however this problem will be studied in depth and presented in another doctoral dissertation of the group.

This thesis addresses the issue of construction of appropriate spline spaces over T-meshes for its application in Isogeometric Analysis. The main contribution of the dissertation is a new strategy for defining spline blending functions that span spaces with nice properties. The technique is designed for hierarchical T-meshes (multilevel meshes) with a quad- and octree subdivision scheme. This type of meshes can be efficiently implemented with tree data structures [51], which are frequently used in engineering. Due to the elevated complexity of all current strategies, the main goal we pursue here is the simplicity and low computational cost, both in 2D and 3D. For that, we have to assume a restriction on the T-mesh. Namely, the T-mesh should fulfil the requirement of being a strongly balanced mesh. Assuming this reasonable and frequently used restriction over the T-mesh, we can define easily cubic spline functions that span spaces with desirable properties: linear independence, $C^2$-continuous, polynomial reproduction property, nested spaces and a straightforward implementation.

The manuscript is organized as follows.

- Chapter 2 introduces some basic concepts about B-splines and NURBS and their use for curve and surface modelling, followed by the description of Isogeometric Analysis concept and its comparison with classical Finite Element Method.

- In Chapter 3 the idea of T-splines is revised and some of their shortcomings are discussed and illustrated. Then, we discuss briefly the available solutions for the problem of local refinement and analyse their advantages and shortcomings that has motivated our research.

- Chapter 4. The core of the dissertation is presented. We describe and illustrate the procedure for the construction of cubic spline space over a given T-mesh. The key of our technique is some simple rules used for inferring local knot vectors to define tensor product spline blending functions that span

spaces with desirable properties. The algorithms for an efficient implementation of the strategy for 2D and 3D meshes are given.

- Properties and some characteristics of the constructed spaces are discussed in Chapter 5. Proof of some properties for 2D case is provided.

- Chapter 6 presents a brief description of another result of our research: a method for spline parameterization of complex 2D geometries, which will be necessary for the computational examples.

- Chapter 7. The approximation properties of the proposed spline spaces are tested in geometric modelling and Isogeometric Analysis. We apply IGA for different types of problems that involve adaptive refinement. Convergence behaviour is analysed. Some computational examples include a comparison with Finite Element Method.

- Chapter 8 exposes the summary, concluding remarks and future research directions.

# CHAPTER 2

# Preliminaries

In this chapter we introduce some basic concepts about spline theory relevant to this thesis, namely, B-splines and NURBS and their use for curve and surface modelling and Isogeometric Analysis. For more comprehensive review we refer the reader to [1, 52, 2]. Then, the Isogeometric Analysis concept is explained in details.

## 2.1 Spline theory. Basic concepts

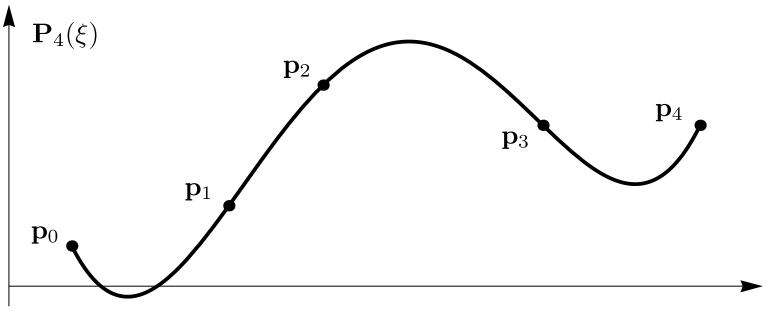
### 2.1.1 Spline interpolation

Data fitting is a fundamental concept for geometric design. A common problem in curve modelling is data interpolation: given data points $\mathbf{p}_i$ and their corresponding parameter values $\xi_i$, find a parametric curve that passes through the points $\mathbf{p}_i$. A well known method to solve this problem is polynomial interpolation, that is, find a $n$-order polynomial $\mathbf{P}_n$ that satisfy interpolation conditions

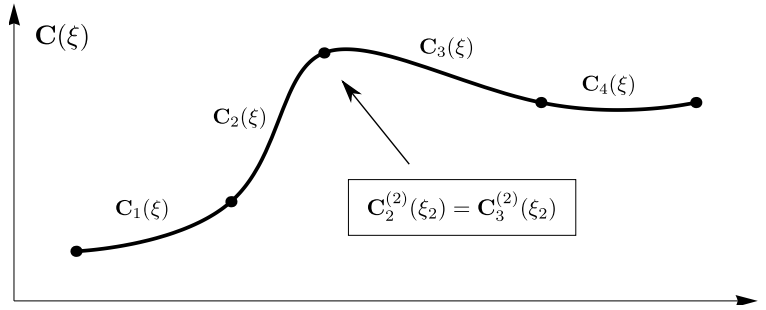
$$\mathbf{P}_n(\xi_i) = \mathbf{p}_i, \quad i = 0, \ldots, n.$$

However, a single segment polynomial interpolation have some shortcomings: a large number of data points requires a high degree polynomial which is inefficient and can present oscillations, known as Runge's phenomenon, and numerical unstability (see [2] for an example of ill-conditioned Lagrange polynomial interpolation). A solution to this problem is piecewise polynomial interpolation. The idea is to construct a polynomial segments $\mathbf{C}_i(\xi)$ for each parameter interval $[\xi_{i-1}, \xi_i]$ so that the resulting piecewise curve $\mathbf{C}(\xi)$ passes through the data points $\mathbf{p}_i$ and the polynomial segments join with a certain level of continuity, i.e., $\mathbf{C}_i^{(k)}(\xi_i) = \mathbf{C}_{i+1}^{(k)}(\xi_i)$. This piecewise polynomial curve $\mathbf{C}(\xi)$ is called *spline*, see Fig. 2.1. The term *spline* comes from the name of flexible rulers that were used for technical drawings and were free to bent to pass through the specified points. Spline curve can be seen as a mathematical generalization of the physical spline.

The set of all piecewise polynomial curves of degree $n$ and some level of




(a) A single 4-th order polynomial fitting data points.




(b) Cubic spline curve fitting the same data points with $C^2$-continuity between the segments.

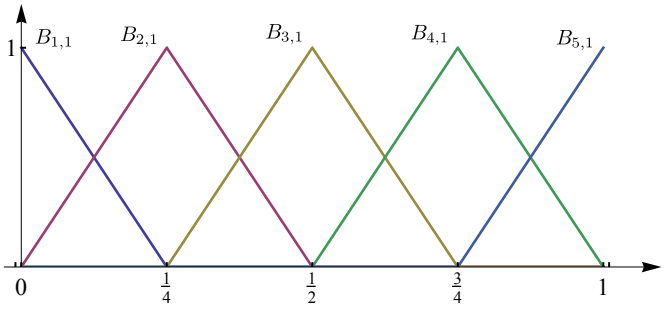
**Figure 2.1:** *Example of data interpolation. (a) Lagrange polynomial interpolation. (b) Piecewise polynomial interpolation.*

continuity over a given sequence of parameter values $\{\xi_i\}$ form a vector space. It is possible to define some elemental piecewise polynomial functions of the same degree and level of continuity that form a basis for this space. Then, any spline curve can be constructed as linear combination of the basis piecewise polynomials. These basis functions are called B-splines (Basis splines).

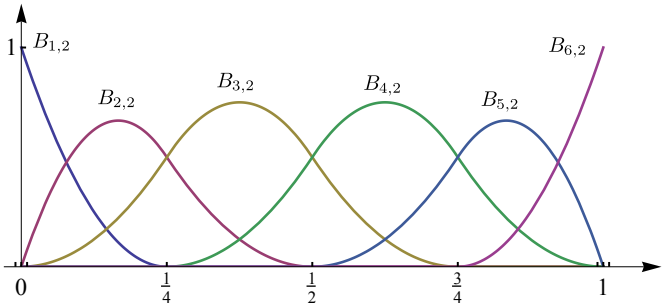
### 2.1.2 B-spline basis functions

Consider a non-decreasing sequence of parameter values $\Xi = \{\xi_1, \xi_2, \ldots, \xi_{n+p+1}\}$. The values $\xi_i$ are called *knots* and the sequence $\Xi$ the *knot vector*. The intervals $(\xi_i, \xi_{i+1})$ are called *knot spans*. Then, we can define a set of B-spline basis functions $\{B_{i,p}\}_{i=1}^{n}$ of degree $p$, inferred from the knot vector $\Xi$, by the Cox-de Boor recurrence formula as follows

$$B_{i,0}(\xi) = \begin{cases} 1 & \text{if} \quad \xi_i \leq \xi < \xi_{i+1}, \\ 0 & \text{otherwise.} \end{cases}$$




(a) A set of first order B-splines inferred from the open knot vector $\Xi = \{0, 0, 1/4, 1/2, 3/4, 1, 1\}$.




(b) A set of quadratic B-splines inferred from the open knot vector $\Xi = \{0, 0, 0, 1/4, 1/2, 3/4, 1, 1, 1\}$.

**Figure 2.2:** *Examples of linear and quadratic B-spline functions.*

for $p \geq 1$

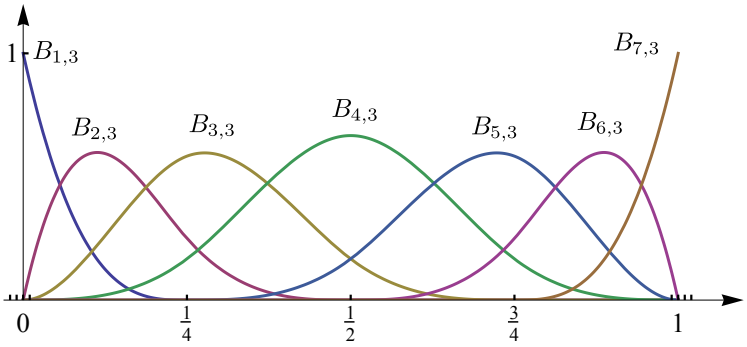
$$B_{i,p}(\xi) = \frac{\xi - \xi_i}{\xi_{i+p} - \xi_i} B_{i,p-1}(\xi) + \frac{\xi_{i+p+1} - \xi}{\xi_{i+p+1} - \xi_{i+1}} B_{i+1,p-1}(\xi),$$

where $(\xi - \xi_i)/(\xi_{i+p} - \xi_i) \equiv 0$ if $\xi_{i+p} = \xi_i$.

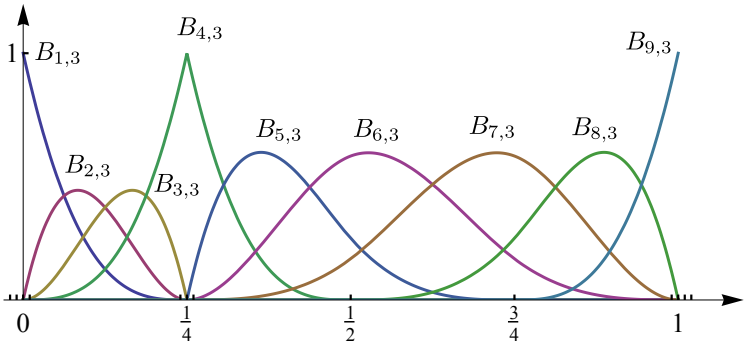
A knot vector $\Xi$ is called *open knot vector* if the first and the last knot are repeated $p+1$ times, see Fig. 2.3(a). If a knot $\xi_i$ is repeated $m$ times ($m \leq p+1$), it is said to have multiplicity $m$, and in this case the basis functions are $C^{p-m}$-continuous at the knot, see Fig. 2.3(b) and Fig. 2.4. Next we enumerate some basic properties of B-spline functions.

**Properties of B-splines**:

- *Local support*: $B_{i,p}(\xi) = 0$ for $\xi$ outside the interval $[\xi_i, \xi_{i+p+1})$.
- *Non-negativity*: $B_{i,p}(\xi) \geq 0$.
- *Partition of unity*: $\sum_{i=1}^{n} B_{i,p}(\xi) = 1$, $\xi \in [\xi_1, \xi_{n+p+1}]$.
- *Linear independence*: $\sum_{i=1}^{n} c_i B_{i,p}(\xi) \equiv 0 \quad \Rightarrow \quad c_i = 0,\ i = 1, \ldots, n.$




(a) A set of cubic B-splines inferred from the open knot vector $\Xi = \{0, 0, 0, 0, 1/4, 1/2, 3/4, 1, 1, 1, 1\}$. No repeated interior knots.




(b) A set of cubic B-splines inferred from the open knot vector $\Xi = \{0, 0, 0, 0, 1/4, 1/4, 1/4, 1/2, 3/4, 1, 1, 1, 1\}$. Knot 1/4 of multiplicity 3 leads to $C^0$-continuity of B-splines at the knot.

**Figure 2.3:** *Examples of cubic B-spline functions.*

- *Except for the case $p = 0$, $B_{i,p}$ attains exactly one maximum.*
- *Each function $B_{i,p}$ is a piecewise polynomial of degree $p$ with $C^{p-m}$ continuity at its knots of multiplicity $m$, see Fig. 2.4.*
- *The derivative of a B-spline function $B_{i,p}$ can be expressed as a linear combination of B-splines of lower degree:*

$$\frac{d^k}{d^k\xi} B_{i,p} = \frac{p}{\xi_{i+p} - \xi_i} \left( \frac{d^{k-1}}{d^{k-1}\xi} B_{i,p-1} \right) - \frac{p}{\xi_{i+p+1} - \xi_{i+1}} \left( \frac{d^{k-1}}{d^{k-1}\xi} B_{i+1,p-1} \right).$$

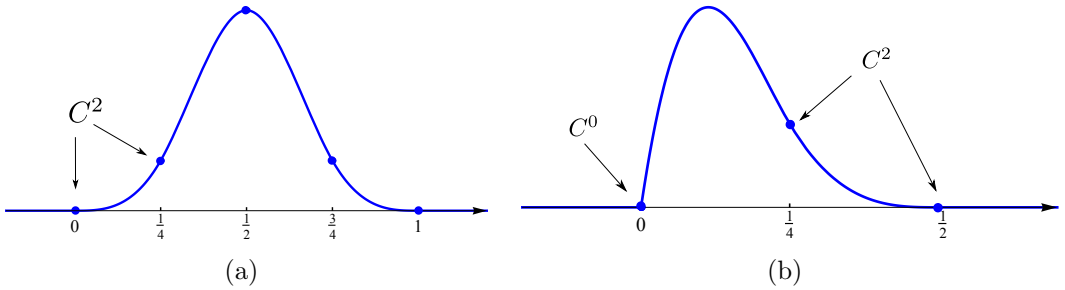



**Figure 2.4:** *Examples of B-spline supports with different regularity at their knots. (a) A cubic B-spline with all the knots of multiplicity 1:* $\{0, 1/4, 1/2, 3/4, 1\}$. *(b) A cubic B-spline, for the knot vector* $\{0, 0, 0, 1/4, 1/2\}$, *is* $C^0$ *at the multiple knot* $\xi_1 = \xi_2 = \xi_3 = 0$.

#### 2.1.2.1 Knot insertion and B-spline refinement

It is known that if a knot $\hat{\xi}$ is inserted into a knot vector $\Xi = (\xi_1, \xi_2, \xi_3, \xi_4, \xi_5)$, then the B-spline $B[\Xi](\xi)$ can be written as linear combination of two new B-spline functions, i.e.,

$$B[\Xi](\xi) = \alpha B[\Xi_1](\xi) + \beta B[\Xi_2](\xi),$$

where $\Xi_1 = (\xi_1, \ldots, \hat{\xi}, \ldots, \xi_4)$ and $\Xi_2 = (\xi_2, \ldots, \hat{\xi}, \ldots, \xi_5)$. Coefficients $\alpha$ and $\beta$ are calculated by the formula

$$\alpha = \begin{cases} 1, & \xi_4 \le \hat{\xi} \le \xi_5 \\ \dfrac{\hat{\xi} - \xi_1}{\xi_4 - \xi_1}, & \xi_1 \le \hat{\xi} \le \xi_4 \end{cases}, \qquad \beta = \begin{cases} \dfrac{\xi_5 - \hat{\xi}}{\xi_5 - \xi_2}, & \xi_2 \le \hat{\xi} \le \xi_5 \\ 1, & \xi_1 \le \hat{\xi} \le \xi_2 \end{cases}. \tag{2.1}$$

For example, for the knot vector $\Xi = (0, 1, 2, 3, 5)$ a new knot $\hat{\xi} = 4$ is inserted between the fourth and the fifth knot. Then, $\Xi_1 = (0, 1, 2, 3, 4)$ and $\Xi_2 = (1, 2, 3, 4, 5)$. And the function is split as $B[\Xi](\xi) = B[\Xi_1](\xi) + \frac{1}{4} B[\Xi_2](\xi)$, see Fig. 2.5.

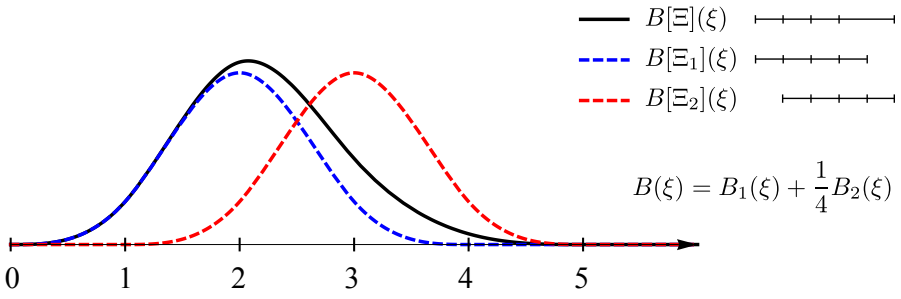


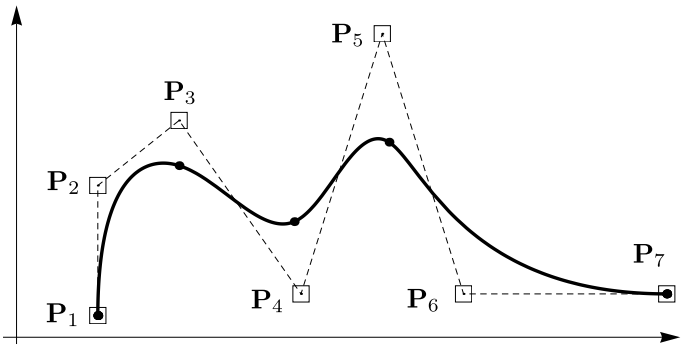

**Figure 2.5:** *Knot insertion operation for B-spline.*

### 2.1.3 B-spline curves

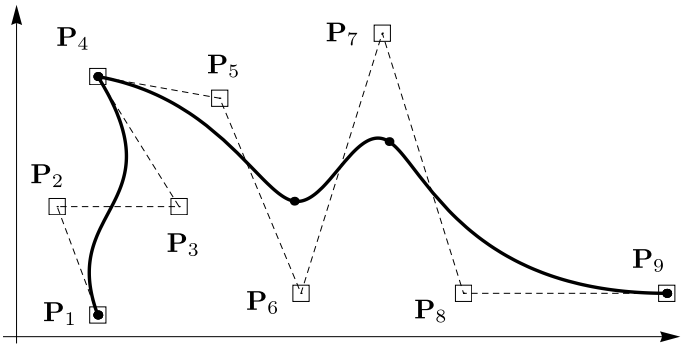
A $p$th-degree B-spline curve is defined as

$$\mathbf{C}(\xi) = \sum_{i=1}^{n} \mathbf{P}_i \, B_{i,p}(\xi),$$

where coefficients $\mathbf{P}_i \in \mathbb{R}^2$ ($\mathbb{R}^3$) are called *control points*, and $\{B_{i,p}\}_{i=1}^{n}$ are the B-spline basis functions defined on the open knot vector $\Xi = \{\xi_1, \xi_2, \ldots, \xi_{n+p+1}\}$. The B-spline curve does not interpolate the control points, except for the first one and the last one, due to open knot vector. The polygon formed by the control points $\mathbf{P}_i$ is called the *control polygon*. Control polygon represents a piecewise linear approximation to the curve. A spline curve attempts to mimic the shape of its control polygon. The continuity of the curve is determined by the continuity of its basis functions. Figure 2.6 shows some examples of B-spline curves. A cubic B-spline curve with $C^2$ regularity is shown in Fig. 2.6(a) and a cubic spline curve with reduced regularity at some point, in order to represent a sharp feature, is given in Fig. 2.6(b).

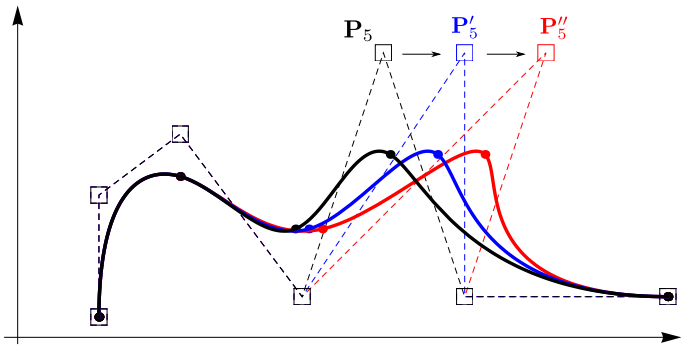



(a) No repeated interior knots.




(b) Multiple knots for representing a sharp feature ($\mathbf{P}_4$).

**Figure 2.6:** *Examples of cubic B-spline curve and its control polygon.*




(a)

**Figure 2.7:** *Local control property of B-spline curve.*

**Properties of B-spline curve**:

- *Convex hull property*: the curve is contained within the convex hull of its control points. This property is useful for checking if two curves intersect each other.
- *Affine invariance*: the curve is invariant under affine transformation of its control points. It implies that an affine transformation of the curve can be obtained by applying the transformation directly to the control points.
- *Variation diminishing property*: no plane has more intersections with the curve than with its control polygon. This property makes the B-spline curve more monotone, compared with oscillating behaviour that can have Lagrange polynomials.
- *Local control property*: moving a control point $\mathbf{P}_i$ affects only locally the shape of the curve $\mathbf{C}(\xi)$, namely only the interval $[\xi_i, \xi_{i+p+1})$, see Fig. 2.7. This property is important for geometric design since it allows to change locally the shape without the necessity to recalculate the entire curve.

#### 2.1.3.1 Refinement

Refinement of a spline curve can be performed by new knots insertion or degree elevation of the curve. The idea is to preserve the original geometry and its parameterization.

##### 2.1.3.1.1 Knot insertion. h-refinement

Let us consider a $p$th-degree B-spline curve $\mathbf{C}(\xi) = \sum_{i=1}^{n} \mathbf{P}_i B_{i,p}(\xi)$, where the B-spline basis functions $\{B_{i,p}\}_{i=1}^{n}$ are defined on the open knot vector $\Xi = \{\xi_1, \xi_2, \ldots, \xi_{n+p+1}\}$.

A new knot $\hat{\xi} \in [\xi_k, \xi_{k+1})$ is inserted, creating a new knot vector

$$\widehat{\Xi} = \{\xi_1, \xi_2, \ldots, \xi_k, \hat{\xi}, \xi_{k+1}, \ldots, \xi_{n+p+1}\}.$$

It is possible then to construct the same curve $\mathbf{C}(\xi)$ using the new basis $\{\widehat{B}_{i,p}\}_{i=1}^{n+1}$ defined on $\widehat{\Xi}$:

$$\mathbf{C}(\xi) = \sum_{i=1}^{n} \mathbf{P}_i B_{i,p}(\xi) = \sum_{i=1}^{n+1} \widehat{\mathbf{P}}_i \widehat{B}_{i,p}(\xi). \tag{2.2}$$

The new control points $\widehat{\mathbf{P}}_i$ are calculated as

$$\widehat{\mathbf{P}}_i = \alpha_i \mathbf{P}_i + (1 - \alpha_i)\mathbf{P}_{i-1},$$

where

$$\alpha_i = \begin{cases} 1, & \text{if} \quad i \leq k - p \\ \dfrac{\hat{\xi} - \xi_i}{\xi_{i+p} - \xi_i}, & \text{if} \quad k - p + 1 \leq i \leq k \\ 0, & \text{if} \quad i \geq k + 1 \end{cases} \tag{2.3}$$

Note that only some of the control points are changed with respect to the original control points $\mathbf{P}_i$. When a new knot $\hat{\xi}$ is inserted into a knot vector $\Xi$, the initial basis functions $\{B_{i,p}\}_{i=1}^{n}$ can be expressed in terms of the new basis functions $\{\widehat{B}_{i,p}\}_{i=1}^{n+1}$ defined on $\widehat{\Xi}$. That is, a knot insertion produces nested spline spaces. Each function $B_{i,p}$ is expressed as linear combination of the refined functions. Inserting these expressions into (2.2) leads to the formula (2.3) for the new control points.

#### 2.1.3.1.2 Degree elevation. p-refinement

When elevating the polynomial order of the basis functions, the inter-element regularity should be preserved. So the multiplicity of the original knots should be increased by one before degree elevation. No new knots are inserted. In this way, degree elevation also produces nested spaces and the original curve $\mathbf{C}(\xi)$ can be preserved.

### 2.1.4 B-spline surfaces

Bivariate B-splines are defined as a tensor product of univariate B-spline functions.

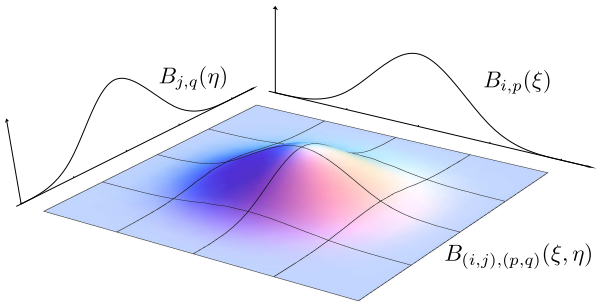
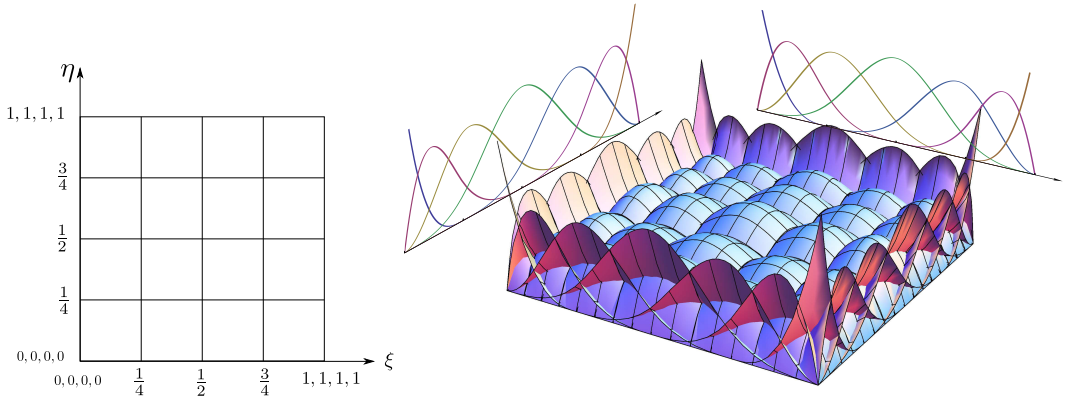



**Figure 2.8:** *Bivariate B-spline function.*

Let $\{B_{i,p}(\xi)\}_{i=1}^{n_1}$ and $\{B_{j,q}(\eta)\}_{j=1}^{n_2}$ be two sets of B-spline basis functions of degree $p$ and $q$, defined over knot vectors $\Xi = \{\xi_1, \xi_2, \ldots, \xi_{n_1+p+1}\}$ and $\mathcal{H} = \{\eta_1, \eta_2, \ldots, \eta_{n_2+q+1}\}$, respectively, which form a Cartesian grid in parametric domain. Then, bivariate basis functions are defined as follows

$$B_{(i,j),(p,q)}(\xi,\eta) = B_{i,p}(\xi)B_{j,q}(\eta), \quad (i,j) \in I,$$

where the multi-index set $I$ is defined by $I = \{1, 2, \ldots, n_1\} \times \{1, 2, \ldots, n_2\}$. See an example of bivariate B-spline function in Fig. 2.8 and an example of a set of B-spline functions defined over a parametric grid in Fig. 2.9.




(a) (b)

**Figure 2.9:** *Bivariate cubic B-spline functions defined over a parametric grid formed by the knot vectors* $\Xi = \mathcal{H} =$*{0,0,0,0,1/4,1/2,3/4,1,1,1,1}.*

Bivariate B-spline functions possess the same properties as the univariate B-splines:

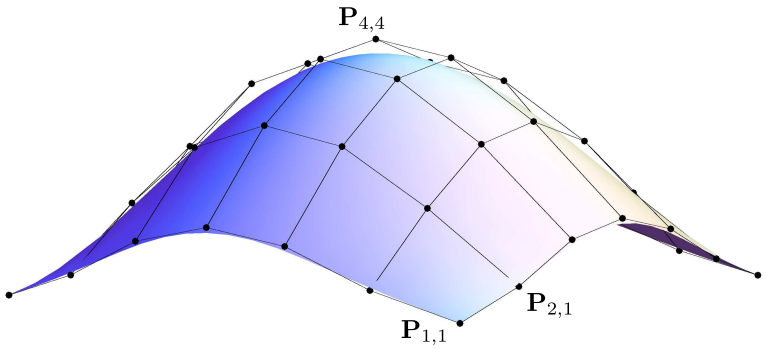



**Figure 2.10:** *B-spline surface and its control net.*

- *Local support*: $B_{(i,j),(p,q)}(\xi,\eta) = 0$ for $\xi$ outside the rectangle $[\xi_i, \xi_{i+p+1}) \times [\eta_j, \eta_{j+q+1})$.
- *Non-negativity*: $B_{(i,j),(p,q)}(\xi,\eta) \geq 0$.
- *Partition of unity*: $\sum_{(i,j)\in I} B_{(i,j),(p,q)}(\xi,\eta) = 1$, $(\xi,\eta) \in [\xi_1, \xi_{n_1+p+1}] \times [\eta_1, \eta_{n_2+q+1})$.
- *Linear independence*: $\sum_{(i,j)\in I} c_{i,j} B_{(i,j),(p,q)} \equiv 0 \quad \Rightarrow \quad c_{i,j} = 0$, $(i,j) \in I$.
- *Except for the case* $p = 0$ *or* $q = 0$, $B_{(i,j),(p,q)}$ *attains exactly one maximum.*
- *Each function* $B_{(i,j),(p,q)}$ *is a piecewise bivariate polynomial. It is* $(p - m)$ *times differentiable in the* $\xi$*-direction, where* $m$ *is the multiplicity of the* $\xi$ *knot. Analogously for* $\eta$*-direction.*

A B-spline surface is defined as a linear combination of bivariate B-spline functions

$$\mathbf{S}(\xi,\eta) = \sum_{(i,j)\in I} \mathbf{P}_{i,j}\, B_{(i,j),(p,q)}(\xi,\eta),$$

where the control points $\mathbf{P}_{\boldsymbol{i}} \in \mathbb{R}^3$ form a *control net*. See an example of B-spline surface in Fig. 2.10. Analogously to B-spline curve, the B-spline surface does not interpolate the control points, except for the four corner control points. If triangulated, the control net forms a piecewise planar approximation to the surface. Also, B-spline surface conserves some properties of the spline curve: convex hull property, affine invariance and local control property, but no variation diminishing property.

**Remark 1.** *B-spline solids are defined in analogous way, as a linear combination of trivariate B-spline basis functions and inherit the properties of its univariate and bivariate counterpart.*

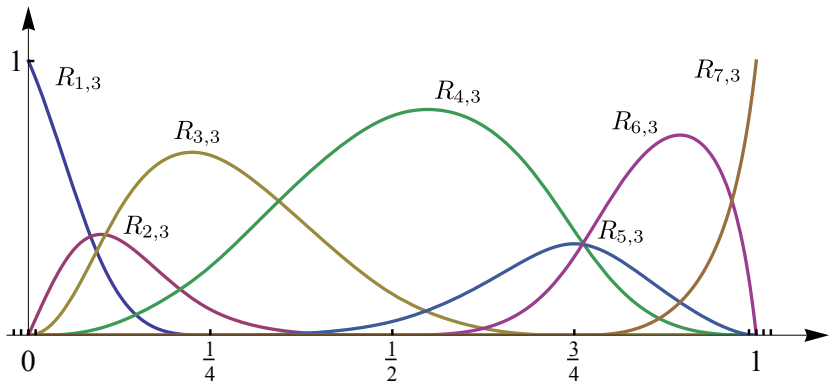



**Figure 2.11:** *Example of NURBS functions derived from B-splines using the weights* $w_1 = 2, w_2 = 1, w_3 = 3, w_4 = 4, w_5 = 1, w_6 = 2$ *and* $w_7 = 1$.

### 2.1.5 Non-Uniform Rational B-splines (NURBS)

Although polynomial splines offer many advantages for the free-form modelling, there exist some important shapes which cannot be represented exactly with B-spline functions, e.g., circles, ellipses, cylinders, cones, spheres. This inconvenience can be solved using rational functions.

A set of Non-Uniform Rational B-splines functions $\{R_{i,p}\}$ for a given knot vector $\Xi = \{\xi_1, \xi_2, \ldots, \xi_{n+p+1}\}$ is defined from a set of B-spline basis functions $\{B_{i,p}\}$ as follows

$$R_{i,p}(\xi) = \frac{w_i B_{i,p}(\xi)}{\sum_{j=1}^{n} w_j \, B_{j,p}(\xi)},$$

where $\{w_i\}$ is a set of $n$ positive weights, see Fig. 2.11.

Note that if $w_j = 1$ for all $j$, then $R_{i,p} = B_{i,p}$. So, polynomial B-spline functions are special case of the NURBS functions $\{R_{i,p}\}$.

In a similar fashion to B-spline curves, NURBS curve is defined as linear combination of the rational basis functions

$$\mathbf{C}(\xi) = \sum_{i \in I} \mathbf{P}_i \, R_{i,p}(\xi),$$

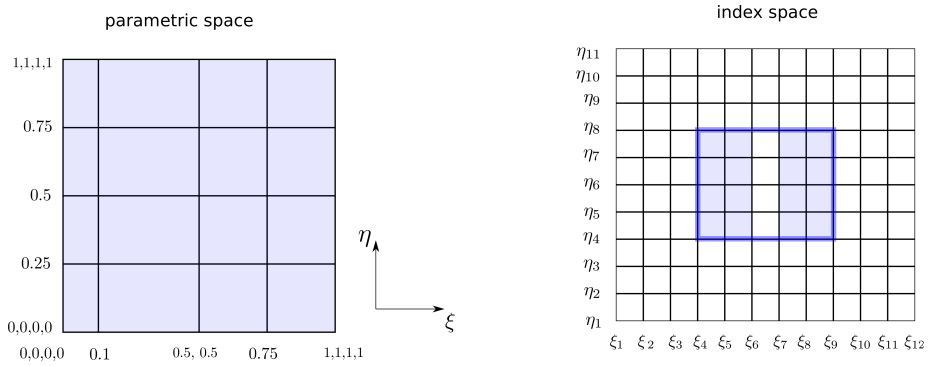
where $\mathbf{P}_i \in \mathbb{R}^2$ are the control points.

A NURBS surface is defined as a linear combination of bivariate NURBS functions

$$\mathbf{S}(\xi, \eta) = \sum_{(i,j) \in I} \mathbf{P}_{i,j} \, R_{(i,j),(p,q)}(\xi, \eta),$$

where $R_{(i,j),(p,q)}(\xi, \eta)$ are bivariate rational basis functions defined as

$$R_{(i,j),(p,q)}(\xi, \eta) = \frac{w_{i,j} B_{(i,j),(p,q)}(\xi, \eta)}{\sum_{(i,j) \in I} w_{i,j} \, B_{(i,j),(p,q)}(\xi, \eta)}.$$




$$\Xi = \{\xi_1, \xi_2, \xi_3, \xi_4, \xi_5, \xi_6, \xi_7, \xi_8, \xi_9, \xi_{10}, \xi_{11}, \xi_{12}\} = \{0,0,0,0,0.1,0.5,0.5,0.75,1,1,1,1\}$$
$$H = \{\eta_1, \eta_2, \eta_1, \eta_2, \eta_3, \eta_4, \eta_5, \eta_6, \eta_7, \eta_8, \eta_9, \eta_{10}, \eta_{11}\} = \{0,0,0,0,0.25,0.5,0.75,1,1,1,1\}$$

**Figure 2.12:** *Index space. Shaded zone corresponds to non-zero knot spans.*

Choosing an appropriate knot vector and values of the weights $\{w_i\}$ for NURBS enables to represent exactly conic sections. NURBS and NURBS curves (surfaces) inherit the properties of B-spline functions and B-spline curves(surfaces), respectively. For more details about NURBS we refer the reader to [1].

### 2.1.6 Index space and parametric space

Since some knots of the parametric space can have the same values, for some purpose it is useful to use the *index space*. In contrast to parametric space, where multiple knots are plotted as the same point, the index space is a uniform grid, where each knot is depicted as the unique knot, taking into account only the index of the knot. In the index space all knot intervals have the same length, while in parametric space some intervals can have zero length. Figure 2.12 illustrates an example of parametric grid and its corresponding index grid. Knot vector $\Xi$ contains an interior multiple knot $\xi_6 = \xi_7 = 0.5$ and repeated boundary knots. Knot vector $\mathcal{H}$ has repeated knots only on the boundary.

## 2.2 Isogeometric Analysis

In this section we revise the idea of Isogeometric Analysis. Assuming that the reader is familiar with the Finite Element Method, we focus our attention on the similarities and the main differences between IGA and Finite Element Analysis. The main idea of Isogeometric Analysis is to use for the solution space the same basis functions that are used to model the geometry. This notion is called isoparametric concept, and it is common for IGA and FEA. However, *"The fundamental difference between this new concept of Isogeometric Analysis and the old concept of isoparametric finite element analysis is that, in classical FEA, the basis chosen to approximate the* unknown *solution fields is then used to approximate* known *geometry. Isogeometric Analysis turns this idea around and selects a basis capable of exactly representing the known geometry and uses it as a basis for the fields we wish to approximate. In a sense, we are reversing the isoparametric arrow such that it points from the geometry toward the solution space, rather that vice versa."* (Cottrell et al. in [42].)

### 2.2.1 Variational formulation and Galerkin's projection method

Theoretical framework for both, Finite Element Analysis and Isogeometric Analysis, is based on two main steps: variational (weak) formulation of the problem and Galerkin's projection method to approximate the solution of the weak problem. Next, we revise the method using the Poisson model problem.

Let us consider a strong form of the boundary value problem

$$\begin{aligned} -\Delta u &= f \qquad \text{in } \Omega, \\ u &= g \qquad \text{on } \Gamma_D, \\ \nabla u \cdot \mathbf{n} &= h \qquad \text{on } \Gamma_N, \end{aligned} \tag{2.4}$$

where $\Omega$ is a Lipschitz domain with boundary $\partial\Omega = \Gamma = \Gamma_D \cup \Gamma_N$, and the functions $f : \Omega \to \mathbb{R}$, $g : \Gamma_D \to \mathbb{R}$, $h : \Gamma_N \to \mathbb{R}$ are given.

Let $V$ be a suitable Hilbert space $V = H^1(\Omega) := W_2^1(\Omega)$, where the Sobolev space $W_2^k(\Omega) = \{v \in L^2(\Omega) : \partial^\alpha v \in L^2(\Omega),\ \forall\ |\alpha| \leq k\}$. Then, the space $V = H^1(\Omega)$ is equipped with the norm

$$\|v\|_V = \left( \sum_{|\alpha| \leq 1} \int_\Omega (\partial^\alpha v)^2 \, d\Omega \right)^{1/2}$$

and seminorm

$$|v|_V = \left( \sum_{|\alpha| = 1} \int_\Omega (\partial^\alpha v)^2 \, d\Omega \right)^{1/2}.$$

The test function space $V_0(\Omega)$ is defined as

$$V_0(\Omega) = \{v \in H^1(\Omega) : v \mid_{\Gamma_D} = 0\},$$

and the solution space $V_g(\Omega)$ is the set of functions from $V$ which fulfil the essential boundary conditions, i.e.,

$$V_g(\Omega) = \{v \in H^1(\Omega) : v \mid_{\Gamma_D} = g\}.$$

If a function $u$ satisfies the equation (2.4), it also satisfies

$$\int_\Omega -\Delta u \, v \, \mathrm{d}\Omega = \int_\Omega f \, v \, \mathrm{d}\Omega, \quad \forall v \in V_0(\Omega).$$

Applying the Green's formula for integration by parts leads to

$$\int_\Omega \nabla u \cdot \nabla v \, \mathrm{d}\Omega = \int_\Omega f \, v \, \mathrm{d}\Omega + \int_{\Gamma_N} h \, v \, \mathrm{d}\Gamma + \int_{\Gamma_D} \frac{\partial u}{\partial \mathbf{n}} \, v \, \mathrm{d}\Gamma, \quad \forall v \in V_0(\Omega),$$

where the last term vanishes.

Then, variational formulation for the problem (2.4) consists in finding $u \in V_g(\Omega)$ such that

$$a(u, v) = F(v), \quad \forall v \in V_0(\Omega), \tag{2.5}$$

where $a(\cdot, \cdot)$ is bilinear symmetric form

$$a(u, v) = \int_\Omega \nabla u \cdot \nabla v \, \mathrm{d}\Omega$$

and $F(\cdot)$ is a linear functional

$$F(v) = \int_\Omega f \, v \, \mathrm{d}\Omega + \int_{\Gamma_N} h \, v \, \mathrm{d}\Gamma.$$

Under certain assumptions the variational problem (2.5) has a unique solution. Namely,

**Definition 1.** A bilinear form $a(\cdot, \cdot)$ on a normed vector space $(V, \|\cdot\|)$ is *coercive* (elliptic), if there exist a constant $\alpha > 0$ such that $|a(u, u)| \geq \alpha \|u\|^2, \quad \forall u \in V.$

**Definition 2.** A bilinear form $a(\cdot, \cdot)$ is called bounded (continuous), if there exist a constant $M$ such that $|a(u, v)| \leq M \|u\| \|v\|, \quad \forall u, v \in V.$

Then, the existence and uniqueness of a solution for the problem (2.5) is guaranteed by the Lax-Milgram theorem.

**Theorem 1** (Lax-Milgram)**.** *Let $V$ be a Hilbert space, $a(\cdot, \cdot) : V \times V \to \mathbb{R}$ is a bounded, coercive bilinear form, and $F(\cdot) : V \to \mathbb{R}$ is a continuous linear functional. Then, there exists a unique $u \in V$ such that $a(u, v) = F(v), \quad \forall v \in V.$*

Next, we apply Galerkin method. To approximate the solution of the weak problem (2.5) we construct a finite-dimensional subspaces $V_{0,h}$ and $V_{g,h}$ defined over the discretized domain $\Omega_h$. Let $V_h = \text{span}\{\phi_1, \phi_2, \ldots, \phi_N\}$ be a finite-dimensional space spanned by basis functions defined over the discretization $\Omega_h$. Then, $V_{0,h} \subset V_h$ is a subspace of functions that vanish on the boundary and $V_{g_h,h}$ is the subspace of functions that are equal to $g_h$ on the boundary, where $g_h$ is an interpolant of $g$. Then, finite-dimensional problem consists in finding $u_h \in V_{g_h,h}$ such that

$$a(u_h, v_h) = F(v_h), \quad \forall v_h \in V_{0,h}(\Omega). \tag{2.6}$$

Under the conditions of Theorem 1, there exists a unique $u_h$ that solves the discrete problem (2.6). We search $u_h \in V_{g_h,h}$ as

$$u_h = \sum_{i=1}^{n} u_i \, \phi_i + g_h, \tag{2.7}$$

where $\{\phi_i\}_{i=1}^n \subset V_{0,h}$. Substituting (2.7) into (2.6), Galerkin method gives rise to a linear algebraic system

$$\sum_{i=1}^{n} u_i \, a(\phi_i, \phi_j) + a(g_h, \phi_j) = F(\phi_j) \quad \forall j = 1, ..., n.$$

That is,

$$\mathbf{K}\mathbf{u} = \mathbf{f}, \qquad \text{where}$$

$\mathbf{K} = [\mathbf{K}_{j,i}]_{j,i=1}^n$ is the stiffness matrix, being $\mathbf{K}_{j,i} = a(\phi_i, \phi_j)$;
$\mathbf{f} = (F(\phi_j) - a(g_h, \phi_j))_{j=1}^n$ is the load vector, and
$\mathbf{u} = (u_1, u_2, \ldots, u_n)^T$ are the unknown coefficients called *degrees of freedom*.

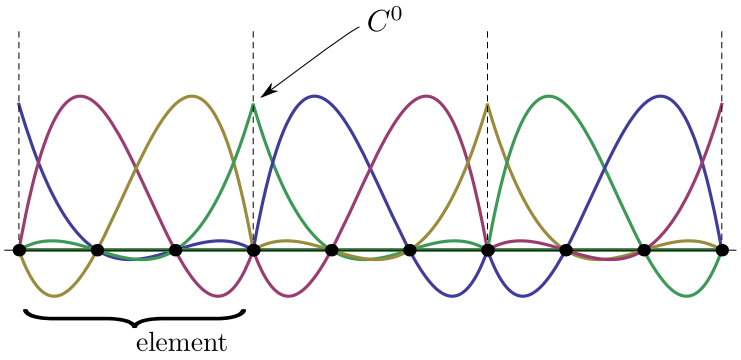
Under the assumptions of Theorem 1, the Céa's Lemma provides a bound of the error in norm $\|\cdot\|_V$ between the exact solution and the approximate solution obtained by Galerkin projection:

$$\|u - u_h\|_V \leq \frac{M}{\alpha} \inf_{v_h \in V_h} \|u - v_h\|_V .$$

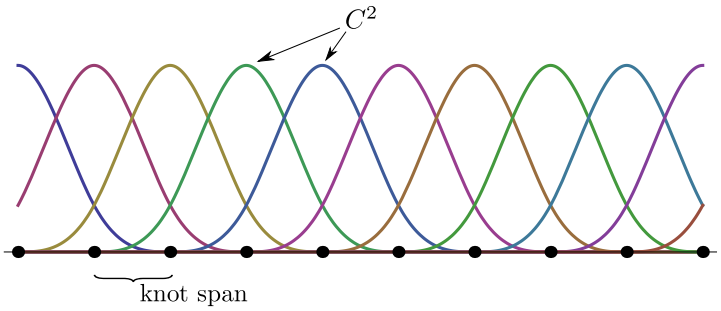
The term $\inf_{v_h \in V_h} \|u - v_h\|_V$ measures the capacity of the finite dimensional space $V_h$ to approximate the function $u$, and it is typically called an interpolation error. This error, in its turn, converges in terms of the mesh-size parameter $h$ with a certain order $s$, which depends on the smoothness of $u$ and the degree of basis functions of the space $V_h$:

$$\inf_{v_h \in V_h} \|u - v_h\|_V \leq C(u) h^s.$$

Thus, the Lemma asserts that the error of the finite element solution is of the same




(a) FEM basis. Lagrange functions.




(b) IGA basis. B-spline functions.

**Figure 2.13:** *Comparison of basis functions for FEM and IGA.*

order that the interpolation error:

$$\|u - u_h\|_V \leq \frac{M}{\alpha} C(u) h^s.$$

This bound on the error is called a priori error bound and it shows that finite element solutions $u_h$ converges to $u$ in $H^1$-norm as $h \to 0$.

### 2.2.2 Basis functions

The main difference between Finite Element Analysis and IGA lies in the type of basis functions used for discrete approximation space. In FEM the basis functions are defined locally on each *element*, which is key concept of the method. The discrete solution space $V_h$ is spanned by the union of all basis functions defined on each element of the mesh, see Fig. 2.13(a). In IGA, however, the set of basis functions (B-splines, NURBS) is inferred from a global knot vector, which represents a discretization of the domain, Fig. 2.13(b). The knots partition the domain into the *knot spans* which is equivalent to the concept of *element* in FEA in the sense that it represents a discretization unit, where the functions are $C^\infty$-continuous. However, it is important to emphasize that a function support is not contained

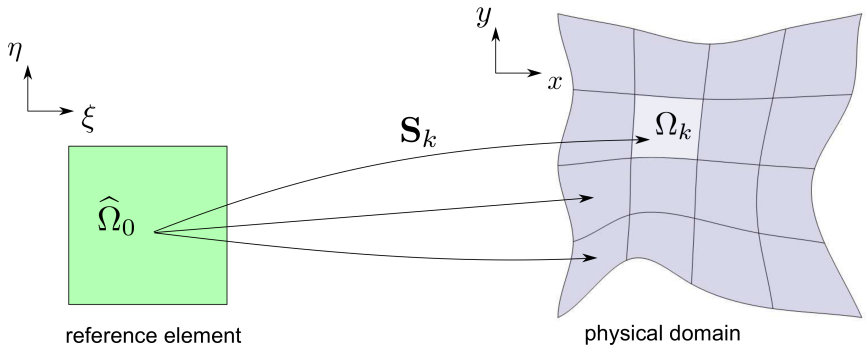



**Figure 2.14:** *Local parametric mapping for each element in Finite Element Method.*

just in one knot span, but it covers a several knot spans.

The most important difference that provide spline basis functions, with respect to the functions used in classical Finite Element Method, is a higher continuity. In FEA any order basis functions are only $C^0$ across the element boundary. Meanwhile, $p$th-degree spline basis functions are $C^{p-1}$-continuous across the element boundary. Figure 2.13 illustrates this difference. For example, cubic B-splines possess $C^2$-continuity (no repeated interior knots). It permits to construct numerical solution that is globally $C^2$-continuous which is, in general, not possible with classical Lagrange type basis functions used in FEM.

### 2.2.3 Domain parameterization

As a consequence of the global definition of the basis function in IGA, another difference, with respect to FEM, is the paremeterization of computational domain. In Finite Element Method each element of the physical domain has its own parametric mapping that transforms the reference element into the physical one. This local parameterization depends only on the nodes of the physical element and it is constructed with the basis functions defined on the reference element, see an illustration for 2D case in Fig. 2.14. For each physical element $\Omega_k$ a parametric mapping $\mathbf{S}_k : \widehat{\Omega}_0 \to \Omega_k$ is defined as a linear combination

$$\mathbf{S}_k(\xi, \eta) = \sum_j \mathbf{P}_j^k \, \hat{\phi}_j(\xi, \eta),$$

where $\{\hat{\phi}_j\}$ is the basis on the unique reference element $\widehat{\Omega}_0$ and $\mathbf{P}_j^k \in \mathbb{R}^2$. Then, the basis function $\phi_i^k$ on the physical element $\Omega_k$ is defined as $\phi_i^k = \hat{\phi}_i \circ \mathbf{S}_k^{-1}$, where $\hat{\phi}_i$ is the corresponding function on the reference element $\widehat{\Omega}_0$.

In Isogeometric Analysis, in contrast, the parameterization of computational domain is defined globally. A unique global parametric mapping $\mathbf{S} : \widehat{\Omega} \to \Omega$

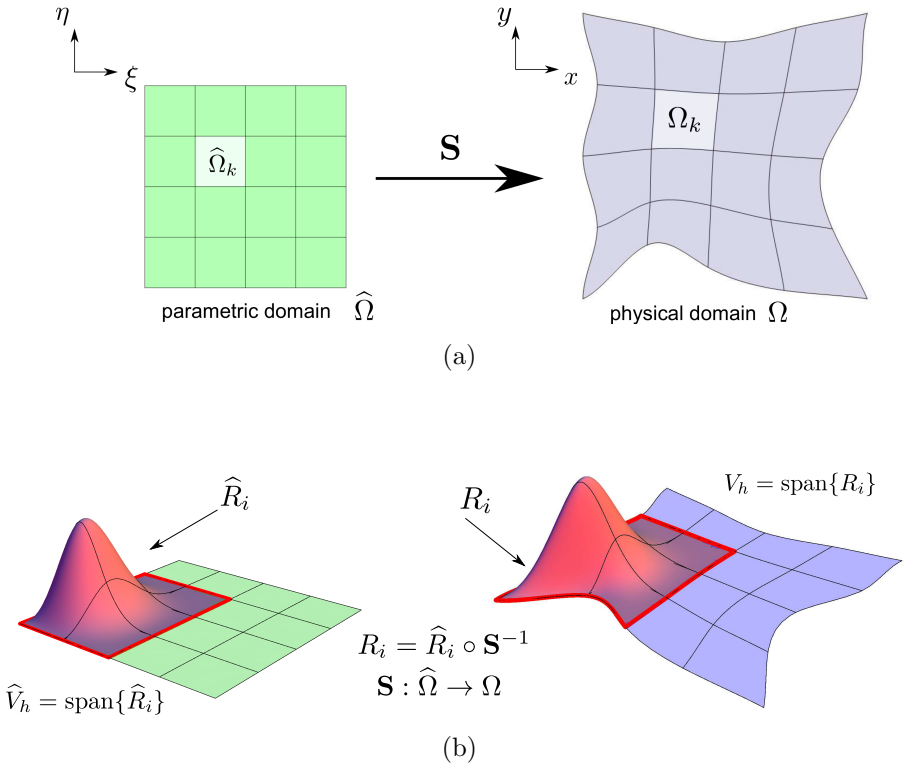



**Figure 2.15:** *(a) Global parametric mapping in Isogeometric Analysis. (b) Basis function in parametric and physical space.*

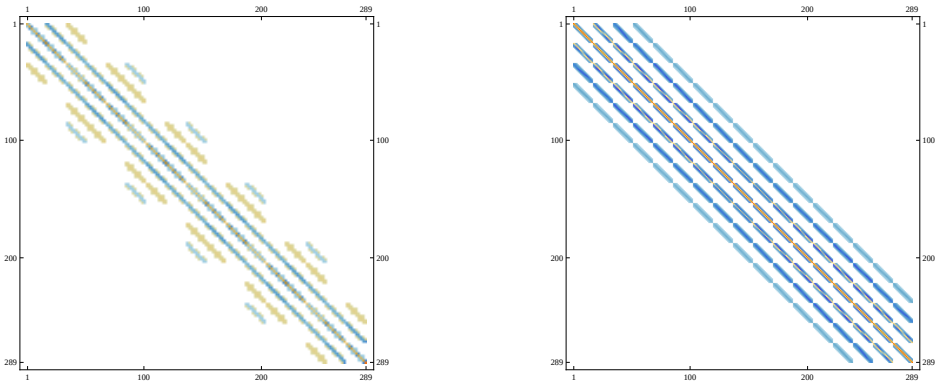
transforms the parametric domain into the physical domain and it is defined as

$$\mathbf{S}(\xi, \eta) = \sum_j \mathbf{P}_j \, \widehat{R}_j(\xi, \eta),$$

where $\{\widehat{R}_j\}$ is the set of NURBS basis functions defined on the whole parametric domain $\widehat{\Omega}$, which is a rectangular grid formed by two global knot vectors $\Xi$ and $\mathcal{H}$, see Fig. 2.15(a). Each element $\Omega_k$ in physical domain has its own counterpart $\widehat{\Omega}_k$ in the parametric domain. And the parametric mapping $\mathbf{S}$ restricted to $\Omega_k$ depends not only on the control points associated with the element but also on the neighbouring elements. The discrete space spanned by the basis functions defined in parametric space, we denote as $\widehat{V}_h = \text{span}\{\widehat{R}_i\}$. Once the parameterization is constructed, the basis function $R_i$ in physical space is defined as $R_i = \widehat{R}_i \circ \mathbf{S}^{-1}$, where $\widehat{R}_i$ is the corresponding function in parametric space, see Fig. 2.15(b). Thus, the discrete approximation space on physical domain is

$$V_h = \text{span}\{R_i \in H^1(\Omega) : R_i = \widehat{R}_i \circ \mathbf{S}^{-1}, \text{for all } \widehat{R}_i \in \widehat{V}_h\}.$$

Then, numerical solution is searched as $u_h = \sum_i c_i \, R_i$, where $R_i \in V_h$.



(a) FEM. Lagrange basis. (b) IGA. NURBS basis.

**Figure 2.16:** *Sparsity of the stiffness matrix for FEA (a) and IGA (b) using degree* 3 *basis functions. Bandwidth is 49 for both cases.*

**Remark 2.** *Note that, unlike in FEM, the coefficients $c_i$, that are called control variables, do not represent the value of the function $u_h$ at the node, since spline basis functions do not posses Kronecker delta property. To evaluate the function value at any point, the basis function supports that intersect with the point should be determined and the values of corresponding basis functions are calculated to compound the value of $u_h$.*

### 2.2.4 System assembling

To build the stiffness matrix $\mathbf{K}$ and load vector $\mathbf{f}$, some integrals are to be calculated. Analogously to FEA, the global integrals are evaluated by summing up the contribution from each element of the domain disctretization. To calculate the integrals over a given element, only the basis functions whose support intersects this element are used. The number of basis functions on each element is determined by the functions order $p$, it is equal to $(p+1)^d$ for $d$-dimensional domain. Both for IGA and FEA (quadrilateral elements), the bandwidth of the stiffness matrix using $p$-order basis functions will be $(2p+1)^d$. Figure 2.16 illustrates examples of the stiffness matrix arisen from both methods using bicubic basis functions. Matrix of Fig. 2.16(b) corresponds to the Isogeometric Analysis over a unit square domain with discretization formed by $16 \times 16$ uniform grid, which gives rise to 289 B-spline test functions. Matrix of Fig. 2.16(a) corresponds to the Finite Element Analysis over the same domain, disctretized by $6 \times 6$ uniform grid, which gives rise to 289 Lagrange tensor product test functions.

To perform integration over the physical element $\Omega_e$, the integral is pulled back into the parametric element $\widehat{\Omega}_e = [\xi_i, \xi_{i+1}] \times [\eta_j, \eta_{j+1}]$ via parametric mapping $\mathbf{S}$, and then from the parametric element it is pulled back to the parent element $\widetilde{\Omega} = [-1, 1] \times [-1, 1]$ via an affine mapping $\phi$, see Fig. 2.17.

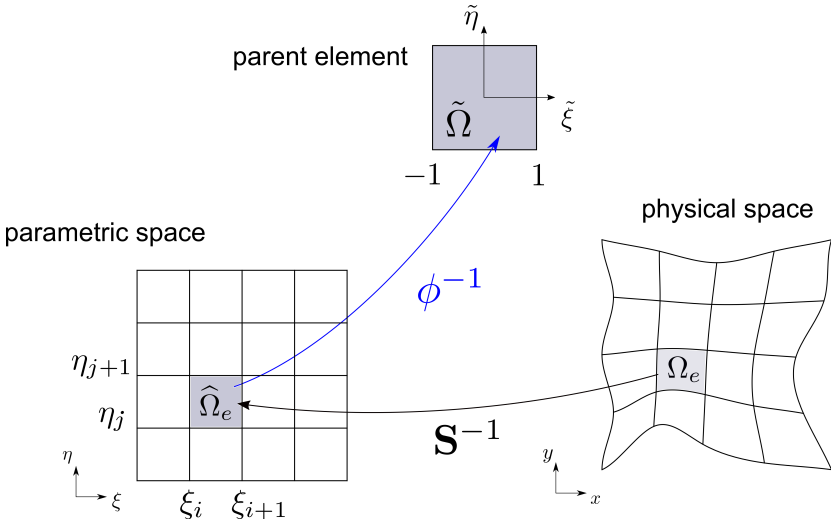



**Figure 2.17:** *Integration over a physical element. The integral is pulled back into the parametric element, and then from the parametric element it is pulled back to the parent element.*

To pull back the integrals from physical space to the parametric one, an appropriate change of variables is necessary. Let us consider the parametric mapping $\mathbf{S} : (\xi, \eta) \to (x(\xi, \eta), y(\xi, \eta))$ and its Jacobian matrix

$$\mathbf{J} = \begin{pmatrix} \partial_\xi x & \partial_\eta x \\ \partial_\xi y & \partial_\xi y \end{pmatrix}.$$

Then, the gradient of the basis function $R_i$ with respect to physical space coordinates is given by

$$\nabla R_i = \hat{\nabla} \widehat{R}_i \, \mathbf{J^{-1}},$$

where $\hat{\nabla} \widehat{R}_i = (\partial_\xi \widehat{R}_i, \partial_\eta \widehat{R}_i)$ and $\nabla R_i = (\partial_x R_i, \partial_y R_i)$. Then, the entry $\mathbf{K}^e_{i,j}$ of the stiffness matrix over an element $\widehat{\Omega}_e$ is calculated as follows

$$\mathbf{K}^e_{i,j} = \int_{\Omega_e} \nabla R_j \cdot \nabla R_i \, d\Omega = \int_{\widehat{\Omega}_e} \hat{\nabla} \widehat{R}_j \, \mathbf{J^{-1}} \cdot \hat{\nabla} \widehat{R}_i \, \mathbf{J^{-1}} \, |\mathbf{J}| \, d\widehat{\Omega}.$$

# CHAPTER 3

## Local refinement problem

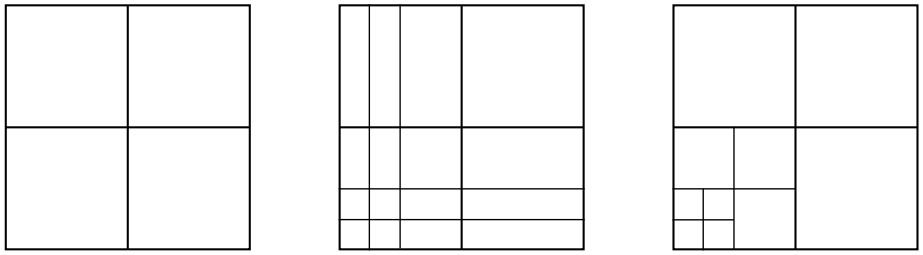
In this chapter we discuss the problem of local refinement for B-splines and analyse some alternatives for performing local enrichment of spline spaces. Namely, we revise the idea of T-splines and the hierarchical refinement scheme.

## 3.1 Limitation of NURBS

To achieve a desired accuracy of the numerical solution with as few basis functions as possible, it is necessary to identify, using some a posteriori error estimation, the areas that contribute the most to the global error and increase the number of basis functions in this area. Both for analysis and design, it is of great importance to perform efficient local enrichment of approximation space. NURBS functions is the most common tool used by CAD software at the moment, and originally Isogeometric Analysis concept was proposed and tested with NURBS. However, it suffers from an important drawback: their tensor product structure does not allow for local refinement. NURBS surface is defined over a rectangular grid. A new knot insertion induces the insertion of a knot line extended through the entire domain, see Fig. 3.1(b). T-splines were proposed by Sederberg et al. [21] as an alternative to NURBS that permits local refinement. T-splines can be seen as a generalization of NURBS. T-splines allows the insertion of a partial line of new knots that does not propagate throughout the domain, which makes them locally refinable, see Fig. 3.1(c).

## 3.2 T-meshes and T-splines

To overcome the drawback of tensor product structure, it is necessary to admit *T-junctions* in the mesh. The concept *T-junction* is similar to *hanging node* in the classical Finite Element Method. An axes-aligned grid that allows T-junctions is called *T-mesh*. The underlying idea of T-splines consists in defining the blending functions by means of a set of *local knot vectors* instead of a global knot vector, as



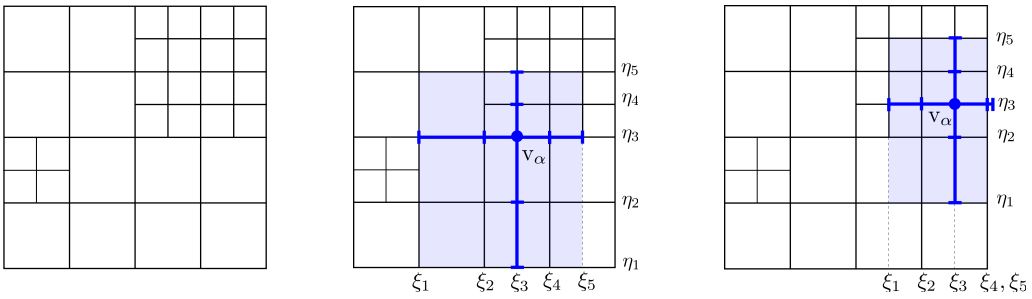
(a) initial mesh (b) global refinement (c) local refinement

**Figure 3.1:** *Refinement of B-splines. (b) A knot insertion induces the insertion of an entire knot line. (c) Local refinement with T-mesh.*

in the case of B-splines or NURBS. To define a set of basis functions for a given T-mesh, to each vertex (node) $\mathrm{v}_\alpha$ of the mesh a blending function $B_\alpha$ is associated. The vertex $\mathrm{v}_\alpha$ is called *anchor* of the function. Next, for simplicity, we are going to explain the idea for the case of cubic T-spline functions over 2D T-meshes. Local knot vectors $\Xi_\alpha = (\xi_1, \xi_2, \xi_3, \xi_4, \xi_5)$ and $\mathcal{H}_\alpha = (\eta_1, \eta_2, \eta_3, \eta_4, \eta_5)$ for each bivariate function $B_\alpha$ are inferred by marching through the T-mesh edges in both parametric directions, starting from its anchor $\mathrm{v}_\alpha = (\xi_3, \eta_3)$, until a perpendicular edge is encountered, see Fig. 3.2(b). When, during this marching the boundary is reached, the knots are repeated to create an open knot vector structure along the boundary, see Fig. 3.2(c).

For a pair of local knot vectors $\Xi_\alpha$ and $\mathcal{H}_\alpha$ the bicubic spline function $B_\alpha$ is defined as $B_\alpha(\xi, \eta) = B[\Xi_\alpha](\xi)\, B[\mathcal{H}_\alpha](\eta)$, where $B[\Xi_\alpha](\xi)$ and $B[\mathcal{H}_\alpha](\eta)$ are the univariate B-splines corresponding to the knot vector $\Xi_\alpha$ and $\mathcal{H}_\alpha$, respectively.

T-splines should be normalized in order to form a partition of unity. This




(a) (b) (c)

**Figure 3.2:** *An example of T-mesh and inferring of local knot vectors for bicubic T-spline functions by traversing T-mesh edges.*

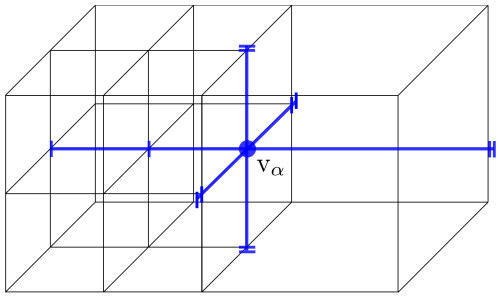



**Figure 3.3:** *3D T-mesh. Inferring local knot vectors for a trivariate T-spline function.*

leads to rational blending functions:

$$R_\alpha(\xi,\eta) = \frac{B_\alpha(\xi,\eta)}{\sum_{\beta\in A_T} B_\beta(\xi,\eta)},$$

where $A_T$ is the index set of the basis spanned by the T-mesh $T$. A T-spline approximation is constructed as a linear combination of all blending functions: $S(\xi,\eta) = \sum_{\alpha\in A_T} P_\alpha\ R_\alpha(\xi,\eta)$. The rational T-spline functions are capable to reproduce a constant function, but, in general, cannot reproduce a polynomial of a higher order. That is, T-spline blending functions do not span a complete polynomial space.

**Remark 3.** *Generalization of bivariate T-splines to 3D case is straightforward. Trivariate T-spline function are defined using three local knot vectors inferred by marching through the 3D T-mesh until intersect perpendicularly a mesh face, see Fig. 3.3. Then, the function is constructed as a product of three univariate B-spline functions:* $B_\alpha(\xi,\eta,\zeta) = B[\Xi_\alpha](\xi)B[\mathcal{H}_\alpha](\eta)B[Z_\alpha](\zeta)$.

## 3.3 T-splines as a basis for Isogeometric Analysis and design. Drawbacks and limitations

To be used in analysis, the functions should possess some necessary properties as linear independence, polynomial reproduction property and nesting behaviour of the approximation spaces in order to guarantee a proper convergence. In general, T-splines are not linearly independent. In [53] the authors give an example of a T-mesh with linearly dependent T-spline functions. However, this example involves a very unusual mesh with repeated interior knots, which is not common in practice. The great majority of T-meshes leads to independent T-spline functions. It is shown in [54] that T-splines without repeated interior knots are linearly independent. Another important property is the capacity to reproduce exactly polynomial functions. Unfortunately, in general, T-splines do not possess

this property. This can result in a poor approximation ability and convergence behaviour. Let us see an example illustrating this fact. We are going to use T-splines to interpolate over a square domain $\Omega = [0,1]^2$ a smooth function given by $u(\xi,\eta) = \exp -20((\xi-0.5)^2+(\eta-0.5)^2)$, see Fig. 3.4(a). We define a set of rational bicubic T-spline functions $\{R_\alpha\}_{\alpha\in A_T}$ for a given mesh $T$ and interpolate the given function. The spline approximation is built as a linear combination of the blending functions

$$u_h(\boldsymbol{\xi}) = \sum_{\alpha\in A_T} c_\alpha\, R_\alpha\left(\boldsymbol{\xi}\right).$$

The control values $c_\alpha$ are found by imposing the interpolation conditions

$$u(\boldsymbol{\xi}_\beta) = \sum_{\alpha\in A_T} c_\alpha\, R_\alpha\left(\boldsymbol{\xi}_\beta\right), \qquad \forall \boldsymbol{\xi}_\beta,\ \beta\in A_T,$$

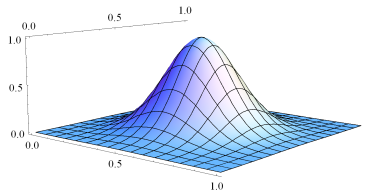
As interpolation points we use the anchors of the functions, i.e., the vertices of the mesh. Adaptive refinement is performed according to the indicator based on the exact $L^2$ interpolation error:

$$\eta(\Omega_e)^2 = \|u-u_h\|^2_{L^2(\Omega_e)} = \int_{\Omega_e} (u-u_h)^2\ \mathrm{d}\Omega.$$

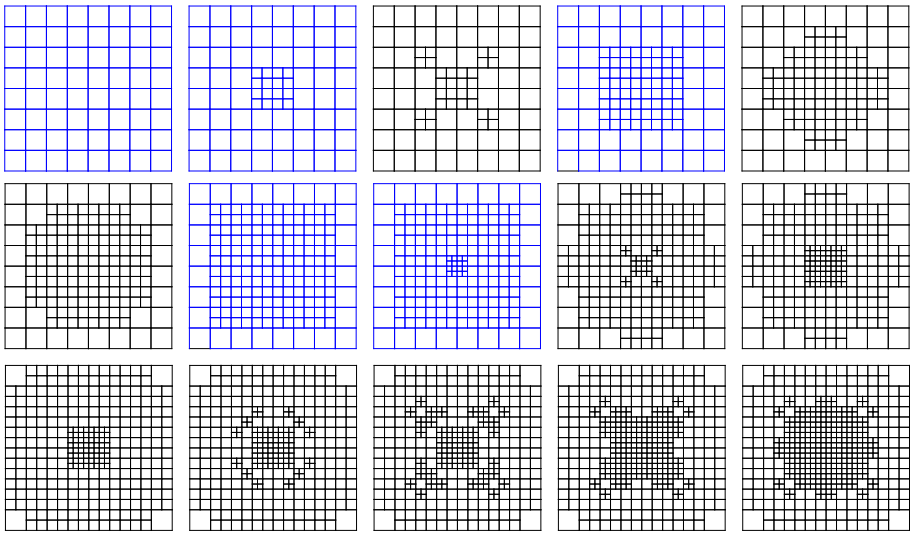
A cell $\Omega_e$ is marked to be refined if $\eta(\Omega_e) > \gamma\ \max_i\{\eta(\Omega_i)\}$, being $\gamma\in[0,1]$.

Some steps of the adaptive refinement are shown in Fig. 3.4(b), and Fig. 3.4(c) illustrates the error convergence in $L^2$-norm. As can be seen, no monotone convergence is obtained, since the error presents a significant jump for some mesh configurations. T-spline space corresponding to some meshes have poor approximation ability. To clarify the issue, the following study was carried out. Rational functions $\{R_\alpha\}$ are obtained from a set of polynomial spline blending functions $\{B_\alpha\}_{\alpha\in A_T}$ by normalization. For each of these meshes we can check whether its blending functions $\{B_\alpha\}_{\alpha\in A_T}$ span a complete polynomial space over $\Omega$. For that we interpolate some polynomial functions of degree 3 and check if the interpolation gains to represent exactly the polynomial (in the sense that the exact interpolation error $\|u-u_h\|_{L^2}$ is zero). It was observed that some error increases on the Fig. 3.4(c) correspond to the meshes $T$ with "incomplete" space: $\mathbb{P}_3(\Omega) \not\subset S_T = \mathrm{span}\{B_\alpha\}_{\alpha\in A_T}$. These meshes and their corresponding error are plotted in black colour in Fig. 3.4(b) and (c), respectively. The meshes that do span a complete polynomial space over $\Omega$ are marked in blue; and a proper convergence behaviour takes place for the sequence of these meshes. In Fig. 3.4(c) we also depict the error convergence for the same interpolation performed with polynomial spline functions proposed in this thesis, which, as can be seen, leads to monotone convergence.

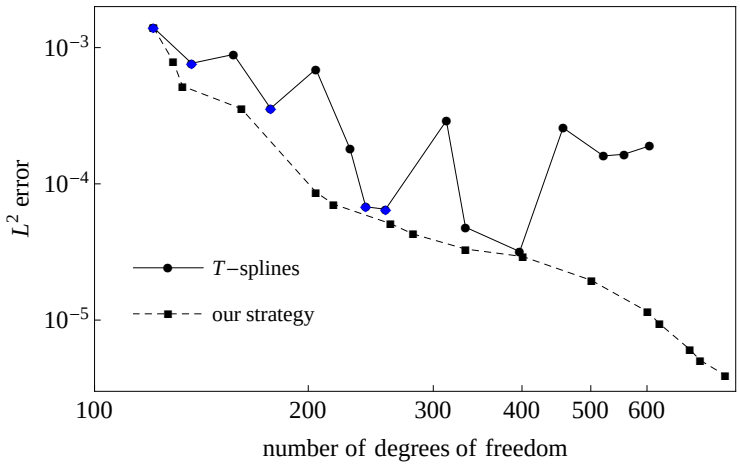
As another illustration of a poor approximation ability of T-splines, let us see an example of 2D parameterization. Suppose we are given a T-mesh $T$, a set of interpolation points $\{\boldsymbol{\xi}_\alpha\}_{\alpha\in A_T}\in\mathbb{R}^2$ in parametric space and their images in the




(a) The function to approximate.



(b) A sequence of T-meshes during the adaptive refinement. Polynomial T-meshes (with polynomial reproduction property) are represented in blue.




(c) Error convergence, blue points corresponds to the meshes marked in blue.

**Figure 3.4:** *Example of using rational T-splines for interpolation.*

physical space $\{\boldsymbol{x}_\alpha\}_{\alpha\in A_T} \in \mathbb{R}^2$. We build a T-spline representation of the geometry as lineal combination of T-spline blending functions

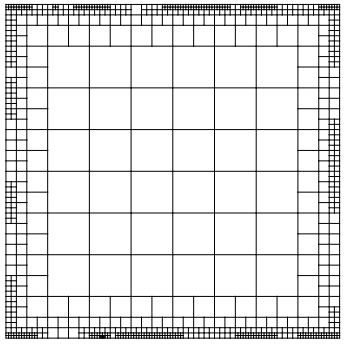
$$\mathbf{S}(\boldsymbol{\xi}) = \sum_{\alpha\in A_T} \mathbf{P}_\alpha \, R_\alpha(\boldsymbol{\xi}), \tag{3.1}$$

where $\mathbf{P}_\alpha \in \mathbb{R}^2$ is the control point corresponding to the $\alpha$-th blending function. The control points $\mathbf{P}_\alpha$ are found by imposing the interpolation conditions

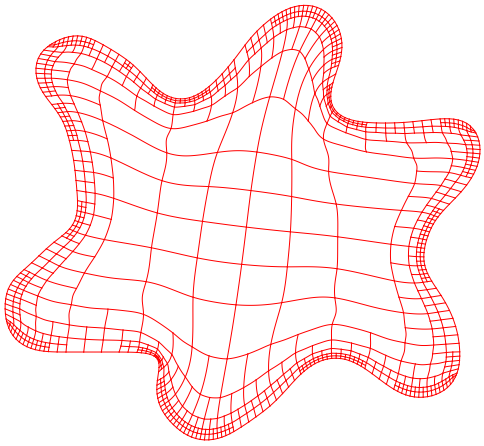
$$\mathbf{x}_\beta = \mathbf{S}(\boldsymbol{\xi}_\beta) = \sum_{\alpha\in A_T} \mathbf{P}_\alpha \, R_\alpha(\boldsymbol{\xi}_\beta), \quad \forall \boldsymbol{\xi}_\beta,\ \beta \in A_T \tag{3.2}$$

The resulting T-spline parameterization is shown in Fig. 3.5(b). On the other hand, for a comparison we perform the interpolation of the same data points using polynomial spline blending functions, which is the main result of the thesis. Our blending functions $\{N_\alpha\}_{\alpha\in A_T}$ defined over the parametric T-mesh span a complete polynomial space over $T$, i.e., $\mathbb{P}_3(\Omega) \subset \text{span}\{N_\alpha\}_{\alpha\in A_T}$. The resulting spline parameterization is shown in Fig. 3.5(c). It can be appreciated that our polynomial parameterization leads to a more uniform parametric mapping, meanwhile rational T-spline parameterization presents some oscillation, see Fig. 3.5(d).

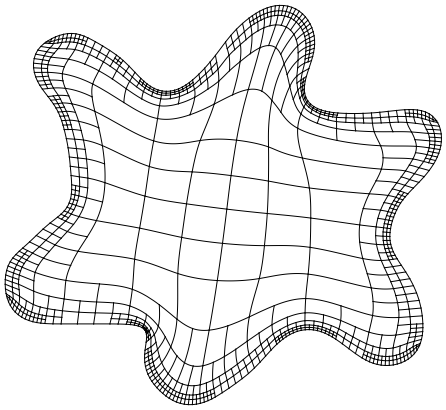
**Remark 4.** *Also, it worth mentioning that recently the issue of using rational spline functions, was discussed in [55] (It is Time to Drop the "R" from NURBS). The authors discuss the elevated complexity and computational cost of NURBS and that using of rational forms for parameterization also entails some disadvantages, regarding the quality. The use of NURBS in recent decades was motivated by their capacity of representing exactly conic sections. However, the authors suggest that there is no need for the rational form, since the conics can be approximated with sufficient accuracy in a robust and efficient manner with polynomial B-splines.*



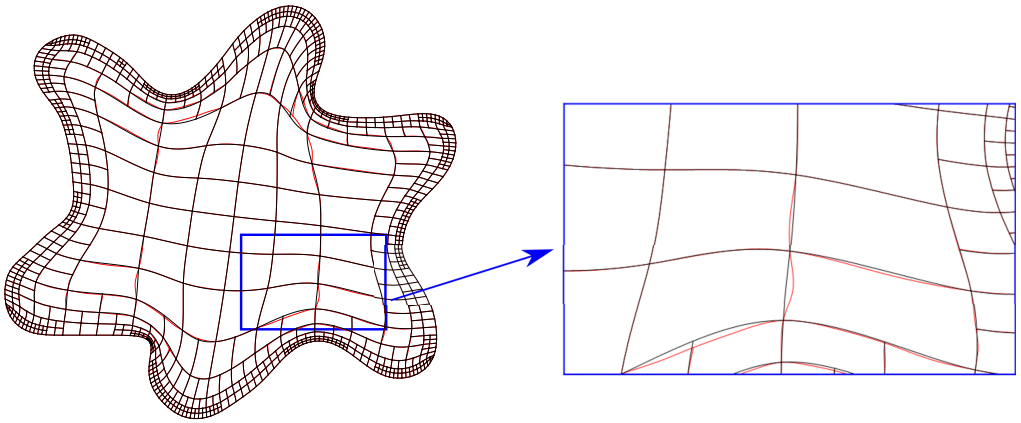
(a) Parametric T-mesh.



(b) T-spline parameterization.



(c) Polynomial spline parameterization.



(d) Some oscillations of rational T-spline parametrization.

**Figure 3.5:** *Example of using rational T-splines for domain parameterization.*

## 3.4 Analysis-suitable T-splines

T-splines was an important invent that overcame the drawback of tensor product NURBS spaces and provided a flexible tool for geometric modelling. However, for their use in analysis, T-splines lack some essential properties. As a solution to this problem, *analysis-suitable T-splines* were proposed in [23], which is a class of T-splines defined over T-meshes with certain topological restrictions formulated in terms of *T-junction extensions*. T-junction extensions include face and edge extensions, which are line segments originating at a T-junction, see Fig. 3.6. For cubic T-splines a face extension is created by marching from the T-junction, in the direction of a missing edge, until two perpendicular edges are intersected. An edge extension is then formed by marching in the opposite direction until intersect one edge. A T-mesh is called analysis-suitable if no T-junction extension intersects perpendicularly another T-junction extension.

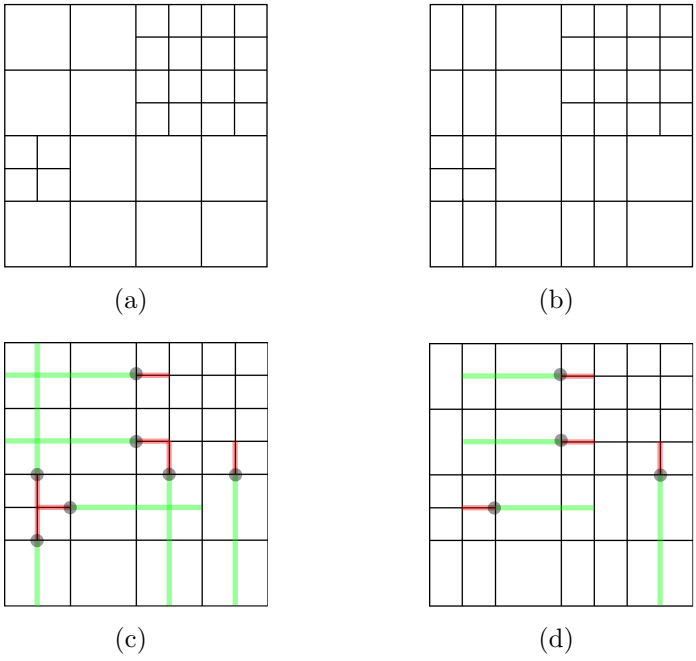



**Figure 3.6:** *Analysis-suitable T-spline. (a) Initial, not analysis-suitable T-mesh. (b) Possible extensions to obtain analysis-suitable T-mesh. (c) T-junction extensions of the initial mesh, edge extension in red, face extension in green. (d) T-junction extensions of the final mesh do not intersect.*

T-spline functions defined over an analysis-suitable T-mesh are linearly independent [24] and possess polynomial reproduction property. The refinement algorithm allows to accomplish highly localized refinements and construct nested polynomial spline spaces with optimal approximation properties. However, to guarantee all these properties, it is necessary to work with T-meshes that fulfil the

required topological restrictions. A given T-mesh should be extended by adding new edges until obtaining an analysis-suitable T-mesh, which, in general, is not trivial task that involves a global search of optimal T-mesh extension. Besides, the solution is not unique. The strategy presents an elevated implementation complexity and, as far as we know, the extension of the strategy to 3D cases is still an open question.

## 3.5 Hierarchical refinement scheme

Another well known approach for local refinement is the hierarchical refinement scheme. This approach is based on a simple and natural idea to construct multilevel spaces by replacing coarse level basis functions by the finer functions. Starting from an initial uniform mesh, hierarchical refinement scheme leads to sequential construction of nested spline spaces with linearly independent basis functions. This approach is based on the refinability property of B-spline functions. Let $\Xi^0$ be the initial uniform knot vector. We can perform $k$ uniform subdivisions to obtain a $k$-level knot vector $\Xi^k$. Then, a basis function $B^k_{i,p}$ defined on $\Xi^k$ can be expressed as a linear combination of $p+2$ basis functions defined on $\Xi^{k+1}$, i.e.,

$$B^k_{i,p} = \sum_{j=0}^{p+1} c_j \, B^{k+1}_{j,p}, \quad \text{where} \quad c_j = 2^{-p} \binom{p+1}{j}.$$

The $p+2$ basis functions $B^{k+1}_{j,p}$ of level $k+1$ are called *children* of the $k$-level function $B^k_{i,p}$. For example, Fig. 3.7 shows a cubic B-spline function $B^k_{0,3}$ defined on the knot vector $\Xi^k = \{1, 2, 3, 4, 5\}$. An uniform subdivision of $\Xi^k$ leads to the knot vector $\Xi^{k+1} = \{1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5\}$ and 5 children functions $B^{k+1}_{j,3}$ $(j = 0, \ldots, 4)$ defined on it. Then, the function $B^k_{0,3}$ is expressed as linear combination of its children $B^k_{0,3} = \frac{1}{8}B^{k+1}_{0,3} + \frac{1}{2}B^{k+1}_{1,3} + \frac{3}{4}B^{k+1}_{2,3} + \frac{1}{2}B^{k+1}_{3,3} + \frac{1}{8}B^{k+1}_{4,3}$.

Figure 3.8 illustrates the idea of hierarchical refinement scheme. Figure 3.8(a) shows the initial (level 0) B-spline space, where the interval $[4, 9]$ is to be refined. The basis functions (blue discontinuous line), whose support is contained in the refined interval $[4, 9]$, are replaced by the finer basis functions of level 1 shown in Fig. 3.8(b). Next, the interval $[5, 8]$ is refined and all basis functions of level 1, whose support is contained in the refined area, are replaced by level 2 functions shown in Fig. 3.8(c). The final spline space, shown in Fig. 3.8(d), is composed by the basis functions from three figures represented in black continuous line. Note that all the functions are a scaled and translated version of each other. The idea of hierarchical refinement scheme for 2D is illustrated in Fig. 3.9.

Relatively simple implementation and straightforward generalization to 3D make it an attractive option for local refinement. However, there are some drawbacks. Hierarchical refinement scheme permits to enrich a given spline space by replacing some functions by new functions of finer level, but it is not clear how to

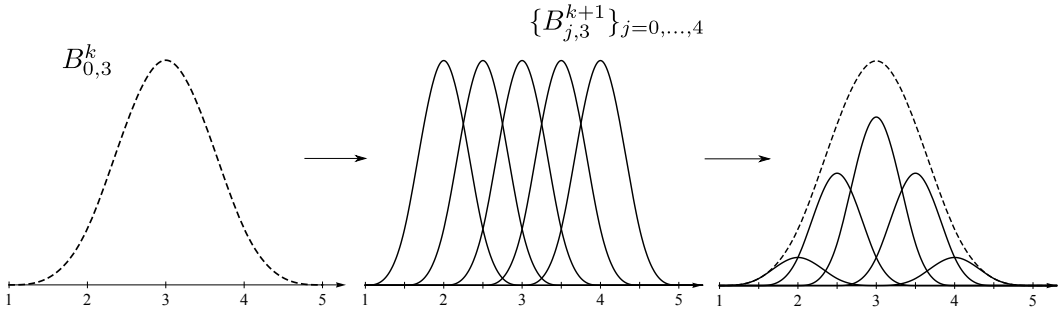
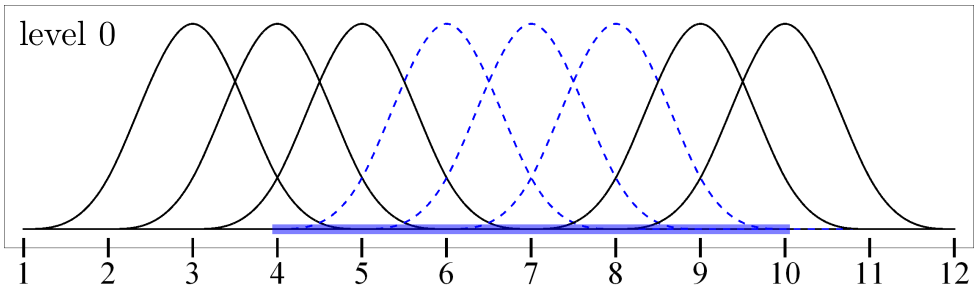



**Figure 3.7:** *Refinability of B-spline functions. Rightmost figure shows the function $B_{0,3}^{k}$ and its children $B_{j,3}^{k+1}$ scaled by their corresponding coefficients.*

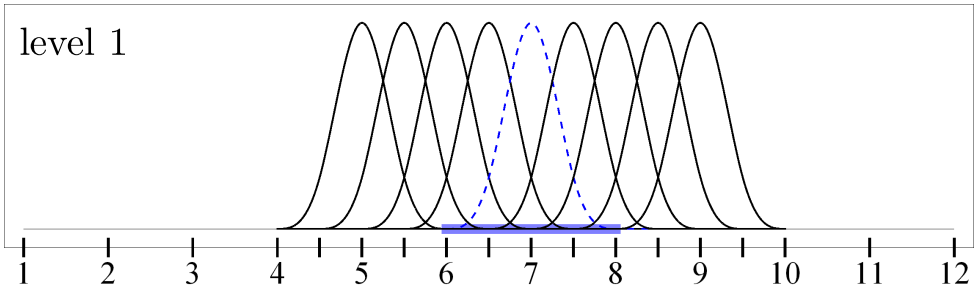
define a spline space for a given T-mesh if no previous mesh (or space) is given. In other words, unlike T-splines, this strategy does not define a spline space from a given arbitrary mesh, but perform a sequential enrichment of spline spaces by adding new functions and eliminating the old ones to avoid linear dependency. In addition, hierarchical refinement scheme imposes a certain obligatory requirements on the refined area that extend the refined zone more than necessary. For example, for 2D mesh, in the worst case, refinement of only one cell leads to the refinement of totally 25 cells and introduction of 49 new basis functions. In 3D case the same situation leads to 125 new cells. And finally, another shortcoming of the strategy is an excessive support overlapping, which can affect the sparsity and condition number of the stiffness matrix. To solve this issue, a truncation technique was proposed in [30]. Truncated hierarchical splines are derived from the hierarchical basis by redefining the functions, namely, eliminating the contributions of the functions of finer levels to coarser level ones, and thus improving their locality. However, this complicates the strategy.

## 3.6 Motivation of our work

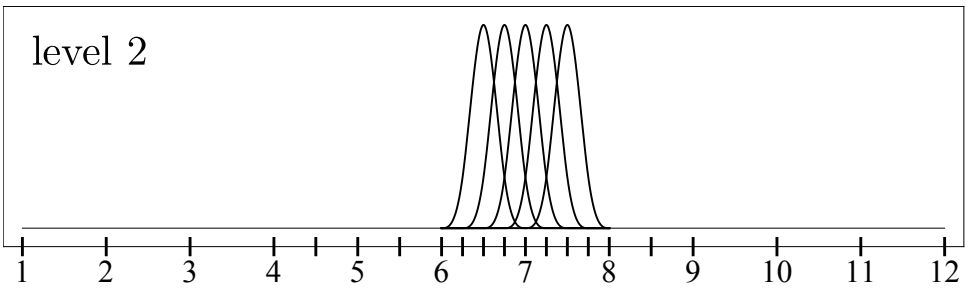
We need to construct spline spaces with *nice* properties for their use in analysis and design. The main goal we pursue here is to find a simple practical solution to the problem of local refinement with straightforward implementation, both in 2D and 3D. Inspired in T-splines and hierarchical refinement scheme, the aim of our work was to elaborate another possible strategy to define spline spaces that combine only nice features of each strategy. The rest of the paper exposes the developed technique.



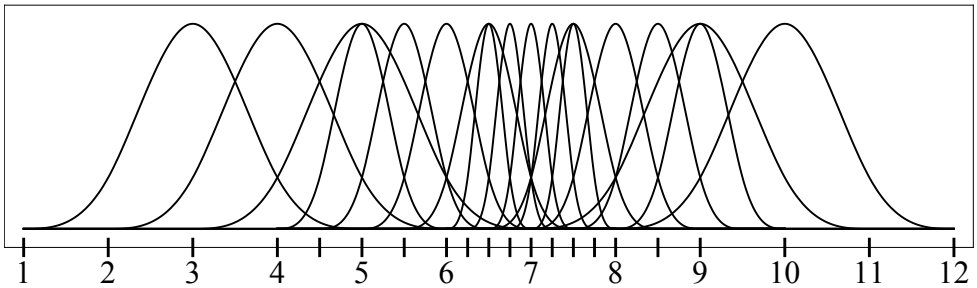

(a)




(b)




(c)




(d) Final space composed by the functions of three levels plotted in black.

**Figure 3.8:** *Hierarchical refinement scheme in 1D.*

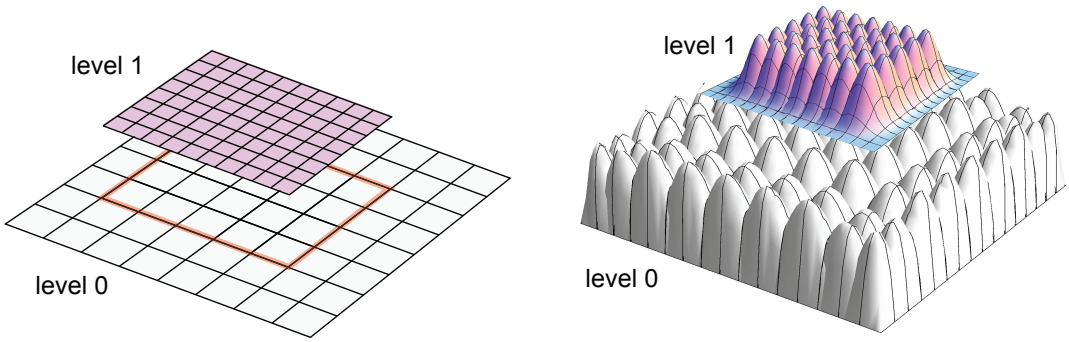



**Figure 3.9:** *Hierarchical refinement scheme in 2D.*

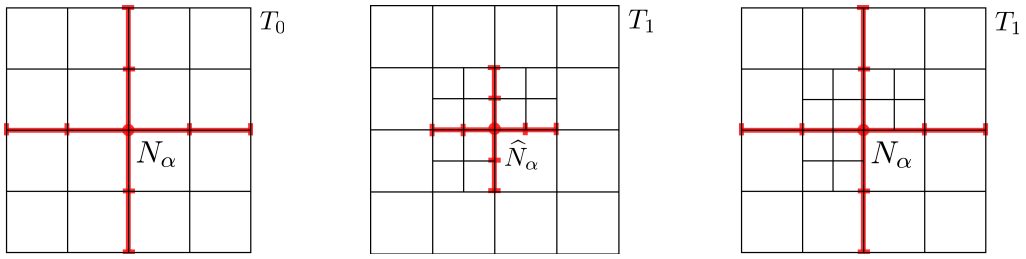
# CHAPTER 4

## Strategy for construction of polynomial spline spaces over hierarchical T-meshes

In this chapter we expose the main result of the thesis, that is, a new strategy to define tensor product spline functions over T-meshes. Due to the elevated complexity of all current strategies, the main goal we pursue here is the simplicity and low computational cost, both in 2D and 3D. For this, we have to assume a restriction on the T-mesh. Namely, the T-mesh should fulfil the requirement of being a *0-balanced* (strongly balanced) quadtree/octree mesh. Assuming this reasonable restriction over the T-mesh, we can define easily cubic spline functions that span spaces with desirable properties: linear independence, $C^2$-continuous, cubic polynomial reproduction property, nestedness of spanned spaces and a straightforward implementation. The key of the strategy lies in some simple rules used for inferring local knot vectors for each blending function.

## 4.1 Main steps of the strategy.

The strategy we propose here has some similarity with T-splines inasmuch as we define the blending functions from local knot vectors that are inferred by traversing T-mesh edges. Some additional rules and requirements are imposed for the local knot vectors in order to obtain spline spaces with desired properties. These additional rules were elaborated by studying and analysing different situations, when T-spline functions do not span a complete polynomial space. At the same time some features of the resulting rules and requirements are inspired by the hierarchical refinement scheme. The strategy is motivated by the idea of preserving nesting behaviour of the spaces under the mesh refinement. We have to prevent the situation, when a blending function cannot be reproduced with the basis of the new spline space after the refinement. This would guarantee the nestedness of the spline spaces. For example, in the initial mesh of the Fig. 4.1(a), we define a T-spline basis function $N_\alpha$ associated to the vertex $\alpha$. Then, we perform some cell refinements as shown in Fig. 4.1(b). It is easy to check that the new




(a) (b) T-splines. (c) Our method.

**Figure 4.1:** *Motivation of the strategy. (a) Initial mesh and a basis function $N_\alpha$. (b) Refined mesh, where the new T-spline space cannot reproduce the original function $N_\alpha$. (c) We try to preserve the original basis function $N_\alpha$ in the new space if it cannot be recovered.*

T-spline space, obtained in Fig. 4.1(b), is not capable to reproduce the original basis function $N_\alpha$. However, if we define the function $\widehat{N}_\alpha$ as shown in Fig. 4.1(c), keeping, in this case, the initial support unaltered in the new refined space, we guarantee the nestedness of the spline spaces. This is one of the situations, where we need to interfere to guarantee nesting behaviour of the spaces. The resulting rules, that we have elaborated for inferring function supports, attempt to conserve nesting behaviour for all possible configurations that can take place for a strongly balanced T-mesh.

The process of construction of spline space for a given T-mesh can be divided in the following three steps:

1. *Mesh pretreatment (0-balancing)*
2. *Inferring local knot vectors by traversing T-mesh edges*
3. *Modification of local knot vectors*

Next, we give a description of each step of the process.

## 4.2 Mesh pretreatment. 0-balanced quadtree and octree T-meshes

The technique we present here is designed for hierarchical T-meshes (multilevel meshes) with a quad- and octree subdivision scheme. This type of meshes can be efficiently implemented with tree data structures [51], which are frequently used in engineering.

Quadtree data structure is a common hierarchical spatial data structure used for recursive decomposition of the space. A given bi-dimensional region, normally

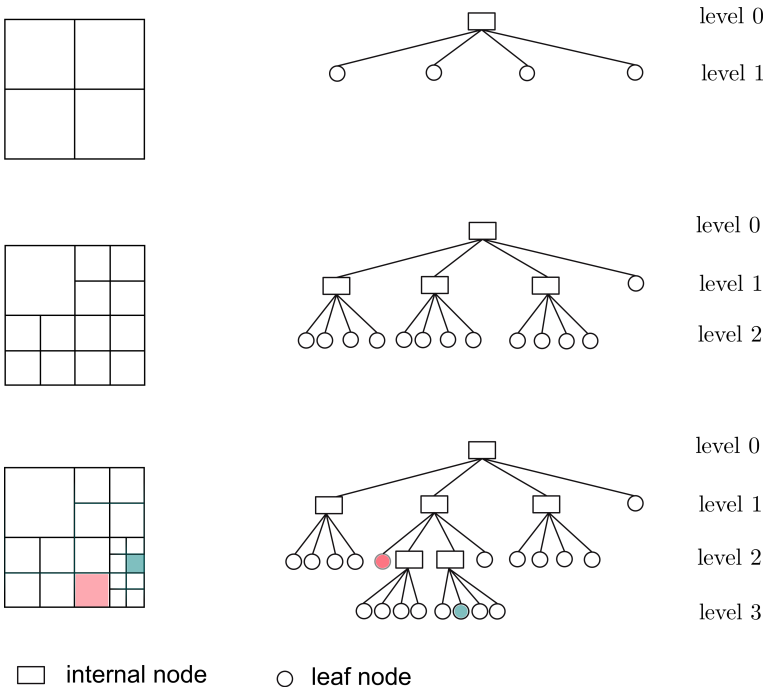



**Figure 4.2:** *Hierarchical T-mesh implementation using quadtree data structure.*

a square, is divided into four quadrants that are called its *children*. Each of these new quadrants can be divided in the same manner and so on. The elements of the quadtree are called *tree nodes*. The node that has children is called *internal node* and the node that does not have children is called *leaf node*. Each leaf node of the quadtree represents a cell of the T-mesh, see Fig. 4.2. If an internal node belongs to level $k$, then its children belong to level $k+1$. The root node has level $k=0$. If a $k$-level leaf node is refined then it turns into a $k$-level internal node. A quadtree mesh $T$ is said to be of level $k$ if its smallest cell is of level $k$. The concept of quadtree T-mesh is illustrated in Fig. 4.2.

Due to their simplicity, quadtree and octree T-meshes are an attractive tool for performing adaptive refinement in IGA and geometric modelling. To guarantee a good quality of the approximation space constructed over a mesh, it is preferable to have a gradual transition from the coarse mesh to the finely refined zone. That is why it is common to work with balanced quadtree and octree meshes. The strategy we propose in this thesis is designed exclusively for the 0-balanced T-meshes. A mesh with tree-like structure is said to be *0-balanced* if for any $k$, no cell at level $k$ shares a vertex (0-face) with a cell at level greater than $k+1$. In other words, a 0-balanced quadtree mesh implies that any cell can contact (through vertex, edge or face) only the cells that differ at most in one level of depth. An example of 0-balanced quadtree is shown in Fig. 4.3. To obtain a 0-balanced quadtree, a standard balancing procedure is applied. Note that refinements performed during the 0-balancing procedure do not propagate, see [56]. Also, in this manuscript we

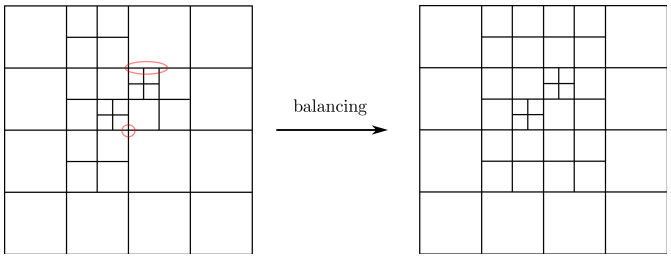



**Figure 4.3:** *Example of 0-balancing procedure. Leftmost figure shows a 0-unbalanced quadtree mesh. Rightmost image corresponds to resulting 0-balanced mesh after balancing procedure.*

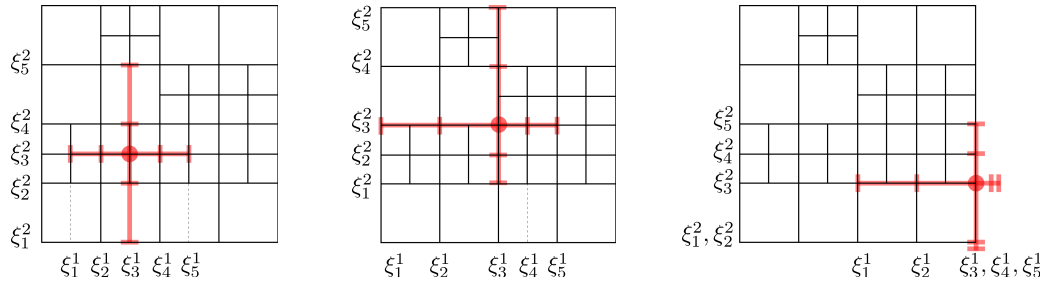
are going to use the term *strongly balanced* quadtree/octree, which is equivalent to the term *0-balanced* quadtree/octree.

It should be highlighted that 0-balanced T-mesh is an essential prerequisite for the construction of spline spaces by means of our technique. In general, if the T-mesh is not 0-balanced, our rules for inferring local knot vectors do not lead to a space with desirable properties. Also, it is important to emphasize that, for our 2D (3D) parametric T-meshes, a subdivision of any cell is performed by subdividing the cell in 4 (8) equal sub-cells, so all cells of the same level have the same size, and the edge size of a $k$-level cell is twice larger than the edge size of a $(k+1)$-level cell.

## 4.3 Inferring local knot vectors

Let us consider a T-mesh $T$ of the square parametric domain $\Omega = [0,1]^d$, $d = 2$ or 3. Valence 4 node of the mesh is called *regular node*, and *T-junction* is a node (vertex) of valence 3. Also, we are going to refer to T-junctions as *hanging nodes*. We assign functions only to the regular nodes of the mesh. The *skeleton* of a $d$-dimensional mesh $T$ is the geometric set of points composed of the union of all $(d-1)$-faces of the mesh and it is denoted by $\mathrm{skt}(T)$. That is, for a 2D mesh, the *skeleton* is the union of all the edges of the mesh, and the *skeleton* of a 3D mesh is the union of all its faces.

To define our cubic tensor product spline blending functions over a given $d$-dimensional T-mesh, a local knot vector for $d$ parametric directions should be assigned to each function $N_\alpha$: $\Xi_\alpha^j = \left(\xi_1^j, \xi_2^j, \xi_3^j, \xi_4^j, \xi_5^j\right)$, $j = 1, ..., d$. Similarly to T-splines [21], these knot vectors are inferred by traversing the T-mesh *skeleton*. For simplicity, let us describe this procedure for a two-dimensional T-mesh. Starting from the central knot $(\xi_3^1, \xi_3^2)$, which is called the *anchor* of the function, we walk across the T-mesh until intersect perpendicularly a mesh edge. According to our strategy, we should skip over the encountered T-junctions, where the missing edge is perpendicular to the direction of our marching, see Fig. 4.4. When the boundary




(a) No T-junction is skipped.

(b) Two T-junctions are skipped in each direction.

(c) Two T-junctions are skipped and the knots are repeated when the boundary is reached.

**Figure 4.4:** *Inferring local knot vectors for a bivariate function by traversing T-mesh edges.*

of the parametric domain is reached, while walking across the mesh, we repeat the knots creating an open knot vector structure along the boundary, see Fig. 4.4(c). Note that all interior knots have multiplicity 1. Then, for a pair of local knot vectors $\Xi_\alpha^1$ and $\Xi_\alpha^2$, the bicubic spline function $N_\alpha$ is defined as $N_\alpha(\xi^1, \xi^2) = B[\Xi_\alpha^1](\xi^1)B[\Xi_\alpha^2](\xi^2)$, where $B[\Xi_\alpha^j](\xi^j)$ is univariate B-spline corresponding to the knot vector $\Xi_\alpha^j$. Thus, we obtain for the mesh $T$ a set of blending functions $\{N_\alpha\}_{\alpha \in A_T}$, where $A_T$ is the index set corresponding to the mesh T. Figure 4.5 illustrates an example of T-mesh in the parameter space and the anchors of all blending functions defined over this mesh. Any interior regular node has exactly one function associated to it, and the boundary nodes have more than one function associated due to the open knot vector structure.

The process of inferring local knot vectors can be resumed as follows:

- *Blending functions are associated only to regular nodes of the mesh.*
- *Local knot vectors are inferred by walking across the mesh until intersecting perpendicularly the mesh skeleton. This intersection should not coincide with a T-junction perpendicular to the direction of our marching.*
- *Boundary knots are repeated to create an open knot vector structure along the boundary.*

Next, in order to span a spline space with good properties, some of the inferred knot vectors should be modified. This issue is addressed in the next subsection.

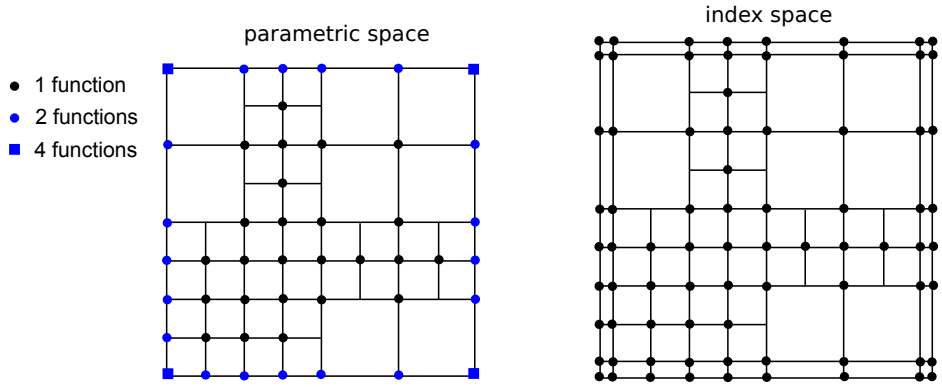



**Figure 4.5:** *An example of T-mesh and its anchors, parametric mesh and index mesh. Leftmost figure: black circles represent the interior nodes that have one blending function associated, blue circles are the boundary nodes that have 2 blending functions and the blue squares are the boundary nodes with 4 blending functions due to the open knot vector structure along the boundary.*

## 4.4 Modification of local knot vectors

The key of our strategy lies in some simple rules used for modification of function supports that lead to the construction of a spline space with desirable properties. In order to describe the idea, let us introduce some notation. For the local knot vector $\Xi_\alpha^j = \left(\xi_1^j, \xi_2^j, \xi_3^j, \xi_4^j, \xi_5^j\right)$, $j = 1, ..., d$ let us denote the length of each knot interval as $\Delta_i^j = \xi_{i+1}^j - \xi_i^j$, $j = 1, ..., d$ and $i = 1, ..., 4$.

The support of a $d$-variate blending function $N_\alpha$ is a $d$-dimensional rectangular *box*: $[\xi_1^1, \xi_5^1] \times \cdots \times [\xi_1^d, \xi_5^d]$. We are going to call *frame* of a function support the union of all $(d-2)$-faces of this *box* and we denote it by $\mathrm{frm}(\mathrm{supp}\, N_\alpha)$. That is, for the rectangular support of a bivariate function, the *frame* is the union of the four vertices of this rectangle. For the cuboidal support of a trivariate function, its *frame* is composed of the union of the twelve edges of this cuboid.

Once the function supports are inferred, we modify them in such a way that for each blending function $N_\alpha$ its knot vectors $\Xi_\alpha^j$ verify the following simple conditions:

**Condition 1:** *Local knot vectors of the $d$-variate function $N_\alpha$ verify*[1]

$$\Delta_1^j \geqslant \Delta_2^j = \Delta_3^j \leqslant \Delta_4^j, \quad j = 1, ..., d. \tag{4.1}$$

**Condition 2:** *The frame of the function support should be situated over the mesh skeleton*:

$$\mathrm{frm}(\mathrm{supp}\, N_\alpha) \in \mathrm{skt}(T). \tag{4.2}$$

[1]Except the cases involving repeated knots that are explained at the end of Section 4.5.1.

Thus, the function supports that do not meet Conditions 1 and 2 should be modified. To perform this modification we extend the original support by changing some knot intervals until the resulting support satisfies both conditions. We are going to refer to these support modifications as *Extension rule* 1 and *Extension rule* 2, respectively.

In the next section we give a detailed description of this procedure for 2D and 3D cases.

## 4.5 Support modification rules

Here, we present simple support *Extension rules* 1 and 2 to obtain local knot vectors that fulfil Conditions 1 and 2 formulated in the previous section. We proceed as follows. First, if after traversing the T-mesh skeleton the local knot vectors of a function do not satisfy Condition 1, we modify some of their knots in order to meet Condition 1. Then, the fulfilment of Condition 2 is checked and, if it is not satisfied, another appropriate modification of the support is carried out. As a result of these modifications we obtain a new extended support with local knot vectors verifying both conditions. These modifications are easily implemented taking into account the balanced tree structure of the mesh. Let see in detail this procedure.

To simplify the notation, in the rest of the chapter we denote the parametric coordinates as $(\xi, \eta, \zeta)$, which is related to the previous notation as $(\xi^1, \xi^2, \xi^3) = (\xi, \eta, \zeta)$. Consequently, $(\Xi^1, \Xi^2, \Xi^3) = (\Xi, \mathcal{H}, \mathcal{Z})$ and $(\Delta_i^1, \Delta_i^2, \Delta_i^3) = (\Delta_i^\xi, \Delta_i^\eta, \Delta_i^\zeta)$.

### 4.5.1 Support extension rules for 2D meshes

In order to facilitate the description and illustration of the strategy, some concepts and notation introduced in section 4.4 have to be particularized to the 2D case. The skeleton $\mathrm{skt}(T)$ of a two-dimensional mesh $T$ is the union of all edges of the mesh. For a bivariate function let us denote the vertices of its rectangular support as $V_{1,1} = (\xi_1, \eta_1)$, $V_{5,1} = (\xi_5, \eta_1)$, $V_{5,5} = (\xi_5, \eta_5)$ and $V_{1,5} = (\xi_1, \eta_5)$. Then, the *frame* of a function support is the union of its four vertices, i.e. $\mathrm{frm}(\mathrm{supp}\, N_\alpha) = \{V_{n,m},\ n, m \in \{1, 5\}\}$. Figure 4.6 illustrates the introduced notation for a bivariate function support.

Formulation of Condition 1 for the local knot vectors $\Xi$ and $\mathcal{H}$ is simple. It means that *two central knot intervals should have the same size:* $\Delta_2^\xi = \Delta_3^\xi$*, and the other two exterior knot intervals* $\Delta_1^\xi$ *and* $\Delta_4^\xi$ *should not be smaller than the central intervals:* $\Delta_1^\xi \geqslant \Delta_2^\xi = \Delta_3^\xi \leqslant \Delta_4^\xi$. Analogously for $\mathcal{H}$. Condition 2, adapted to 2D case, is formulated as follows: *The four vertices of a function support should be situated over the mesh edges.*

Extension rule 1. If the local knot vector $\Xi$ of a function does not satisfy Condition 1, we modify this vector by skipping the minimal number of the knots until $\Delta_1^\xi \geqslant \Delta_2^\xi = \Delta_3^\xi \leqslant \Delta_4^\xi$ is verified, and analogously for $\mathcal{H}$. This modification is made independently for each parametric direction. Let see an example of support

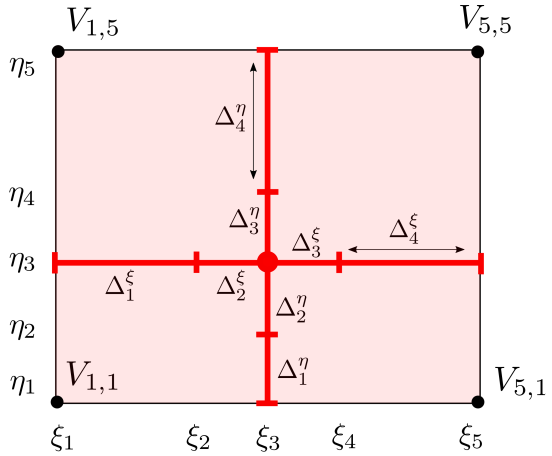
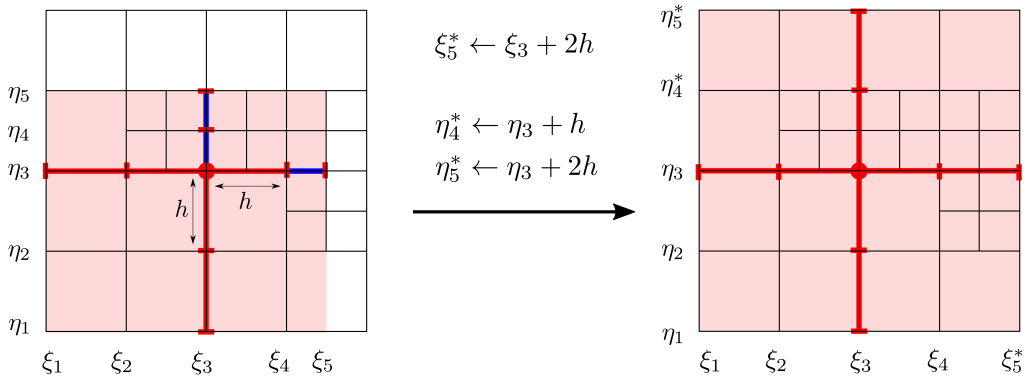



**Figure 4.6:** *Support notation for 2D case.*

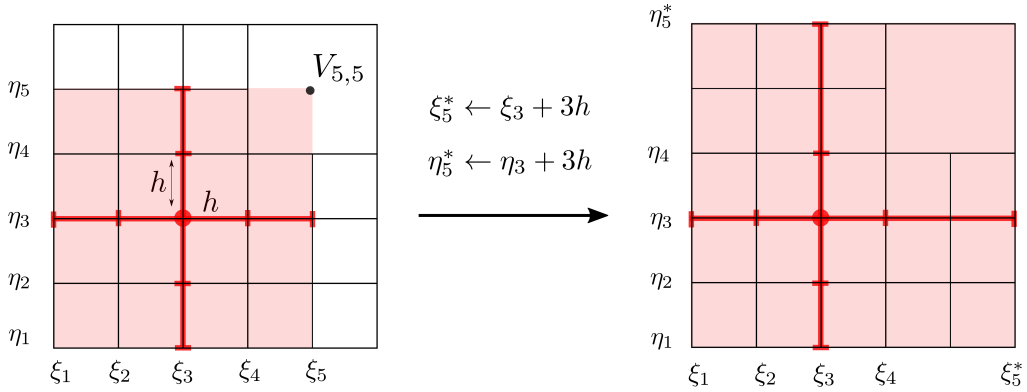
modification for a bivariate function. Leftmost function support shown in Fig. 4.7(a) does not meet Condition 1. For the knot vector $\Xi$ we have $\Delta_3^\xi > \Delta_4^\xi$, so the knot interval $\Delta_4^\xi$ should be modified. Let us denote $h = \max(\Delta_2^\xi, \Delta_3^\xi) = \max(\Delta_2^\eta, \Delta_3^\eta)$. Note that both maxima coincide due to the quadtree structure and the fact that the T-junctions are skipped. Then, to modify $\Delta_4^\xi$, we just double its size by redefining the fifth knot $\xi_5$ as $\xi_5^* \leftarrow \xi_3 + 2h$. For the local knot vector $\mathcal{H}$ we have $\Delta_2^\eta > \Delta_3^\eta$, so the knot intervals $\Delta_3^\eta$ and $\Delta_4^\eta$ should be modified. Again, we just double their size. Namely, the knots $\eta_4$ and $\eta_5$ are redefined as $\eta_4^* \leftarrow \eta_3 + h$, $\eta_5^* \leftarrow \eta_3 + 2h$. Note that basically we skip over some knots of the mesh in order to double the size of some knot intervals.

Extension rule 2. Once Condition 1 is satisfied, in order to fulfil Condition 2, we check whether the vertices of the function support are situated over the mesh edges. If not, we modify the knot vectors by skipping over a knot for both parametric directions and placing this vertex over the mesh edges. Note that, again it implies to double the size of some knot intervals. In this case only exterior knot intervals are extended. An example of a function support violating Condition 2 is illustrated in Fig. 4.7(b). The corner vertex $V_{5,5} = (\xi_5, \eta_5)$ of this support is not situated over a mesh edge, so the intervals $\Delta_4^\xi$ and $\Delta_4^\eta$ are to be extended. For this the fifth knots for both parametric directions are redefined as $\xi_5^* \leftarrow \xi_3 + 3h$, $\eta_5^* \leftarrow \eta_3 + 3h$, and thus, the new vertex $V_{5,5}$ is placed over the mesh edges. The checking of Condition 2 and the extension is performed independently for each of the four quadrants of the function support. Note that for our 0-balanced quadtree we should make this checking only for some functions. For example, without loss of generality, the support vertex $V_{5,5} = (\xi_5, \eta_5)$ must be checked only if $\Delta_3^\xi = \Delta_4^\xi = \Delta_3^\eta = \Delta_4^\eta$.

**Remark 5.** *Note that, when a function support violates Condition 2, the extension of this support, in order to place its vertex over the mesh skeleton, can be made*




(a) Condition 1 is not satisfied because $\Delta_4^\xi < \Delta_3^\xi$ and $\Delta_3^\eta < \Delta_2^\eta$. Knot intervals to be changed are marked in blue.




(b) Condition 2 is not satisfied because $V_{5,5} \notin skt(T)$.

**Figure 4.7:** *Extension rules. (a) An example of support modification by Extension rule 1. (b) An example of support modification by Extension rule 2.*

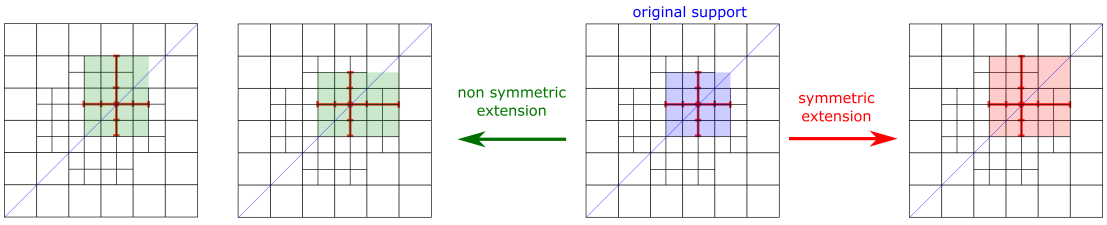



**Figure 4.8:** *Extension rule 2. Other possibilities to place the corner vertex over the mesh skeleton.*

*by modifying only one of the knot vectors instead of the both, see Fig. 4.8. This option leads to the loss of symmetry for the spline space. That is, a symmetric T-mesh would have a non symmetric function supports. For simplicity we extend the knot vectors in both parametric directions and conserve the symmetry.*

**Algorithm 1:** Extension rule 1.

**Input**: A knot vector $\Xi = (\xi_1, \xi_2, \xi_3, \xi_4, \xi_5)$, $\xi_i \in [0, 1]$.

**Function** ***Modify1***$(\Xi)$
- $\Xi^* \leftarrow \Xi$
- $h = \max(\Delta_2, \Delta_3)$
- **if** $\Delta_2 < \Delta_3$ *and* $\xi_2 > 0$ **then**
  - $\xi_2^* \leftarrow \xi_3 - h$
  - $\xi_1^* \leftarrow \xi_2^*$
  - **if** $\xi_1^* > 0$ **then** $\xi_1^* \leftarrow \xi_3 - 2h$
- **if** $\Delta_1 < \Delta_2$ *and* $\xi_1 > 0$ **then**
  - $\xi_1^* \leftarrow \xi_3 - 2h$
- **if** $\Delta_2 > \Delta_3$ *and* $\xi_4 < 1$ **then**
  - $\xi_4^* \leftarrow \xi_3 + h$
  - $\xi_5^* \leftarrow \xi_4^*$
  - **if** $\xi_5^* < 1$ **then** $\xi_5^* \leftarrow \xi_3 + 2h$
- **if** $\Delta_4 < \Delta_3$ *and* $\xi_5 < 1$ **then**
  - $\xi_5^* \leftarrow \xi_3 + 2h$
- **return** $\Xi^*$

**Output**: A corrected knot vector $\Xi^*$ that satisfies Condition 1.

**Algorithm 2:** Extension rule 2 in 2D.

**Input**: A *0-balanced* mesh $T$ and a pair of local knot vectors $S = \{\Xi, \mathcal{H}\}$.

**Function** ***Modify2***$(T,\ S)$
- $S^* \leftarrow S$
- $h = \max(\Delta_2^\xi, \Delta_3^\xi)$
- **for** $n \in \{1, 5\}$ **do**
  - **for** $m \in \{1, 5\}$ **do**
    - **if** $(\xi_n, \eta_m) \notin skt(T)$ **then**
      - $\xi_n^* \leftarrow \xi_3 + 3h\,\mathrm{sgn}(\xi_n - \xi_3)$
      - $\eta_m^* \leftarrow \eta_3 + 3h\,\mathrm{sgn}(\eta_m - \eta_3)$
- **return** $S^*$

**Output**: A modified support $S^* = \{\Xi^*, \mathcal{H}^*\}$ that satisfies Condition 2.

The extension of any other function support is completely analogous to these two examples. In all possible cases, the extension of a function support implies to change some knot intervals by doubling its size.

Detailed algorithms for Extension rule 1 and 2, used to modify a bivariate

function support according to Conditions 1and 2, are given in Algorithms 1 and 2.

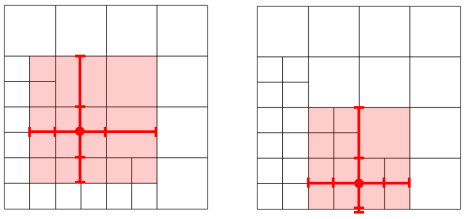
Figure 4.9 shows some examples of support extension procedure. Functions that satisfy both conditions and do not need be modified are given in Fig. 4.9(a). Examples of support extension according to Condition 1 are shown in Fig. 4.9(b), (c), (d) and (e). And Fig. 4.9(f), (g) and (h) illustrate support extension according to Condition 2 or both.

Note that an exception for Condition 1 is a knot vector that contains a knot interval of length 0 due to the open knot vector structure along the boundary. In this case, a knot vector should fulfil the inequality (4.1) not taking into account the knot intervals of length 0. That is, Extension rule 1 is applied only to non-zero knot intervals, see Fig. 4.9(c), (d) and (e).

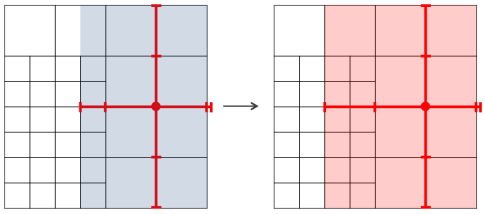
**Remark 6.** *It is important to highlight that application of the extension rules always place the redefined knots over the mesh edges, i.e., the extension rules just skip over some knots of the mesh, but do not invent new knots that are not induced by the T-mesh.*

**Remark 7.** *It is important to underline that inferring of each function support does not depend on the rest of the functions, so the process can be parallelized.*

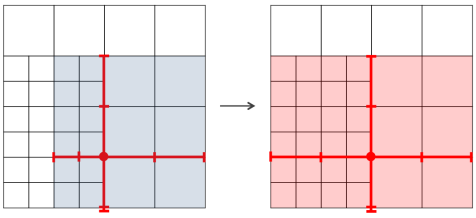
**Remark 8.** *As was said at the beginning of the chapter, it is important take into account that the rules and algorithms we formulate here are designed for uniform quadtree subdivisions, where all cells of the same level have the same size. However, if for some reason, it is necessary to deal with non-uniform subdivision, the process of inferring of functions supports for such mesh can be translated to another, isomorphically equivalent, mesh (let us call it reference mesh), where all subdivisions are strictly uniform.*



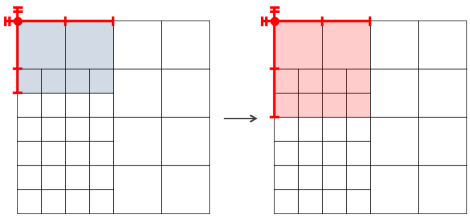
(a) Initial supports that satisfy both conditions and should not be modified.



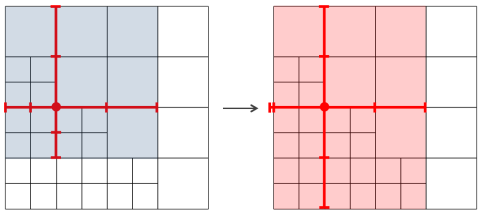
(b) Condition 1 is not satisfied because $\Delta_1^\xi < \Delta_2^\xi$, so $\xi_1^* \leftarrow \xi_3 - 2h$.



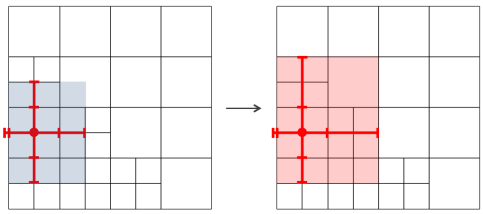
(c) Condition 1 is not satisfied because $\Delta_2^\xi < \Delta_3^\xi$, so $\xi_1^* \leftarrow \xi_3 - 2h$ and $\xi_2^* \leftarrow \xi_3 - h$.



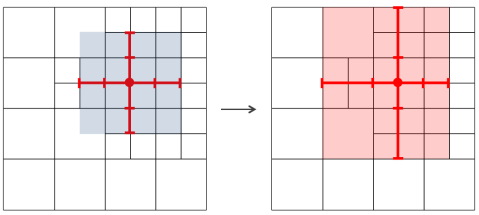
(d) Condition 1 is not satisfied because $\Delta_1^\eta < \Delta_2^\eta$, so $\eta_1^* \leftarrow \eta_3 - 2h$.



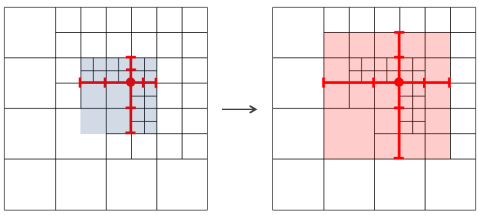
(e) Condition 1 is not satisfied because $\Delta_2^\xi < \Delta_3^\xi$ and $\Delta_2^\eta < \Delta_3^\eta$, so $\xi_2^* \leftarrow \xi_3 - h$, $\eta_2^* \leftarrow \eta_3 - h$ and $\eta_1^* \leftarrow \eta_3 - 2h$.



(f) Condition 2 is not satisfied because $V_{5,5} \notin skt(T)$, so $\xi_5^* \leftarrow \xi_3 + 3h$ and $\eta_5^* \leftarrow \eta_3 + 3h$.



(g) Condition 2 is not satisfied because $V_{1,1}$ and $V_{1,5} \notin skt(T)$, so $\xi_1^* \leftarrow \xi_3 - 3h$, $\eta_1^* \leftarrow \eta_3 - 3h$ and $\eta_5^* \leftarrow \eta_3 + 3h$.



(h) Conditions 1 and 2 are not satisfied because $\Delta_2^\xi > \Delta_3^\xi$, $\Delta_2^\eta > \Delta_3^\eta$ and $V_{1,1} \notin skt(T)$, so $\xi_4^* \leftarrow \xi_3 + h$, $\xi_5^* \leftarrow \xi_3 + 2h$, $\eta_4^* \leftarrow \eta_3 + h$, $\eta_5^* \leftarrow \eta_3 + 2h$, $\xi_1^* \leftarrow \xi_3 - 3h$, $\eta_1^* \leftarrow \eta_3 - 3h$.

**Figure 4.9:** *Examples of function support modification with Extension rule 1 and 2. Initial support is marked in blue and extended support in red.*

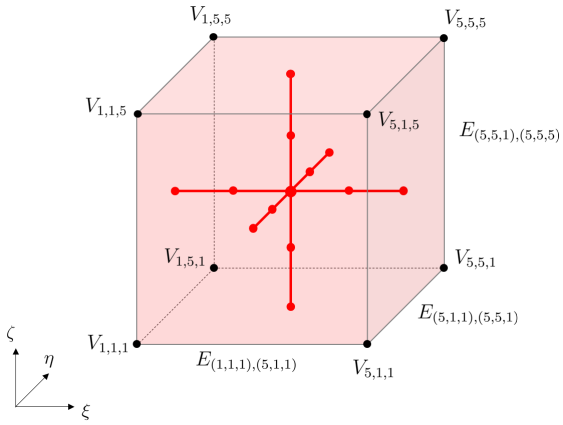



**Figure 4.10:** *Support notation for 3D case.*

### 4.5.2 Support extension for 3D meshes

In this section we give a description and illustration of the proposed strategy for defining trivariate spline functions over 0-balanced octree T-meshes.

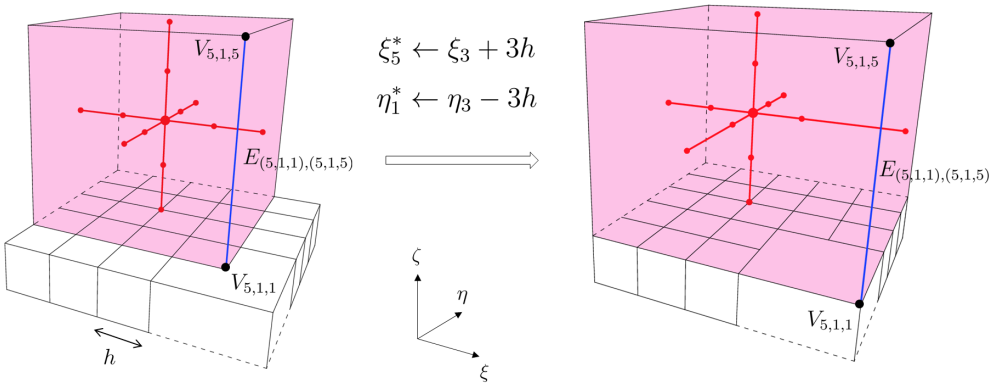
The skeleton $\mathrm{skt}(T)$ of a three-dimensional mesh $T$ is the union of all faces of the mesh. For a trivariate function let us denote the vertices of its support as $V_{n,m,k} = (\xi_n, \eta_m, \zeta_k)$, where $n, m, k \in \{1, 5\}$. And the edge formed by two vertices $V_{n,m,k}$ and $V_{p,q,r}$ is denoted by $E_{(n,m,k),(p,q,r)}$. Then, the *frame* $\mathrm{frm}(\mathrm{supp}\, N_\alpha)$ of a trivariate function support is the union of its twelve edges. Figure 4.10 illustrates the introduced notation for the support of a trivariate blending function.

The formulation of Condition 1 for the local knot vectors of a trivariate function is the same as for the 2D case. Condition 2 adapted to 3D meshes is stated as follows: *Edges of the cuboidal function support should be situated over the mesh faces.*

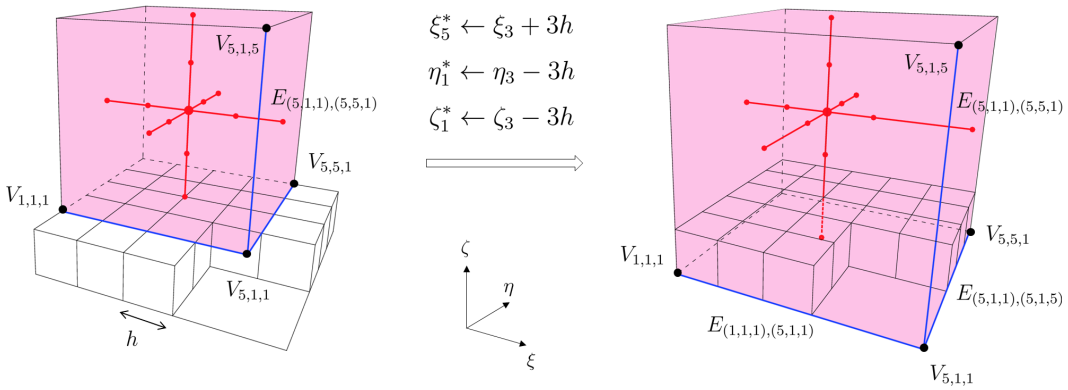
The implementation of the strategy for 3D is similar to the 2D case. To satisfy Condition 1, *Extension rule* 1 is applied to each of the three local knot vectors of a function, analogously to the 2D case, using Algorithm 1.

Extension rule 2. In order to fulfil Condition 2, we check whether the edges of a function support are situated over the mesh faces. If not, the two knot vectors perpendicular to this edge should be modified by skipping over a knot for both parametric directions and placing this edge over the mesh faces. Analogously to 2D, we double the size of some exterior knot intervals. The checking is performed independently for each of the eight quadrants of the function support and, in each quadrant, three edges should be checked. Figure 4.11 illustrates the support extension procedure for the quadrant of the vertex $V_{5,1,1}$. Due to the octree structure only two cases can take place: (i) the quadrant contains one edge that is not situated over the mesh faces or (ii) the quadrant contains three edges and a vertex that are not situated over the mesh faces. Let see each case.

(i) If a quadrant contains one edge that does not fulfil Condition 2, then two




(a) Only one edge (in blue) violating Condition 2. Node $V_{5,1,1}$ is sited in the center of the face of size $2h$. The support is extended in two directions.




(b) Three edges (in blue) violating Condition 2. Node $V_{5,1,1}$ is sited in the center of the cell of size $2h$. The support is extended in three directions.

**Figure 4.11:** *Extension rule 2 for support modification of a trivariate function.*

knot vectors perpendicular to this edge are modified, see Fig. 4.11(a). For the function support shown in Fig. 4.11(a) left, the edge $E_{(5,1,1),(5,1,5)}$ is not situated over the mesh faces. Therefore, two knot vectors $\Xi$ and $\mathcal{H}$, perpendicular to this edge, are modified in order to place the edge over the mesh faces, namely, the knots $\xi_5$ and $\eta_1$ are redefined as $\xi_5^* \leftarrow \xi_3 + 3h$ and $\eta_1^* \leftarrow \eta_3 - 3h$, where $h = \max(\Delta_2^\xi, \Delta_3^\xi) = \max(\Delta_2^\eta, \Delta_3^\eta) = \max(\Delta_2^\zeta, \Delta_3^\zeta)$.

(ii) If a quadrant contains three edges that are not situated over the mesh faces, then the three knot vectors are modified by skipping over a knot for each of the three parametric directions, see Fig. 4.11(b). Vertex $V_{5,1,1}$ and the three edges connected to it are not situated over the mesh skeleton, so all the three knot vectors are extended to place the three edges over the mesh faces: $\xi_5^* \leftarrow \xi_3 + 3h$, $\eta_1^* \leftarrow \eta_3 - 3h$, $\zeta_1^* \leftarrow \zeta_3 - 3h$.

**Algorithm 3:** Extension rule 2 in 3D.

**Input**: A *0-balanced* T-mesh $T$ and three local knot vectors $S = \{\Xi, \mathcal{H}, \mathcal{Z}\}$.

**Function** ***Modify2*** $(T,\ S)$
- $S^* \leftarrow S$
- $h = \max(\Delta_2^\xi, \Delta_3^\xi)$
- $mov(i) := i + 4\,\mathrm{sgn}(3 - i)$
- **for** $n \in \{1, 5\}$ **do**
  - **for** $m \in \{1, 5\}$ **do**
    - **for** $k \in \{1, 5\}$ **do**
      - **if** $E_{(n,m,k),(mov(n),m,k)} \notin skt(T)$ **then**
        - $\eta_m^* \leftarrow \eta_3 + 3h\,\mathrm{sgn}(\eta_m - \eta_3)$
        - $\zeta_k^* \leftarrow \zeta_3 + 3h\,\mathrm{sgn}(\zeta_k - \zeta_3)$
      - **if** $E_{(n,m,k),(n,mov(m),k)} \notin skt(T)$ **then**
        - $\xi_n^* \leftarrow \xi_3 + 3h\,\mathrm{sgn}(\xi_n - \xi_3)$
        - $\zeta_k^* \leftarrow \zeta_3 + 3h\,\mathrm{sgn}(\zeta_k - \zeta_3)$
      - **if** $E_{(n,m,k),(n,m,mov(k))} \notin skt(T)$ **then**
        - $\xi_n^* \leftarrow \xi_3 + 3h\,\mathrm{sgn}(\xi_n - \xi_3)$
        - $\eta_m^* \leftarrow \eta_3 + 3h\,\mathrm{sgn}(\eta_m - \eta_3)$
- **return** $S^*$

**Output**: A modified support $S^* = \{\Xi^*, \mathcal{H}^*, \mathcal{Z}^*\}$ that satisfies Condition 2.

Algorithm 3 explains the Extension rule 2 used to modify a trivariate function support according to Condition 2.

## 4.6 Classification of the functions supports (2D case)

It is worth noting that the proposed rules and the 0-balanced restriction for the mesh lead to a very few number of all possible function supports that can be defined over a T-mesh.

Let us consider any knot vector $\Xi = (\xi_1, \xi_2, \xi_3, \xi_4, \xi_5)$ and its knot intervals $\Delta_i^\xi = \xi_{i+1} - \xi_i$, $i = 1, ..., 4$. The set $\boldsymbol{\Delta}^\xi = (\Delta_1^\xi, \Delta_2^\xi, \Delta_3^\xi, \Delta_4^\xi)$ determines a knot vector $\Xi$. Due to the Extension rule 1 and the balanced mesh condition, we have the following possible configurations for the set $\boldsymbol{\Delta}^\xi$: $(h, h, h, h)$, $(h, h, h, 2h)$, $(2h, h, h, h)$ and $(2h, h, h, 2h)$, see Fig. 4.12(a). All possible bivariate function supports are obtained by combining all configurations for $\Xi$ and $\mathcal{H}$, i.e., combining two sets $\boldsymbol{\Delta}^\xi$ and $\boldsymbol{\Delta}^\eta$. Note that, since we ignore the hanging nodes while inferring two knot vectors, the value of $h$ is the same for $\boldsymbol{\Delta}^\xi$ and $\boldsymbol{\Delta}^\eta$, i.e., $h = \Delta_2^\xi = \Delta_3^\xi = \Delta_2^\eta = \Delta_3^\eta$, see Fig. 4.12(b), and the value of $h$ determines the level of a function. We say that

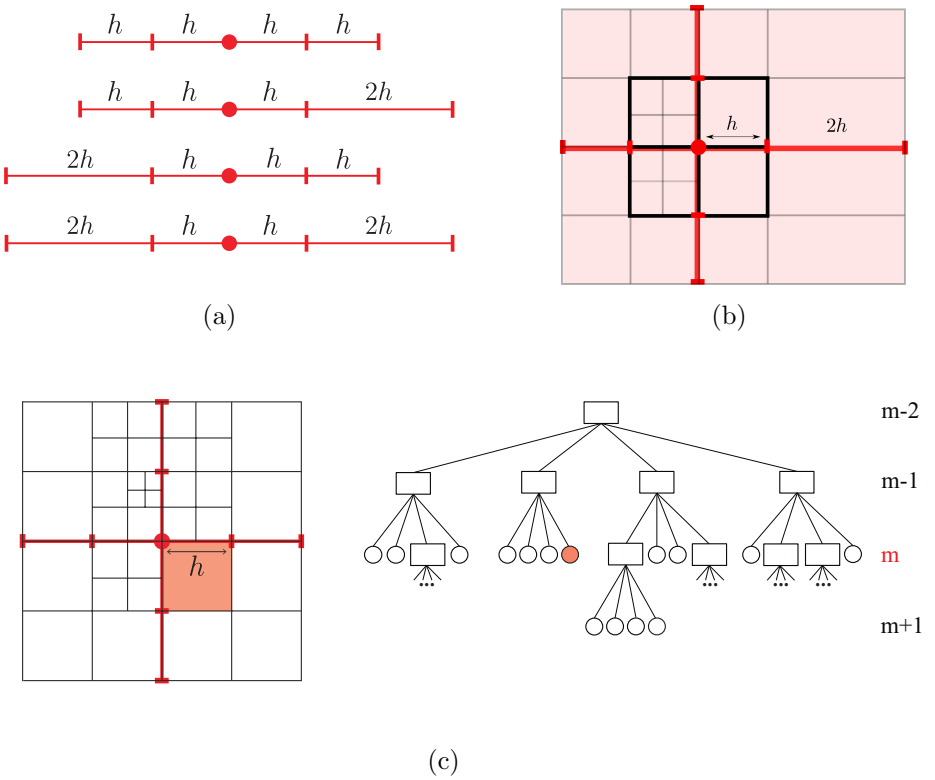
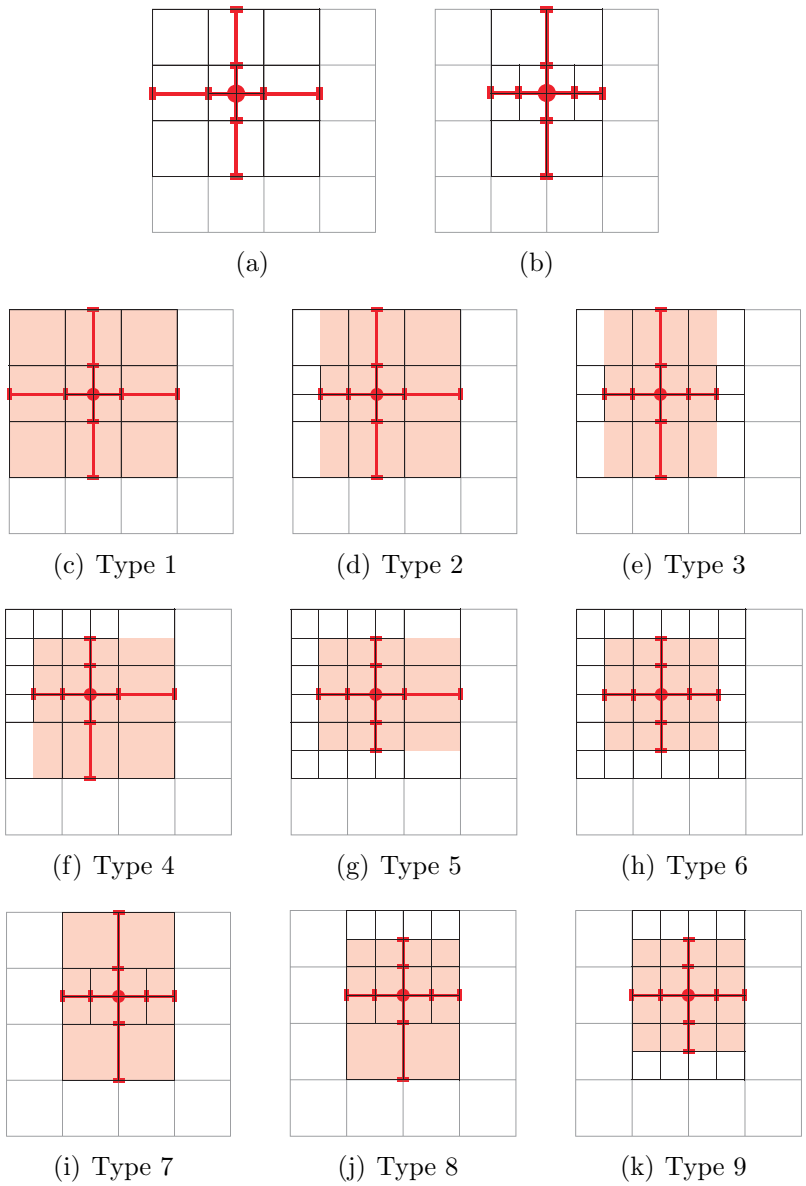



**Figure 4.12:** *(a) All possible configurations for a knot vector (without repeated knots). (b) For any function support* $h = \Delta_2^\xi = \Delta_3^\xi = \Delta_2^\eta = \Delta_3^\eta$. *(c) A function support of level* $m$*, which is determined by the cell marked in red.*

a blending function is of level $m$ if the largest cell contacting its anchor is of level $m$, see Fig. 4.12(b)(c). Thus, the functions we define according our strategy have a hierarchy.

We can classify all function supports that can take place as follows. First, we can distinguish different function supports according to the underlying local mesh that leads to this support. Basically, it depends on the position of the anchor of the function, see Fig. 4.13. For scenario (a) the anchor of the $m$-level function coincides with the center of a *(m-1)*-level cell of the quadtree; scenario (b) the rest of the functions, i.e., the anchor of the $m$-level function does not correspond to any center of a *(m-1)*-level cell of the quadtree. Second, taking into account all the considerations about values of $\Delta^\xi$ and $\Delta^\eta$, we can classify all function supports in nine types that are illustrated in Fig. 4.13. First six types correspond to scenario (a) and the other three types correspond to scenario (b).

Note that the supports in Fig. 4.13 are illustrated over the simplest (minimal) possible mesh configuration that gives rise to this support. However, it is important to notice that the same function support can take place over another (more refined) underlying mesh configuration, see for example Fig. 4.12(b)(c).



(a) (b)

(c) Type 1 (d) Type 2 (e) Type 3

(f) Type 4 (g) Type 5 (h) Type 6

(i) Type 7 (j) Type 8 (k) Type 9

**Figure 4.13:** *Two possible scenarios for a function support according to the underlying local mesh. (a) The anchor of the m-level function coincides with the center of a (m-1)-level cell of the quadtree. (b) The rest of the functions. (c)-(k) Function support classification according to the underlying local mesh and configurations for $\Delta^{\xi}$ and $\Delta^{\eta}$.*

**Remark 9.** *For convenience, we are going to put a name to the spline blending functions defined over T-mesh according to the proposed strategy. From now on, we are going to refer to the functions as* ***Extended polynomial splines (EP-splines)****.*

# CHAPTER 5

## Properties of the constructed spline spaces

Here, we summarize the properties and some characteristics of the spline spaces constructed by means of our method.

## 5.1 Properties

We conjecture that for any strongly balanced T-mesh $T$ of the domain $\Omega$ the set of EP-spline functions spans a space $S_T(\Omega) = \text{span}\{N_\alpha : \alpha \in A_T\}$ with the following properties:

1. *Functions* $\{N_\alpha\}_{\alpha \in A_T}$ *are* $C^2$*-continuous.*
2. *Non-negativity:* $N_\alpha \geq 0$.
3. *Functions* $\{N_\alpha\}_{\alpha \in A_T}$ *are linearly independent (globally).*
4. *Non-negative weighted partition of unity:* $\sum_{\alpha \in A_T} c_\alpha N_\alpha = 1$, *where* $c_\alpha \geq 0$.
5. *Spaces spanned by nested T-meshes are also nested:*
   $T_1 \subset T_2 \;\Rightarrow\; S_{T_1} \subset S_{T_2}$.
6. *Order 3 polynomial reproduction property:* $\mathbb{P}_3(\Omega) \subset S_T(\Omega)$.

The first two properties are evident. The third, the fourth and the fifth have to be proved. The sixth property follows from the fifth one, since the initial B-spline space possess this property.

For now, we can provide a complete proof of the nesting behaviour of the spaces and non-negative partition of unity property for 2D case. Also some reasoning that hints the linear independence of the functions (2D case).

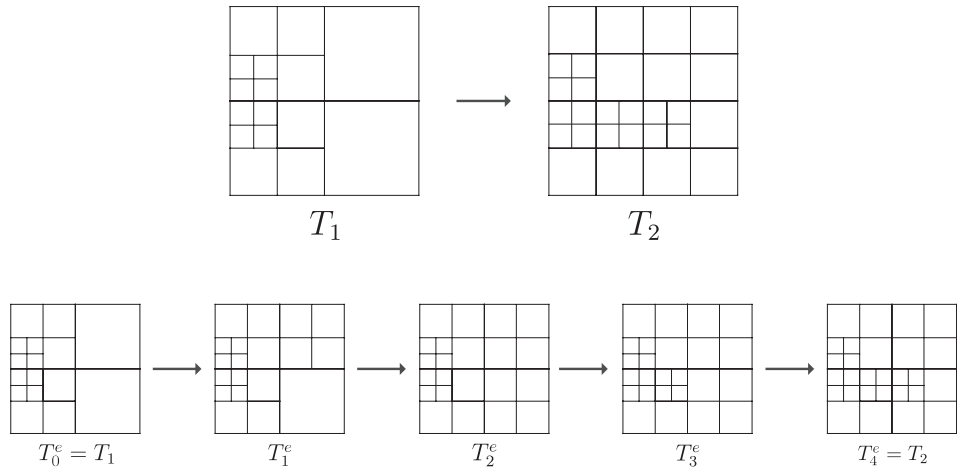



**Figure 5.1:** *Elemental refinements. A possible sequence of* 0*-balanced meshes* $\{T_i^e\}$*, where* $T_i^e$ *is obtained from* $T_{i-1}^e$ *by refining only one cell.*

## 5.2 Nesting behaviour of the spaces

Here we give an outline of the proof (2D case) of the nestedness of the EP-spline spaces constructed via our strategy. For complete proof we refer the reader to appendix of this chapter.

First, some basic concepts and notation are revised. A quadtree mesh $T$ is said to be of level $k$ if its smallest cell is of level $k$. We denote it as $k = \text{lev}(T)$, see Fig. 4.2. A blending function is said to be of level $m$ if the largest cell contacting its anchor is of level $m$, see section 4.6 and Fig. 4.12. We say that two meshes are nested $T_1 \subset T_2$, if $T_2$ is a refinement of $T_1$.

As we mentioned before, the proposed rules and the 0-balanced restriction for the mesh lead to a very few number of all possible function supports defined over a T-mesh, see section 4.6. In order to prove the nestedness of the spaces for any two nested T-meshes $T_1 \subset T_2$, we should verify that any blending function of $S_{T_1}$ can be represented as linear combination of blending functions from $S_{T_2}$. To this end, we have to perform this verification for all types of function supports and for any possible mesh configuration. However, applying some reasoning, it is possible to reduce this verification to a few number of basic cases.

First, to simplify the process, we assume that: *For any two 0-balanced nested meshes* $T_1 \subset T_2$ *there exists a sequence of meshes* $\{T_i^e\}_{i=0}^n$ *such that*

*(i)* $T_1 = T_0^e \subset T_1^e \subset T_2^e \subset \cdots \subset T_{n-1}^e \subset T_n^e = T_2$,

*(ii) any mesh* $T_i^e$ *is obtained from mesh* $T_{i-1}^e$ *by refining only one cell,*

*(iii) all the meshes* $\{T_i^e\}_{i=0}^n$ *are 0-balanced, see Fig. 5.1.*

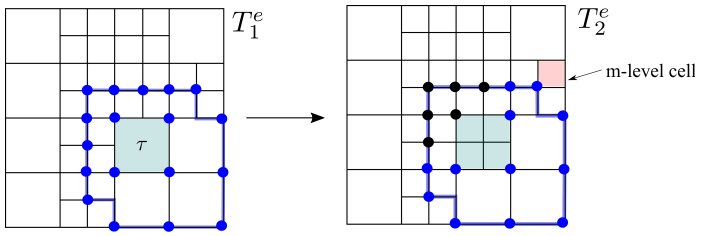



**Figure 5.2:** *An example of elemental refinement $T_1^e \subset T_2^e$ and the one-ring neighbourhood of the cell $\tau$. The contour of the one-ring neighbourhood and the anchors of its functions are marked in blue. Anchors whose functions are changed after the cell refinement are marked in black in the rightmost figure.*

Refinements that fulfil conditions (i)-(iii) we call *elemental refinements*, i.e., a pair of 0-balanced meshes $T_1^e \subset T_2^e$ is called *elemental refinement*, if $T_2^e$ is obtained from $T_1^e$ by refining only one cell.

It is evident that: *If any elemental refinement produces nested spaces, i.e., $T_1^e \subset T_2^e \;\Rightarrow\; S_{T_1^e} \subset S_{T_2^e}$, then any two 0-balanced nested meshes also produce nested spaces, i.e., $T_1 \subset T_2 \;\Rightarrow\; S_{T_1} \subset S_{T_2}$.* Indeed, for the sequence of elemental refinements we have: $S_{T_1} = S_{T_0^e} \subset S_{T_1^e} \subset S_{T_2^e} \subset \ldots \subset S_{T_n^e} = S_{T_2}$. Thus $S_{T_1} \subset S_{T_2}$.

So, taking into account the previous reasoning, our goal is to prove that any elemental refinement produces nested spaces. We have to verify that, after refining only one cell, blending functions of the new space are capable to represent all the functions of the previous space. To verify this, it is fundamental to take into account several considerations about our strategy:

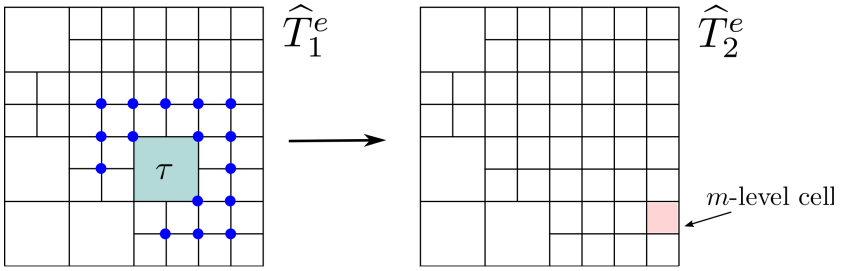
- As was discussed in section 4.6, there are only few types of possible function supports, which are determined by their knot intervals $\Delta^\xi = (\Delta^\xi_{\mathbf{1}}, \Delta^\xi_{\mathbf{2}}, \Delta^\xi_{\mathbf{3}}, \Delta^\xi_{\mathbf{4}})$ and $\Delta^\eta = (\Delta^\eta_{\mathbf{1}}, \Delta^\eta_{\mathbf{2}}, \Delta^\eta_{\mathbf{3}}, \Delta^\eta_{\mathbf{4}})$.

- Due to our rules, the refinement of one cell affects only some of the function supports in the neighbourhood of the cell. Namely, the anchors whose functions are changed after refinement of a cell, belong to the *one-ring neighbourhood* of this cell. We call *one-ring neighbourhood* of the cell $\tau$ the set of all cells that have contact with this cell, see Fig. 5.2. Thus, we have to verify that any function support of the mesh $T_1^e$ from the one-ring neighbourhood of the cell $\tau$ can be recuperated with the functions of the refined mesh $T_2^e$.

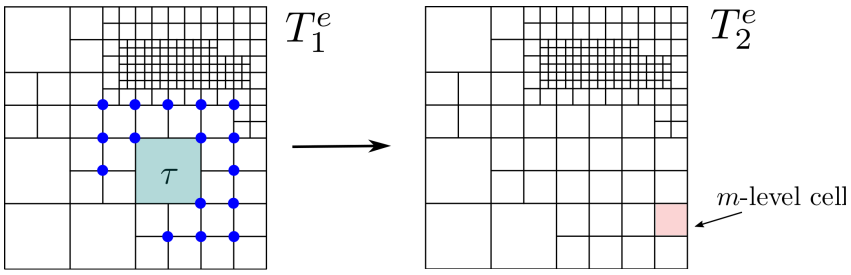
First, we consider a *simple case of elemental refinement* $\widehat{T}_1^e \subset \widehat{T}_2^e$, where a $m$-level mesh $\widehat{T}_2^e$ is obtained from the mesh $\widehat{T}_1^e$ by refining a *(m-1)*-level cell $\tau$ and $\widehat{T}_2^e$ does not include cells of level greater than $m$, see Fig. 5.3(a). We can show that

**Proposition 1.** *Any simple elemental refinement produces nested spaces:*

$$\widehat{T}_1^e \subset \widehat{T}_2^e \quad \Rightarrow \quad S_{\widehat{T}_1^e} \subset S_{\widehat{T}_2^e}.$$




(a) Simple elemental refinement: $\text{lev}(\widehat{T}_2^e) = m$.




(b) General elemental refinement: $\text{lev}(T_2^e) > m$.

**Figure 5.3:** *The idea of Proposition 2 is to reduce the proof for general case (b) to a simple case of elemental refinement (a).*

Next, applying some recursivity reasoning, we can show that the proof for a general case of elemental refinement ($\text{lev}(T_2^e) > m$) can be reduced to a simple elemental refinement. Thus, assuming that Proposition 1 holds, we can prove that

**Proposition 2.** *Any elemental refinement produces nested spaces:*

$$T_1^e \subset T_2^e \quad \Rightarrow \quad S_{T_1^e} \subset S_{T_2^e}.$$

The proof for Proposition 1 and 2 is given in appendix of this chapter. As was said before, the proof of Proposition 2 is based on Proposition 1. Thus, the core of the whole proof is Proposition 1.

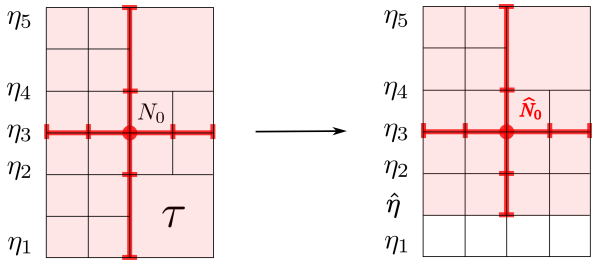
To verify, that for any simple elemental refinement $\widehat{T}_1^e \subset \widehat{T}_2^e$ all functions of the one-ring of the cell $\tau$ can be represented by means of the functions of the new space $S_{\widehat{T}_2^e}$, we proceed as follows. We analyse thoroughly all possible mesh configurations near the cell $\tau$ and show, using knot insertion formula, that any function $N_0$ is a linear combination of the new blending functions. We group all function supports in three types according to its position with respect to the cell $\tau$ and analyse each type. Figure 5.4 illustrates an example of such study. Suppose that the initial knot vector of the function $N_0$ is $\mathcal{H} = (\eta_1, \eta_2, \eta_3, \eta_4, \eta_5)$, and after the cell refinement the new function $\widehat{N}_0$ has its knot vector $\mathcal{H}_2 = (\hat{\eta}, \eta_2, \eta_3, \eta_4, \eta_5)$.

This situation can be seen as an insertion of the new knot $\hat{\eta}$ between the first and the second knot of the vector $\mathcal{H} = (\eta_1, \eta_2, \eta_3, \eta_4, \eta_5)$. Then, using knot insertion formula

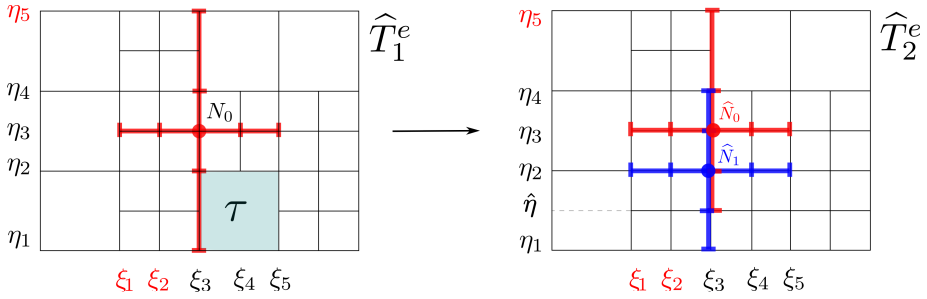
$$N_0 = B[\Xi]B[\mathcal{H}] = \frac{1}{4}B[\Xi]B[\mathcal{H}_1] + B[\Xi]B[\mathcal{H}_2] = \frac{1}{4}\widehat{N}_1 + \widehat{N}_0,$$

where $\mathcal{H}_1 = (\eta_1, \hat{\eta}, \eta_2, \eta_3, \eta_4)$ and $\mathcal{H}_2 = (\hat{\eta}, \eta_2, \eta_3, \eta_4, \eta_5)$. It is easy to see, that the functions $\widehat{N}_1$ and $\widehat{N}_0$ are the ones of the new blending functions from $S_{T_2^e}$. Hence $N_0 \in S_{T_2^e}$.

Note that using some additional reasoning we can reduce the study to a few number of possible cases. In Fig. 5.4(b) some knots of the function $N_0$ are marked in red. According to the knot insertion formula, the value of these knots is irrelevant for the study performed for the function. This implies that we do not need to analyse many other mesh configurations for this type of function, because they are equivalent to this studied case.




(a) The function is changed after refinement of $\tau$.




(b) The function is recuperated with functions of the new space.

**Figure 5.4:** *m-level function* $N_0$ *is split into two functions from the new space.* $N_0 = \frac{1}{4}\widehat{N}_1 + \widehat{N}_0$.

## 5.3 Linear independence

For now we cannot provide a rigorous proof of this property. It is currently under study. However, for a better understanding of the strategy, some clues should be mentioned.

For now, our strategy is designed for T-meshes without repeated interior knots. To prove the linear independence of our blending functions it is essential to take into account that each regular node of the T-mesh has only one function assigned to it. Some reasoning from [54] can be used to proof the linear independence of such functions. In particular, the authors show that T-splines without multiple interior knots can not be linearly dependent. Note that the example of linearly dependent T-splines given in [53] is for a T-mesh with multiple interior knots.

The main reason of the linear independence of our spline functions lies in the fact that all the functions have different anchors (the third knot of its local knot vectors), i.e., there exist no pair of functions that have the same anchor. Thus, the proof of linear independence of our functions boils down to prove that a set of spline functions can be linearly dependent only if there are some functions with the same anchor. This hypothesis seems reasonable taking into account the knot insertion procedure, see Section 2.1.2.1 and Fig. 2.5. According to this formula, a spline function $B_0$ is split in two functions $B_0 = \alpha B_1 + \beta B_2$ under a new knot insertion, where one of the functions has the same anchor as the original function $B_0$. So, for the linearly dependent set $\{B_0, B_1, B_2\}$ there are two functions with the same anchor.

It should be mentioned that the linear independence of our spline functions was verified in numerous computational experiments due to the non-singularity of the interpolation and stiffness matrices.

## 5.4 Non-negative weighted partition of unity

Partition of unity for spline functions provides some useful properties as affine invariance and convex hull property for spline curves and surfaces, which are very important for geometric design.

Taking into account the nesting behaviour of our spline spaces, the constant function 1 belongs to our spline space, i.e.,

$$1 = \sum_{\alpha \in A_T} w_\alpha N_\alpha.$$

Hence we can define weighted functions $N'_\alpha = w_\alpha N_\alpha$. This new basis $\{N'_\alpha\}_{\alpha \in A_T}$ forms a partition of unity:

$$1 = \sum_{\alpha \in A_T} N'_\alpha.$$

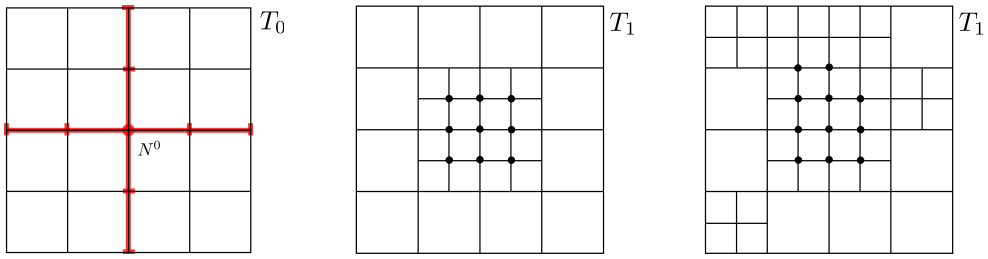



**Figure 5.5:** *Verification of the non-negativity for the coefficients of linear combination for 0-level function $N^0$; (a) uniform mesh $T_0$ and its uniform support $N^0$. (b)(c) some of the $2^{16}$ possible mesh refinements of the support $N^0$ and the corresponding functions of $T_1$ necessary to represent $N^0$.*

However, there are some issues that should be discussed.

(i) Firstly, it is important that the weights are non-negative. In this way the basis $\{N'_\alpha\}_{\alpha\in A_T}$ conserves the property of non-negativity. To verify this fact we reason as follows. Suppose that we start with the uniform mesh $T_0$ of level 0 whose functions form a strictly positive partition of unity: $1 = \sum_{\alpha\in A_{T_0}} N^0_\alpha$, see Fig. 5.5(a). Then, let us consider a refined mesh $T_1$ of level 1. We can show that any function $N^0$ of the mesh $T_0$ is a linear combination of the functions from $T_1$ with non-negative coefficients, i.e.,

$$N^0 = \sum_{i\in A_{T_1}} c_i N^1_i, \quad \text{where } c_i \geq 0.$$

For that we have to analyse all possible mesh configuration within the support of the function $N^0$ and verify for each of them that all coefficients of the linear combination to recover $N^0$ are non-negative. The support of the function $N^0$ comprises 16 cells of level 0 that can be refined or not on the mesh $T_1$, giving rise to different linear combinations to recuperate the function $N^0$. Figure 5.5(b) and (c) illustrates some examples of such configurations and their functions necessary to recuperate the function $N^0$. In total, there are $2^{16}$ different mesh configurations to study. To do this, we perform a computational experiment. We generate all possible $2^{16}$ mesh configurations and verify for each mesh $T_1$ that the function $N^0$ is a linear combination of the functions of $T_1$ with non-negative coefficients. For that we interpolate the function $N^0$ using the approximation space of the refined mesh $T_1$. This interpolation gains to reproduce exactly the function $N^0$ and all coefficients are non-negative. Having done this verification, we deduce that the

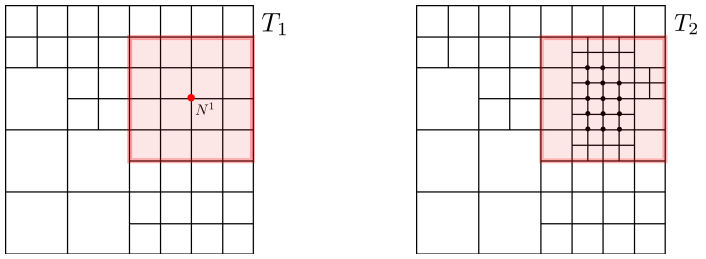



**Figure 5.6:** *Verification of the non-negativity for the coefficients of linear combination for a function $N^1$ of level 1.*

functions of the mesh $T_1$ form a non-negative partition of unity since

$$1 = \sum_{\alpha \in A_{T_0}} N_\alpha^0 = \sum_{\alpha \in A_{T_0}} \left( \sum_{i \in A_{T_1}} c_i^\alpha N_i^1 \right) = \sum_{i \in A_{T_1}} w_i N_i^1, \quad w_i \geq 0.$$

Next, for a T-mesh $T_2$ of level 2 we repeat the reasoning and show that any function $N^1$ of the mesh $T_1$ is a linear combination of the functions of $T_2$ with non-negative coefficients, see Fig. 5.6. Indeed, due to 0-balanced condition, a function support $N^1$ is replaced by the new functions of level 2, Fig. 5.6(b), only if the support of $N^1$ is composed of the 16 cells of level 1, i.e., it is a scaled version of the support $N^0$. So the entire procedure, described before for $N^0$, can be repeated to show that the linear combination to represent the function $N^1$ includes only non-negative coefficients. Thus, assuming the non-negative partition for the mesh $T_1$, we can deduce the non-negative partition for the mesh $T_2$:

$$1 = \sum_{i \in A_{T_1}} w_i N_i^1 = \sum_{i \in A_{T_1}} \left( w_i \sum_{j \in A_{T_2}} c_j^i N_j^2 \right) = \sum_{j \in A_{T_2}} w_j' N_j^2, \quad w_j' \geq 0.$$

Repeating the reasoning we can show the non-negative partition of unity for a mesh of any level k.

(ii) Second issue to discuss. There are mesh configurations for which some weights $w_i$ can be zero. For example, Fig. 5.7(a) shows an initial mesh $T_0$ whose basis forms a strictly positive partition of unity, since it is uniform bivariate B-splines, i.e., $1 = \sum_{\alpha \in A_{T_0}} N_\alpha$. After some refinements around the anchor of the function $N_0$, shown in Fig. 5.7(b), the new space corresponding to $T_1$ consists of the same basis function of the mesh $T_0$ (due to Extension rule 1) plus 5 new functions of the finer level. These 5 new functions are not necessary for a partition of unity. Hence, they should be annulled in order to construct a strictly positive partition of unity. However, if the four cells that share the anchor of $N_0$ are refined, then the

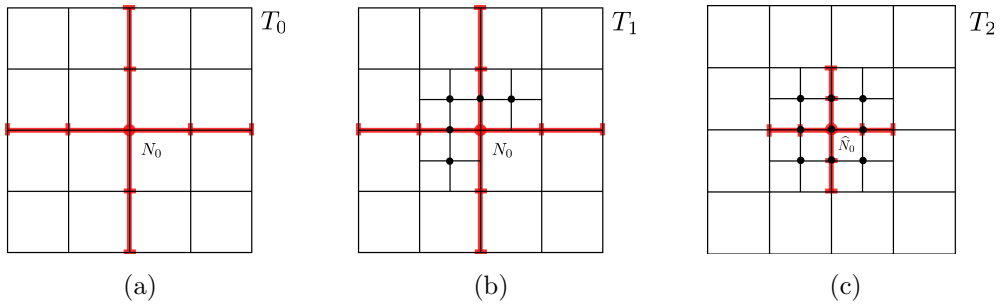



**Figure 5.7:** *(a) A mesh with partition of unity. (b) Refined mesh with some redundant functions. (c) Refined mesh without redundant functions and thus with strictly positive partition of unity.*

function $N_0$ is replaced by the finer ones, see Fig. 5.7(c). In this case, there are no redundant functions and the basis of the mesh $T_2$ form a strictly positive partition of unity. Also, a T-mesh can have some redundant functions due to Extension rule 2. Figure 5.8 shows such examples. The functions of the mesh of Fig. 5.8(a) form a strictly positive partition of unity. Then, it can be checked that the refined mesh of Fig. 5.8(b) has some new functions that are redundant for partition of unity. The anchors of active functions are marked by black circles, i.e., only these functions are necessary to form a partition of unity.

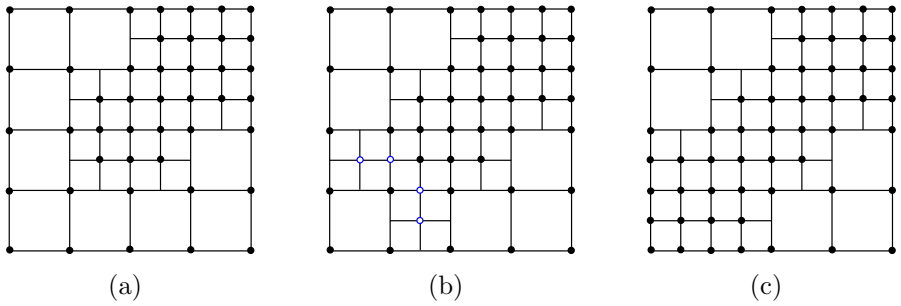



**Figure 5.8:** *(a) Initial mesh with strictly positive partition of unity. (b) Refined mesh with redundant functions due to Extension rule 2. Active functions are marked with black circles, redundant functions with blue circles. (c) Mesh without redundant functions.*

**Remark 10.** *Similar problem also happens in hierarchical refinement scheme. Therefore, an additional requirement is imposed on the refinement pattern, namely the refined zone should at least occupy the area of a function support of the current level in order to produce a replacement of this support by the finer ones. In this way, none of the new functions will be redundant for a partition of unity.*

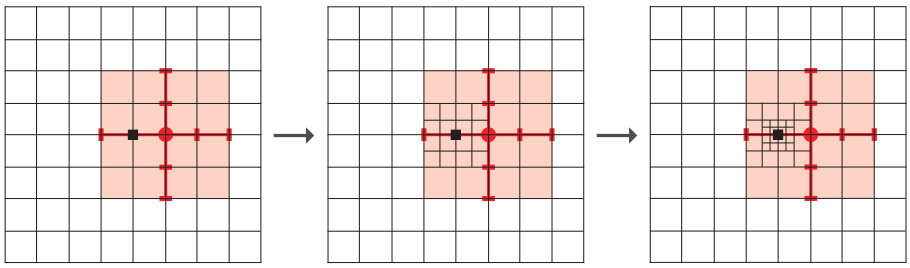
It should be mentioned that in our computational examples presented in this manuscript we do not apply any additional treatment for our spline spaces, so the basis used for the analysis does not form a partition of unity.

## 5.5 Locality of the functions and possible excessive support overlapping

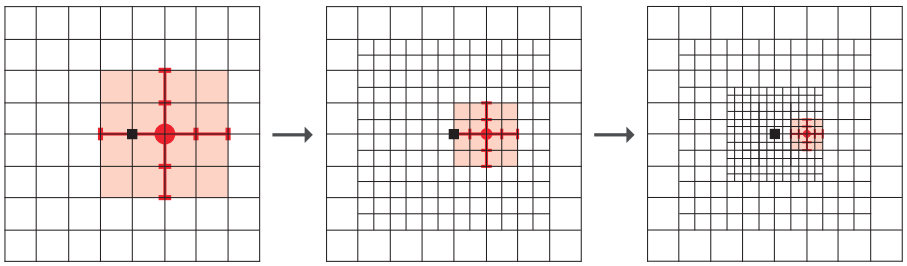
It should be mentioned that the proposed strategy can lead, in some cases, to increased overlapping of the function supports of different refinement levels, which can affect the conditioning and sparsity of the stiffness matrix. Excessive accumulation of the functions can take place in some problems with a very sharp singularity, when the area marked to refine by the error indicator at each iteration is too small. According to our strategy, a cell refinement always adds at least one new blending function of the finer level, while some functions of the coarse level can stay unchanged after a refinement. Note that a coarse level function support is reduced only if the four cells that contact its anchor are refined, see Fig. 5.7(c). Thus, if the adaptive refinement produces a sequence of small nested refined areas, some functions of the coarse levels stay active (not null) in the refined zone, see Fig. 5.9(a).

To avoid a possible accumulation of the functions of different levels, the following strategy can be adopted: if a function support intersects the cell marked to refine, then the coarsest cells sharing the anchor of the function are also refined. Application of this approach is illustrated in Fig. 5.9. In this example, let us suppose that the problem has a singularity at the center of the domain, and at each iteration, we refine only the four cells adjacent to the singularity point. Then at each iteration there are some functions whose support stays unchanged. However, if at each iteration we perform additional refinements, functions of the coarse level are replaced by the finer ones. Application of this additional refinement, where needed, can reduce an excessive function overlapping and improve the stiffness matrix conditioning and sparsity. However, due to the unnecessary extension of the refined zone, the optimal rate of convergence can be lost. In practice, we did not observe a significant advantage of this approach, and taking into account its computational cost, we do not apply this strategy in the computational examples presented in this work. Moreover, it was observed that for a large variety of problems, the excessive support overlapping is avoided naturally after various refinement steps due to the behaviour of the error indicator. Only for one computational example with a very strong singularity (section 7.2.8) the situation described in this paragraph takes place and additional refinements were applied.

On the other hand, another possible solution for this problem is a more accurate and selective definition of the functions in order to obtain spaces with a better function supports locality. We believe that the strategy we propose in this work can be improved by including a little more sophisticated rules for inferring local



(a) Three refinement steps around a singular point (black square) that lead to an accumulation of the function supports. Note that the red support is not changed during refinements.



(b) The same three refinement steps applying additional refinements. Coarse level support is reduced.

**Figure 5.9:** *Additional refinements in order to avoid excessive overlapping of the function supports.*

knot vectors to avoid an unnecessary support extensions in some cases.

**Remark 11.** *It should be mentioned that the similar problem happens for hierarchical refinement scheme. To avoid this problem, analogously to our case, some additional refinements can be imposed, and these refinements are more extensive than in our case. As another solution to this issue, a truncation technique was proposed in [30]. Truncated hierarchical splines are derived from the hierarchical basis by redefining the functions, namely, eliminating the contributions of functions of finer levels to coarser level ones, and thus improving their locality. Truncated basis has smaller supports, non-negativity, partition of unity and local linear independence. Besides, the construction of spline spaces using truncated hierarchical B-splines is strongly stable versus weak stability of classical hierarchical refinement scheme, see [57].*

*Also, it is worth mentioning that a possible supports accumulation in a cell is inherent to the tensor product structure of the basis functions. Note that support accumulation also can take place with T-splines. Namely, it happens for the mesh refinements shown in Figure 5.9(a). However, for T-splines the support overlapping is considerably minor and it happens only for this particular situation.*

# Appendix

## 5.A Nesting behaviour of the spaces

In this section we give a proof of the nestedness of the spline spaces constructed by means of our technique (2D case).

### 5.A.1 Step 1 of the proof

First, we perform some reasoning in order to show that the proof can be reduced to a study of a few number of some basis cases.

**Lemma 1.** *For any two 0-balanced nested meshes $T_1 \subset T_2$ there exists a sequence of meshes $\{T_i^e\}_{i=0}^n$ such that*

*(i)* $T_1 = T_0^e \subset T_1^e \subset T_2^e \subset \cdots \subset T_{n-1}^e \subset T_n^e = T_2$.

*(ii) any mesh $T_i^e$ is obtained from the mesh $T_{i-1}^e$ by refining only one cell.*

*(iii) all the meshes $\{T_i^e\}_{i=0}^n$ are 0-balanced.*

*Proof.* With abuse of notation, for a given T-mesh $T$, we denote its corresponding quadtree also as $T$. The term *tree node* is used exclusively in the context of the tree structures terminology, see section 4.2. Let us consider a 0-balanced quadtree mesh $T_1$. And let $T_2$ be a 0-balanced refinement of $T_1$. We denote $k_1 = \text{lev}(T_1)$ and $k_2 = \text{lev}(T_2)$, then $k_1 \leqslant k_2$. The set of all internal nodes of the quadtree $T_2$ that do not belong to $T_1$ is denoted by $\mathcal{N} = \{\mathcal{N}_i\}_{i=1}^n$. Elements of the set $\mathcal{N}$ can be sorted by increasing order of their levels: $\mathcal{N} := \{\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_n\}$, where $k_1 \leqslant \text{lev}(\mathcal{N}_i) \leqslant \text{lev}(\mathcal{N}_{i+1}) \leqslant k_2$ for any $i = 1, ..., n-1$. Then, we start to refine the mesh $T_1$, adding to the corresponding quadtree one by one the elements of the sorted set $\mathcal{N}$. The addition of a new internal tree node corresponds to the refinement of one cell of the previous mesh. Note that internal node is a tree node that has *children*. So, the addition of the internal node $\mathcal{N}_i$ of level $m$ means that a m-level leaf node (cell) turns into the internal node after being subdivided. Thus, we construct a sequence of refinements that leads to the final mesh: $T_1 = T_0^e \subset T_1^e \subset T_2^e \subset \cdots \subset T_{n-1}^e \subset T_n^e = T_2$, see Fig. 5.A.1.

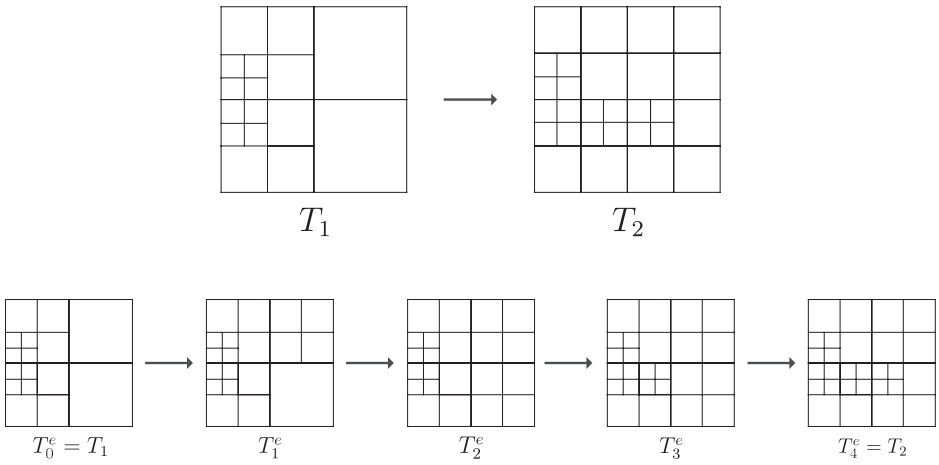



**Figure 5.A.1:** *Elemental refinements. A possible sequence of* 0*-balanced meshes* $\{T_i^e\}$*, where* $T_i^e$ *is obtained from* $T_{i-1}^e$ *by refining only one cell.*

Now, we have to assure that each of these meshes is 0-balanced. Indeed, let us suppose that the mesh $T_i^e$ is the first one of the sequence that is not 0-balanced, i.e., the mesh $T_{i-1}^e$ is 0-balanced and $T_i^e$ is not. If $T_i^e$ is obtained from $T_{i-1}^e$ by refining a cell of level $m$, then, in order to obtain the final 0-balanced mesh $T_2$, at least one neighbouring cell of level *(m-1)* must be refined. By construction (the nodes are sorted in increasing order), this does not occur, i.e., no cell of level *(m-1)* is refined later. Therefore, the rest of the meshes of the sequence, including $T_2$, are also 0-unbalanced. This conclusion contradicts the assumption that the mesh $T_2$ is 0-balanced. □

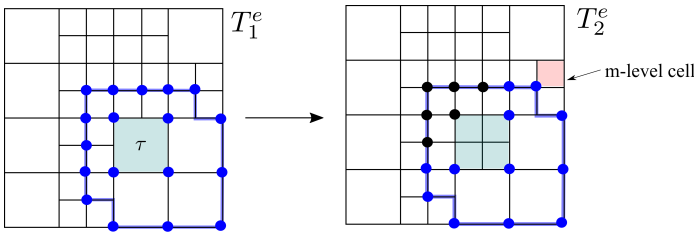
**Definition 3.** Refinements that fulfil the conditions of the Lemma 1 are called *elemental refinement*, i.e., a pair of 0-balanced meshes $T_1^e \subset T_2^e$ is called *elemental refinement* if $T_2^e$ is obtained from $T_1^e$ by refining only one cell.

**Lemma 2.** *If any elemental refinement produces nested spaces:* $T_1^e \subset T_2^e \;\Rightarrow\; S_{T_1^e} \subset S_{T_2^e}$*, then any two 0-balanced nested meshes produce nested spaces:*

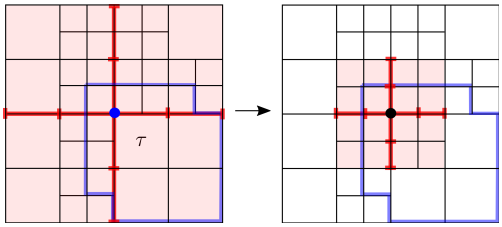
$$T_1 \subset T_2 \;\Rightarrow\; S_{T_1} \subset S_{T_2}.$$

*Proof.* For the sequence of elemental refinements constructed in Lemma 1 we have: $S_{T_1} = S_{T_0^e} \subset S_{T_1^e} \subset S_{T_2^e} \subset \ldots \subset S_{T_n^e} = S_{T_2}$, and therefore $S_{T_1} \subset S_{T_2}$. □

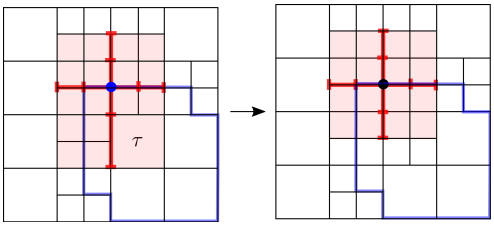
Now, taking into account Lemma 2, our goal is to prove that any elemental refinement produces nested spaces. To proof this fact, it is crucial to take into account several considerations about our strategy. As was discussed in section 4.6, there are only few types of possible function supports. Each function is determined by its knot intervals $\boldsymbol{\Delta}^\xi = (\Delta_1^\xi, \Delta_2^\xi, \Delta_3^\xi, \Delta_4^\xi)$ and $\boldsymbol{\Delta}^\eta = (\Delta_1^\eta, \Delta_2^\eta, \Delta_3^\eta, \Delta_4^\eta)$. Due to the extension rules and the balanced mesh condition, we have the following possible



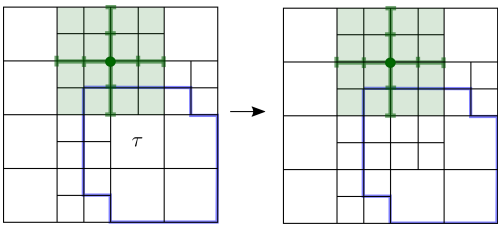

(a) One-ring neighbourhood of the cell $\tau$.



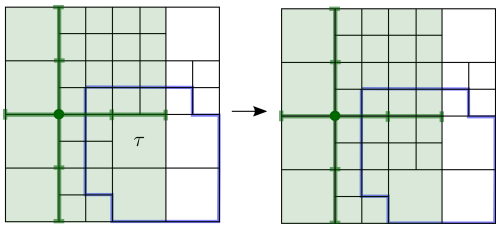

(b) (m-1)-level function of the ring turns into m-level function.




(c) m-level function of the ring turns into another m-level function.




(d) m-level function, with the anchor out of the ring, is not affected.




(e) (m-1)-level function, with the anchor out of the ring, is not affected.

**Figure 5.A.2:** *An example of elemental refinement $T_1^e \subset T_2^e$ and the one-ring neighbourhood of the cell $\tau$. (a) The contour of the one-ring neighbourhood and the anchors of its functions are marked in blue. Anchors, whose functions are changed after the cell refinement, are marked in black in the rightmost figure. (b) and (c) The functions of the one-ring neighbourhood that are changed after the refinement. (d) and (c) Functions whose anchors are out of the ring always stay unchanged.*

configurations for the set $\boldsymbol{\Delta}$: $(h, h, h, h)$, $(h, h, h, 2h)$, $(2h, h, h, h)$ and $(2h, h, h, 2h)$. It is important to highlight that the same function support can take place for different underlying meshes, since we have to skip some knots of the mesh in order to fulfil Condition 1 and Condition 2, see Section 4.4. Consequently, refinement of a cell does not affect some functions whose support is determined by the cells of a coarser level.

Having said that, refinement of a cell affects only some of the function supports in the neighbourhood the cell. Namely, the anchors of the affected function supports belong to the *one-ring neighbourhood* of the cell. We are going to call *one-ring neighbourhood* of the cell $\tau$ the set of all cells that have contact with this

cell, see Fig. 5.A.2(a). Let us analyse in details which functions can be changed by the refinement of *(m-1)*-level cell $\tau$. Suppose we have a mesh $T_1^e$, where a cell $\tau$ is refined leading to a new mesh $T_2^e$. Let us see what can happen with the functions whose anchors are situated out of the ring. Any function support of level $\geq m$, whose anchor is out of the ring, will not be affected by the refinement, since its support does not intersect the cell $\tau$, see Fig. 5.A.2(d); and any function support of level $\leq (m-1)$, whose anchor is out of the ring, is not affected by the refinement due to Condition 1 of our strategy, see Fig. 5.A.2(e). Thus, any function support, whose anchor is situated out of the ring, stays unchanged after refinement of the cell $\tau$. In regard with the anchors of the one-ring neighbourhood of the cell, two situations can take place. Note that due to the balanced mesh condition, one-ring neighbourhood of the cell $\tau$ contains only functions of level *(m-1)* and $m$. The first situation: a function of level *(m-1)* is changed and becomes a function of level $m$. This can happen only when the other three cells, sharing its anchors, are of level $m$, see Fig. 5.A.2(b). The second situation: a function of level $m$ is changed to another function of level $m$, see for example Fig. 5.A.2(c). Thus, we have to verify that any function support of the mesh $T_1^e$, from the one-ring neighbourhood of the cell $\tau$, that is changed after the refinement of the cell, can be recuperated with the functions of the refined mesh $T_2^e$.

First, we consider the simplest case of elemental refinement.

**Definition 4.** We say that $\widehat{T}_1^e \subset \widehat{T}_2^e$ is a *simple elemental refinement*, if the mesh $\widehat{T}_2^e$ is obtained from $\widehat{T}_1^e$ by refining a *(m-1)*-level cell and $\mathrm{lev}(\widehat{T}_2^e) = m$.

That is, for a simple elemental refinement $\widehat{T}_1^e \subset \widehat{T}_2^e$, a *(m-1)*-level cell $\tau$ of the mesh $\widehat{T}_1^e$ is refined, creating four new cells of level $m$ in $\widehat{T}_2^e$, where the mesh $\widehat{T}_2^e$ does not contain cells of level finer than $m$, see Fig. 5.A.3(b). We can show that

**Proposition 1.** *Any simple elemental refinement produces nested spaces:*

$$\widehat{T}_1^e \subset \widehat{T}_2^e \quad \Rightarrow \quad S_{\widehat{T}_1^e} \subset S_{\widehat{T}_2^e}.$$

Proposition 1 is proved in the next subsection. Now, assuming this proposition, we can show the nesting behaviour also for any general elemental refinement: $\mathrm{lev}(T_2^e) > m$, see Fig. 5.A.3(a).

**Proposition 2.** *Any elemental refinement produces nested spaces:*

$$T_1^e \subset T_2^e \quad \Rightarrow \quad S_{T_1^e} \subset S_{T_2^e}.$$

*Proof.* If the elemental refinement is a simple one, then the statement is proved. Otherwise, the mesh $T_2^e$ is such that it is obtained from the mesh $T_1^e$ by refining a *(m-1)*-level cell $\tau$ and $\mathrm{lev}(T_2^e) > m$, i.e., the mesh $T_2^e$ and $T_1^e$ include cells of level greater than $m$, see Fig. 5.A.3(a). As was mentioned before, the anchors whose

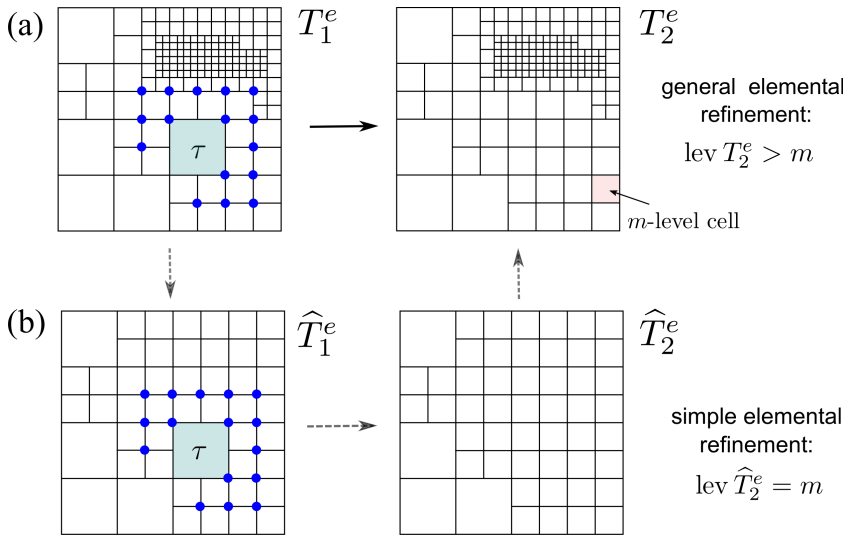



**Figure 5.A.3:** *General elemental refinement $T_1^e \subset T_2^e$, can be reduced to a simple case of elemental refinement $\widehat{T}_1^e \subset \widehat{T}_2^e$, where* $\operatorname{lev}(\widehat{T}_2^e) = m$.

functions are changed after the cell refinement are situated in its one-ring neighbourhood and, due to 0-balanced condition, this ring only contains cells of level $m$ and *(m-1)*. The elemental refinement produces changes only in the functions of level $m$ and *(m-1)*. So we have to verify that these functions can be recovered by the new space $S_{T_2^e}$. Let us denote the set of these functions as $S^* = \{N_i\}_{i \in I}$, $N_i \in S_{T_1^e}$. The anchors of these functions are represented by blue circles in Fig. 5.A.3(a). To show that $S^* \subset S_{T_2^e}$ we proceed as follows.

Due to Extension rule 1, the supports of these functions do not depend on the cells of level greater than $m$. We are going to use a simple elemental refinement $\widehat{T}_1^e \subset \widehat{T}_2^e$. Let mesh $\widehat{T}_1^e$ be a derefinement of $T_1^e$ obtained by eliminating from it all cells of level greater that $m$. We know that this derefinement does not alter the functions of $S^*$, that is, $S^* \subset S_{\widehat{T}_1^e}$. Analogously, the mesh $\widehat{T}_2^e$ is a derefinement of $T_2^e$, then the elemental refinement $\widehat{T}_1^e \subset \widehat{T}_2^e$ is a simple elemental refinement. In virtue of Proposition 1, $S_{\widehat{T}_1^e} \subset S_{\widehat{T}_2^e}$. Taking into account that $S^* \subset S_{\widehat{T}_1^e}$, we have that $S^* \subset S_{\widehat{T}_1^e} \subset S_{\widehat{T}_2^e}$, and thus, $S^* \subset S_{\widehat{T}_2^e}$. Next, we have to show that $S_{\widehat{T}_2^e} \subset S_{T_2^e}$.

For the meshes $\widehat{T}_2^e$ and $T_2^e$ we can construct a sequence of simple elemental refinements $\widehat{T}_2^e = \bar{T}_0^e \subset \bar{T}_1^e \subset \bar{T}_2^e \subset \cdots \subset \bar{T}_{n-1}^e \subset \bar{T}_n^e = T_2^e$. Then, applying Proposition 1 to each pair of this sequence, we obtain $S_{\widehat{T}_2^e} \subset S_{T_2^e}$. Thus, $S^* \subset S_{T_2^e}$, which implies that $S_{T_1^e} \subset S_{T_2^e}$. □

Thus, in this section we have shown that the proof of nesting behaviour is reduced to Proposition 1. The whole reasoning to prove the nesting behaviour for elemental refinements is resumed in the scheme shown in Fig. 5.A.4. As was said before, a general case of elemental refinement (Proposition 2) is reduced to

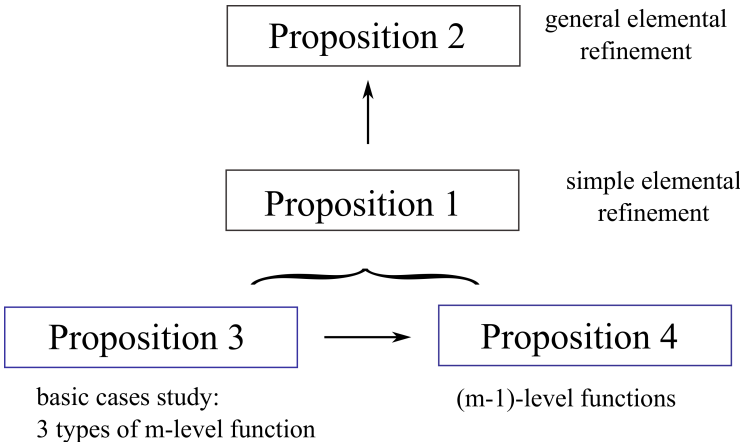



**Figure 5.A.4:** *Scheme of the proof of nesting behaviour for elemental refinements.*

a simple elemental refinement (Proposition 1), which in its turn is split into the study of functions of two levels: Proposition 3 (level $m$) and Proposition 4 (level *m-1*).

Next, we continue the reasoning to proof Proposition 1.

## 5.A.2 Step 2. Proof of Proposition 1

We have to show that for any simple elemental refinement $\widehat{T}_1^e \subset \widehat{T}_2^e$ all function supports of the mesh $\widehat{T}_1^e$, from the one-ring neighbourhood of the cell $\tau$, can be recuperated by the functions of the refined mesh $\widehat{T}_2^e$.

When *(m-1)*-level cell $\tau$ is refined, the functions of its one-ring neighbourhood can be separated in three groups:
(1) the function of level $m$ that are replaced by another $m$-level function,
(2) the function of level *(m-1)* that are replaced by a $m$-level function and
(3) the function of level *(m-1)* and $m$ that stay unchanged after the refinement.

We are interested in verifying that the functions of the group (1) and (2) can be recovered by the new space $S_{\widehat{T}_2^e}$. Thus, the proof of Proposition 1 is split into two parts: Proposition 3 for the group (1) and Proposition 4 for the group (2).

First, we have to study the group (1), because it is necessary for the study of the group (2).

**Proposition 3.** *For any simple elemental refinement $\widehat{T}_1^e \subset \widehat{T}_2^e$ all $m$-level functions of $S_{\widehat{T}_1^e}$ can be reproduced with the functions from $S_{\widehat{T}_2^e}$.*

Proof of this assertion is extensive and we give the proof in the next subsection. Next, taking into account Proposition 3, we show that

**Proposition 4.** *For any simple elemental refinement $\widehat{T}_1^e \subset \widehat{T}_2^e$ all (m-1)-level functions of $S_{\widehat{T}_1^e}$ can be reproduced with the functions from $S_{\widehat{T}_2^e}$.*

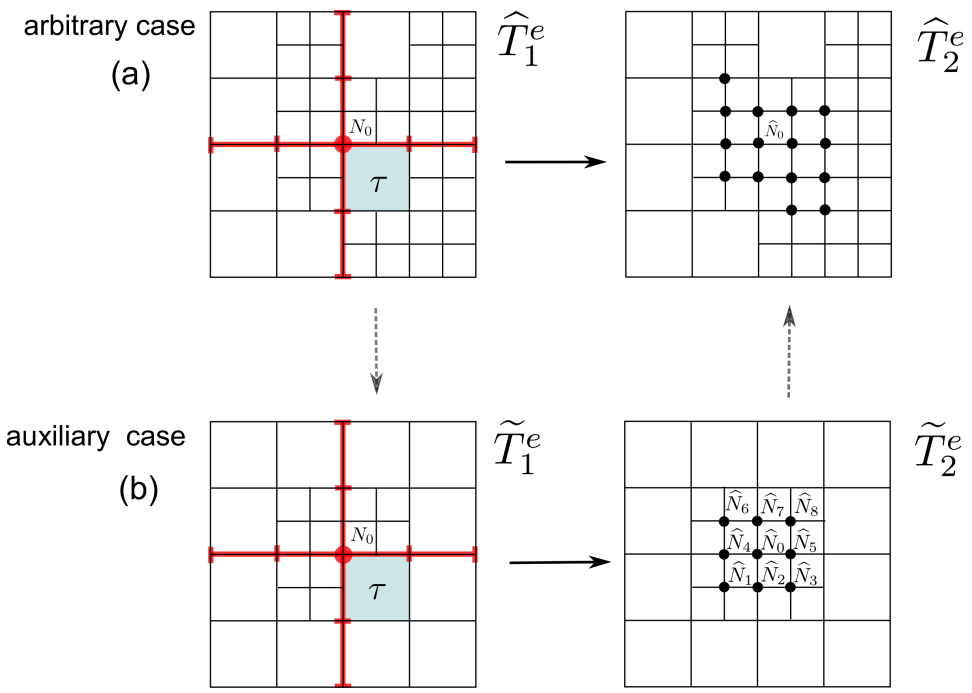



**Figure 5.A.5:** *(a) A simple elemental refinement $\widehat{T}_1^e \subset \widehat{T}_2^e$ that leads to replacement of (m-1)-level function $N_0$ by a new one $\widehat{N}_0$ of level $m$. (b) Auxiliary case $\widetilde{T}_1^e \subset \widetilde{T}_2^e$: (m-1)-level function $N_0$ is a linear combination of 9 $m$-level functions $\widehat{N}_i \in S_{\widetilde{T}_2^e}$, $i = 1, 9$.*

*Proof.* Let us consider a *(m-1)*-level function $N_0 \in S_{\widehat{T}_1^e}$. The function $N_0$ is replaced by the new one $\widehat{N}_0$ of level $m$, after the refinement of the cell $\tau$, only if other three cells sharing its anchor are of level $m$, see Fig. 5.A.5(a). We have to show that $N_0 \in S_{\widehat{T}_2^e}$ for any possible mesh configuration $\widehat{T}_1^e$.

First, we study an auxiliary case $\widetilde{T}_1^e \subset \widetilde{T}_2^e$ illustrated in Fig. 5.A.5(b). For this auxiliary case only 3 cells of level *(m-1)* are refined. This is the "minimal" possible mesh that leads to the replacement of the function $N_0$ by the new $m$-level function $\widehat{N}_0$ after refining the cell $\tau$. In other words, any other mesh $\widehat{T}_1^e$ that leads to this change is a refinement of $\widetilde{T}_1^e$. It is easy to see that, for this auxiliary case, $N_0$ can be expressed as linear combination of nine $m$-level blending functions of $\widetilde{T}_2^e$, see Fig. 5.A.5(b)

$$N_0 = \frac{1}{4}\widehat{N}_1 + \frac{3}{8}\widehat{N}_2 + \frac{1}{4}\widehat{N}_3 + \frac{3}{8}\widehat{N}_4 + \frac{9}{16}\widehat{N}_0 + \frac{3}{8}\widehat{N}_5 + \frac{1}{4}\widehat{N}_6 + \frac{3}{8}\widehat{N}_7 + \frac{1}{4}\widehat{N}_8. \qquad (5.1)$$

**Remark 12.** *The expression (5.1) is obtained using knot insertion formula of Section 2.1.2.1 for the knot vectors of the function $N_0$, and it is for a function without repeated boundary knots. Otherwise, the set of functions from $\widetilde{T}_2^e$ and their coefficients, necessary to recover the function $N_0$, can be different, but this does not*

*affect the reasoning. The only important thing is that* $N_0 \in S_{\widetilde{T}_2^e}$ *for this auxiliary case.*

Now, using this auxiliary case, we can proof the statement for arbitrary simple elemental refinement $\widehat{T}_1^e \subset \widehat{T}_2^e$. The mesh $\widehat{T}_1^e$ is obtained from the mesh $\widetilde{T}_1^e$ by refining some cells of level *(m-1)* (except $\tau$) situated in the interior of the support of $N_0$. Analogously, $\widehat{T}_2^e$ is a refinement of $\widetilde{T}_2^e$. Thus, we have: $\widetilde{T}_1^e \subset \widetilde{T}_2^e \subset \widehat{T}_2^e$ and $\text{lev}(\widetilde{T}_1^e) = \text{lev}(\widetilde{T}_2^e) = \text{lev}(\widehat{T}_2^e) = m$. It is important to notice that the function $N_0$ is the same for $\widehat{T}_1^e$ and $\widetilde{T}_1^e$, i.e., $N_0 \in S_{\widetilde{T}_1^e}$. For the auxiliary elemental refinement $\widetilde{T}_1^e \subset \widetilde{T}_2^e$ we already know that the function $N_0$ can be recovered by the nine $m$-level blending functions $\{\widehat{N}_i\}_{i=1}^9$ of $\widetilde{T}_2^e$. Hence, to show that $N_0 \in S_{\widehat{T}_2^e}$, it is sufficient to show that $\widehat{N}_i \in S_{\widehat{T}_2^e},\ i = 1, ..., 9$. Indeed, for the meshes $\widetilde{T}_2^e \subset \widehat{T}_2^e$ we can construct a sequence of simple elemental refinements, where only *(m-1)*-level cells are refined, and $m$-level functions $\widehat{N}_i$ can be recovered by $S_{\widehat{T}_2^e}$ in virtue of Proposition 3. So we conclude that $\widehat{N}_i \in S_{\widehat{T}_2^e},\ i = 1, .., 9$. Thus, $N_0 \in S_{\widehat{T}_2^e}$. □

Finally, we arrive to the core of the whole reasoning, that is Proposition 3, whose proof includes the study of certain number of basic cases.

### 5.A.2.1 Proof of Proposition 3

To prove Proposition 3 we are going to use the property of refinability of B-splines and knot insertion formula.

#### 5.A.2.1.1 Knot insertion and B-spline refinement

If a knot $\hat{\xi}$ is inserted into a knot vector $\Xi = (\xi_1, \xi_2, \xi_3, \xi_4, \xi_5)$ then the univariate B-spline $B[\Xi]$ can be written as linear combination of two new B-spline functions, i.e.,

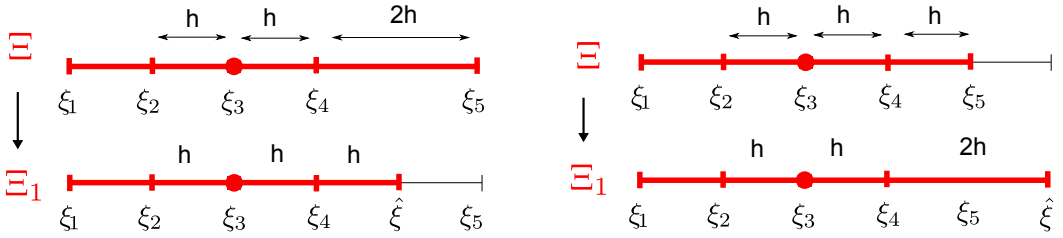
$$B[\Xi] = \alpha B[\Xi_1] + \beta B[\Xi_2],$$

where $\Xi_1 = (\xi_1, \ldots, \hat{\xi}, \ldots, \xi_4)$ and $\Xi_2 = (\xi_2, \ldots, \hat{\xi}, \ldots, \xi_5)$. The coefficients $\alpha$ and $\beta$ are calculated by the formula

$$\alpha = \begin{cases} 1, & \xi_4 \le \hat{\xi} \le \xi_5 \\ \dfrac{\hat{\xi} - \xi_1}{\xi_4 - \xi_1}, & \xi_1 \le \hat{\xi} \le \xi_4 \end{cases}, \qquad \beta = \begin{cases} \dfrac{\xi_5 - \hat{\xi}}{\xi_5 - \xi_2}, & \xi_2 \le \hat{\xi} \le \xi_5 \\ 1, & \xi_1 \le \hat{\xi} \le \xi_2 \end{cases}. \tag{5.2}$$

For a bivariate B-spline function $B[\Xi, \mathcal{H}](\xi, \eta) = B[\Xi](\xi) B[\mathcal{H}](\eta)$ a knot insertion for one parametric direction produces a splitting into two functions

$$\begin{aligned} &B[\Xi, \mathcal{H}](\xi, \eta) = \left(\alpha_1 B[\Xi_1](\xi) + \beta_1 B[\Xi_2](\xi)\right) B[\mathcal{H}](\eta) = \\ &= \alpha_1 B[\Xi_1](\xi) B[\mathcal{H}](\eta) + \beta_1 B[\Xi_2](\xi) B[\mathcal{H}](\eta) = \alpha_1 B_1(\xi, \eta) + \beta_1 B_2(\xi, \eta). \end{aligned} \tag{5.3}$$




(a) Scenario 1. $(x, h, h, 2h) \to (x, h, h, h)$. (b) Scenario 2. $(x, h, h, h) \to (x, h, h, 2h)$.

**Figure 5.A.6:** *Two possible scenarios for a knot vector of m-level function after the refinement.*

If both knot vectors undergo a knot insertion, then the function $B(\xi, \eta)$ is split into four functions

$$\begin{aligned} &B[\Xi, \mathcal{H}](\xi, \eta) = (\alpha_1 B[\Xi_1](\xi) + \beta_1 B[\Xi_2](\xi))(\alpha_2 B[\mathcal{H}_1](\eta) + \beta_2 B[\mathcal{H}_2](\eta)) = \\ &= \alpha_1\alpha_2 B[\Xi_1](\xi)B[\mathcal{H}_1](\eta) + \alpha_1\beta_2 B[\Xi_1](\xi)B[\mathcal{H}_2](\eta) + \\ &+ \beta_1\alpha_2 B[\Xi_2](\xi)B[\mathcal{H}_1](\eta) + \beta_1\beta_2 B[\Xi_2](\xi)B[\mathcal{H}_2](\eta) = \\ &= c_1 B_1(\xi, \eta) + c_2 B_2(\xi, \eta) + c_3 B_3(\xi, \eta) + c_4 B_4(\xi, \eta). \end{aligned} \tag{5.4}$$

Now, we analyse the possible modifications that can undergo a knot vector $\Xi$ of a function of level $m$ after the cell refinement. We have two possible scenarios:

**scenario 1**. The knot vector $\Xi$ is reduced after the refinement. Suppose we have the original knot vector $\Xi = (\xi_1, \xi_2, \xi_3, \xi_4, \xi_5) = (\xi_1, \xi_3 - h, \xi_3, \xi_3 + h, \xi_3 + 3h)$, associated to the anchor $\xi_3$, and its knot intervals are $\boldsymbol{\Delta} = (\Delta_1, \Delta_2, \Delta_3, \Delta_4) = (x, h, h, 2h)$, where $x$ can be 0, $h$ or $2h$. After the refinement the vector $\Xi$ turns into the vector $\Xi_1 = (\xi_1, \xi_2, \xi_3, \xi_4, \hat{\xi})$ with knot intervals $\boldsymbol{\Delta}_1 = (x, h, h, h)$, see Fig. 5.A.6(a). This can be seen as an insertion of the knot $\hat{\xi} = \xi_3 + 2h$ between the fourth and the fifth knots of $\Xi$, then $\Xi_1 = (\xi_1, \xi_2, \xi_3, \xi_4, \hat{\xi})$ and $\Xi_2 = (\xi_2, \xi_3, \xi_4, \hat{\xi}, \xi_5)$. Consequently, using the formula (5.2),

$$B[\Xi] = B[\Xi_1] + \frac{1}{4} B[\Xi_2]. \tag{5.5}$$

Analogously, the vector $\Xi$ with knot intervals $\boldsymbol{\Delta} = (\Delta_1, \Delta_2, \Delta_3, \Delta_4) = (2h, h, h, x)$, turns into the vector $\Xi_2$ with knot intervals $\boldsymbol{\Delta}_1 = (h, h, h, x)$. That is, for the knot vector $\Xi = (\xi_1, \xi_2, \xi_3, \xi_4, \xi_5) = (\xi_3 - 3h, \xi_3 - h, \xi_3, \xi_3 + h, \xi_5)$, the knot $\hat{\xi} = \xi_3 - 2h$ is inserted between the first and the second knots, then $\Xi_1 = (\xi_1, \hat{\xi}, \xi_2, \xi_3, \xi_4)$, $\Xi_2 = (\hat{\xi}, \xi_2, \xi_3, \xi_4, \xi_5)$ and

$$B[\Xi] = \frac{1}{4} B[\Xi_1] + B[\Xi_2]. \tag{5.6}$$

**scenario 2**. The knot vector $\Xi$, associated to the anchor $\xi_3$ becomes larger after the refinement due to Extension rule 2. Namely, the vector $\Xi = (\xi_1, \xi_2, \xi_3, \xi_4, \xi_5) = (\xi_1, \xi_3-h, \xi_3, \xi_3+h, \xi_3+2h)$, with knot intervals $\boldsymbol{\Delta} = (\Delta_1, \Delta_2, \Delta_3, \Delta_4) = (x, h, h, h)$, turns into the vector $\Xi_1 = (\xi_1, \xi_2, \xi_3, \xi_4, \hat{\xi})$, where $\hat{\xi} = \xi_3+3h$, and its knot intervals becomes $\boldsymbol{\Delta}_1 = (x, h, h, 2h)$, see Fig. 5.A.6(b). Then, $\Xi_2 = (\xi_2, \xi_3, \xi_4, \xi_5, \hat{\xi})$ and the original function is split into two new functions

$$B[\Xi] = B[\Xi_1] - \frac{1}{4}B[\Xi_2]. \tag{5.7}$$

Analogously, if the knot vector is extended in the negative direction, i.e.,
$\Xi = (\xi_3 - 2h, \xi_3 - h, \xi_3, \xi_3 + h, \xi_5)$, $\Xi_1 = (\xi_3 - 3h, \xi_3 - 2h, \xi_3 - h, \xi_3, \xi_3 + h)$ and
$\Xi_2 = (\xi_3 - 3h, \xi_3 - h, \xi_3, \xi_3 + h, \xi_3 + 2h)$, then

$$B[\Xi] = -\frac{1}{4}B[\Xi_1] + B[\Xi_2]. \tag{5.8}$$

And, if the knot vector is extended in both directions, i.e.,
$\Xi = (\xi_3 - 2h, \xi_3 - h, \xi_3, \xi_3 + h, \xi_3 + 2h)$,
$\Xi_1 = (\xi_3 - 3h, \xi_3 - 2h, \xi_3 - h, \xi_3, \xi_3 + h)$,
$\Xi_2 = (\xi_3 - 3h, \xi_3 - h, \xi_3, \xi_3 + h, \xi_3 + 3h)$ and
$\Xi_3 = (\xi_3 - h, \xi_3, \xi_3 + h, \xi_3 + 2h, \xi_3 + 3h)$, then

$$B[\Xi] = -\frac{1}{4}B[\Xi_1] + B[\Xi_2] - \frac{1}{4}B[\Xi_3]. \tag{5.9}$$

#### 5.A.2.1.2 Basic cases study

**Proposition 3.** *For a simple elemental refinement $\widehat{T}_1^e \subset \widehat{T}_2^e$ any m-level function of $S_{\widehat{T}_1^e}$ can be reproduced with functions from $S_{\widehat{T}_2^e}$.*

*Proof.* To study thoroughly all possible cases, $m$-level function supports can be divided in three types according to the position of its anchor with respect to the center of the cell $\tau$, as shown in Fig. 5.A.7. Type 1 corresponds to the anchors aligned with an edge of the cell $\tau$. Type 2 anchors are aligned with a diagonal of the cell and type 3 with a symmetry axis of the cell. For each of the three types we are going to study all possible situations when the function $N_0$, associated to a certain anchor, undergoes some changes after refining the cell $\tau$, and show that it can be recovered by the new space $S_{\widehat{T}_2^e}$.

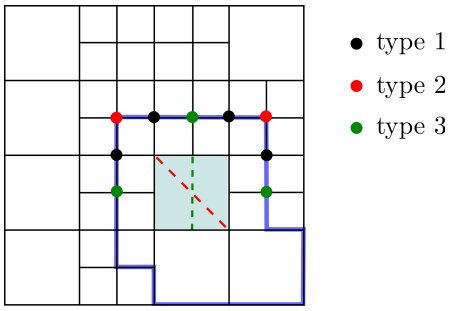


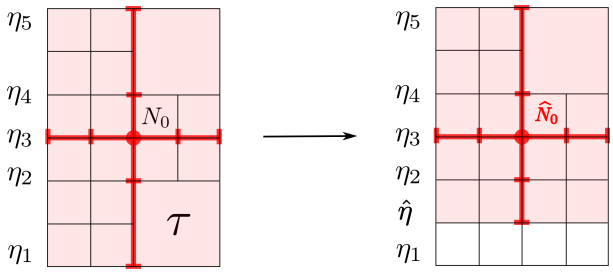

**Figure 5.A.7:** *m-level functions from the one-ring neighbourhood of the $\tau$ are classified in three types depending on the position of the anchor with respect to the cell $\tau$.*

We adopt the following notation:

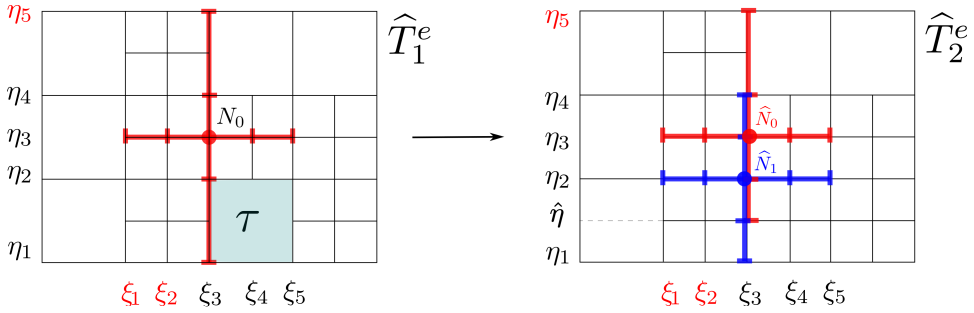
- The function of the mesh $\widehat{T}_1^e$ to study is denoted by $N_0$.
- The blending functions from the refined mesh $\widehat{T}_2^e$ are denoted by $\widehat{N}_i$.
- The functions that belong to the new space $S_{\widehat{T}_2^e}$ are denoted by $\bar{N}_i$. That is, the function $\bar{N}_i$ can be a blending function of $\widehat{T}_2^e$ or a linear combination of those.

**Type 1.** For this type of function only one of its local knot vectors can be changed after the refinement of the cell $\tau$, and only the scenario 1 can take place. Without loss of generality, suppose that the initial knot vector of the function $N_0$ is $\mathcal{H} = (\eta_1, \eta_2, \eta_3, \eta_4, \eta_5)$, and after the cell refinement the new function $\widehat{N}_0$ associated to the same anchor, has its knot vector $\mathcal{H}_2 = (\hat{\eta}, \eta_2, \eta_3, \eta_4, \eta_5)$, see Fig. 5.A.8. Then, this situation can be seen as an insertion of the new knot $\hat{\eta}$ between the first and the second knot of the vector $\mathcal{H} = (\eta_1, \eta_2, \eta_3, \eta_4, \eta_5)$. Then, in virtue of (5.3) and (5.6)

$$N_0 = B[\Xi]B[\mathcal{H}] = \frac{1}{4}B[\Xi]B[\mathcal{H}_1] + B[\Xi]B[\mathcal{H}_2] = \frac{1}{4}\widehat{N}_1 + \widehat{N}_0,$$




(a) Modification of the function after the refinement.



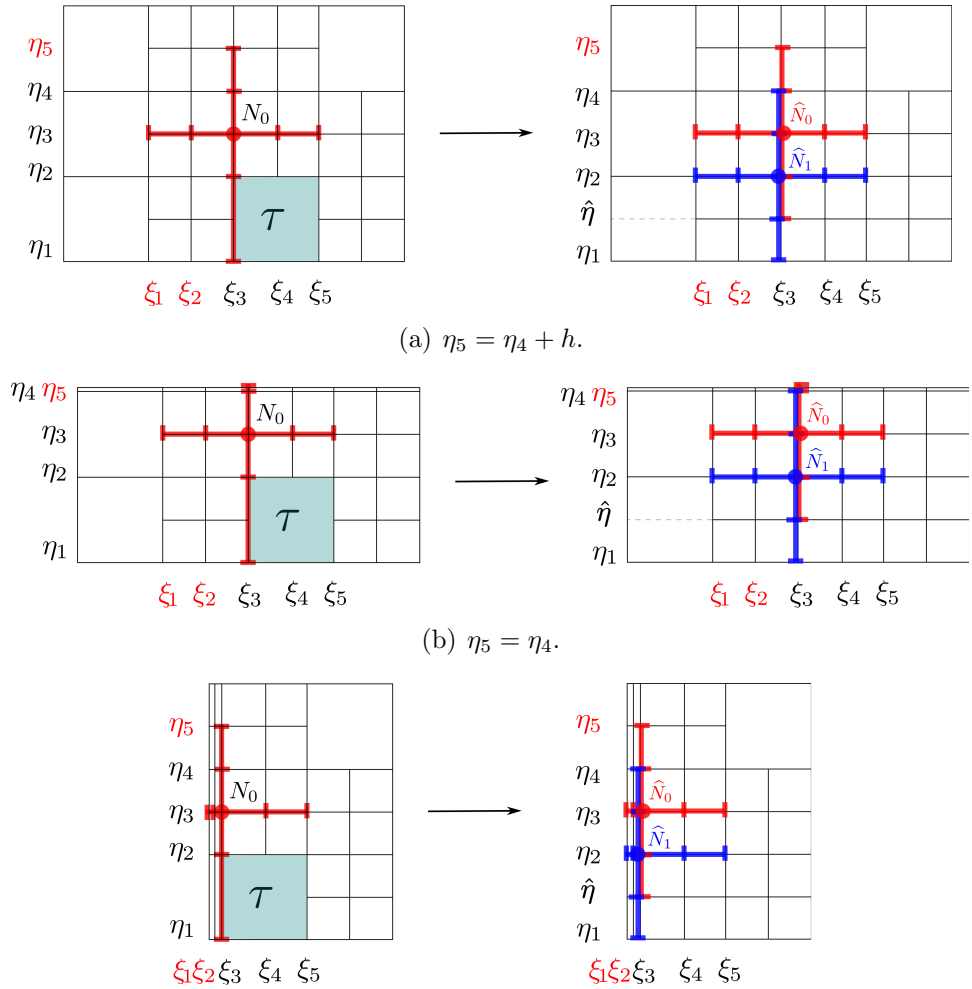

(b) $N_0 = \frac{1}{4}\widehat{N}_1 + \widehat{N}_0$.

**Figure 5.A.8:** *Type 1 function $N_0$ is always split into two blending functions from the new space.*

where $\mathcal{H}_1 = (\eta_1, \hat{\eta}, \eta_2, \eta_3, \eta_4)$ and $\mathcal{H}_2 = (\hat{\eta}, \eta_2, \eta_3, \eta_4, \eta_5)$. It is easy to see that the functions $\widehat{N}_1$ and $\widehat{N}_0$ are new blending functions from $S_{\widehat{T}_2^e}$, see Fig. 5.A.8. Hence $N_0 \in S_{\widehat{T}_2^e}$.

Note that, according to the formula (5.2), the value of the knot $\eta_5$ is irrelevant for the splitting of the function $N_0$. It could be $\eta_5 = \eta_4 + 2h$ or $\eta_5 = \eta_4 + h$ or $\eta_5 = \eta_4$, see Fig. 5.A.9(a-b). Since the splitting is performed for $\eta$ direction, the values of $\xi_1$ and $\xi_2$ are also irrelevant. Another possible case, besides the considered here for the $\Xi$ knot vector, is $\xi_1 = \xi_2 = \xi_3$, see Fig. 5.A.9(c). Combination of all possible values of $\eta_5$ and $\xi_1$ gives en total $6 = 3 \times 2$ possible situations for type 1 function $N_0$. For all these cases the linear combination for the splitting of the function $N_0$ is the same and the functions $\widehat{N}_1$, $\widehat{N}_0$ are blending functions of the mesh $\widehat{T}_2^e$. From now on, to denote which knot values are irrelevant for the subject of our study, we are going to mark them with red font in the figures. We are going to mention the other possible knot values, but we are not going to study mesh configurations corresponding to these cases, since they are equivalent to the studied case.

**Type 2.** The only possible situation when this type of function can undergo some changes, after the refinement of the cell $\tau$, is the following. For the mesh $\widehat{T}_1^e$ the support of the function $N_0$ was extended due to Extension rule 2, because its corner




(a) $\eta_5 = \eta_4 + h$.

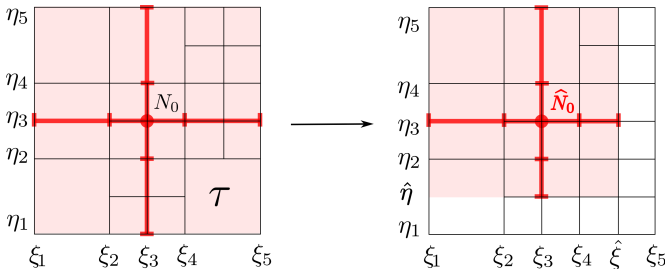
(b) $\eta_5 = \eta_4$.

(c) $\xi_1 = \xi_2 = \xi_3$.

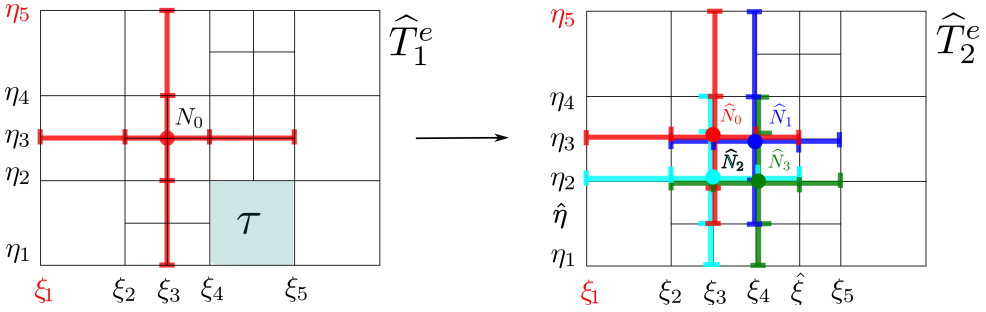
**Figure 5.A.9:** *Type 1 function $N_0$. Other possible values for the irrelevant knots of the Fig. 5.A.8(b).*

was not situated over the mesh edges. However, after the refinement of the cell $\tau$, this does not happen any more for the mesh $\widehat{T}_2^e$. Consequently the support of the function $N_0$ is reduced, because no extension is needed, see for example Fig. 5.A.10(a). Two cases are possible: (1) both knot vectors are changed or (2) only one of the knot vectors of $N_0$ is changed. For both cases only a reduction of the support (scenario 1) can take place. Let us consider each of two possible cases.

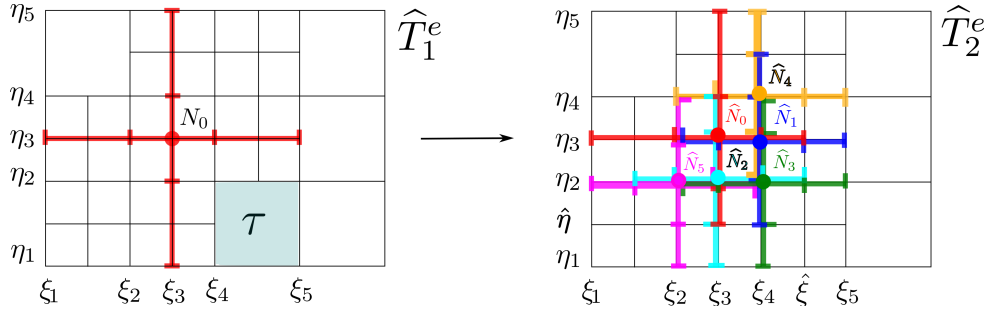
*Case 1.* Both knot vectors of the function $N_0$ are modified. Suppose that for the function $N_0 = B[\Xi]B[\mathcal{H}]$ its knot vector $\Xi = (\xi_1, \xi_2, \xi_3, \xi_4, \xi_5)$ becomes $(\xi_1, \xi_2, \xi_3, \xi_4, \hat{\xi})$ and $\mathcal{H} = (\eta_1, \eta_2, \eta_3, \eta_4, \eta_5)$ becomes $(\hat{\eta}, \eta_2, \eta_3, \eta_4, \eta_5)$ after the refinement of $\tau$. That means that a new knot $\hat{\xi}$ is inserted between the fourth and the fifth knot of $\Xi$ and a new knot $\hat{\eta}$ is inserted between the first and the second knot of $\mathcal{H}$ , see Fig.




(a) Modification of the function after the refinement.




(b) $N_0 = \widehat{N}_0 + \frac{1}{4}\widehat{N}_1 + \frac{1}{16}\widehat{N}_3 + \frac{1}{4}\widehat{N}_2$.




(c) $N_0 = \widehat{N}_0 + \frac{1}{4}\widehat{N}_1 + \frac{1}{4}\widehat{N}_2 + \frac{1}{16}\widehat{N}_3 + \frac{1}{16}\widehat{N}_4 + \frac{1}{16}\widehat{N}_5$.

**Figure 5.A.10:** *Type 2 function $N_0$, case 1: both knot vectors are modified.*

5.A.10(a). Then, in virtue of (5.5), (5.6) and (5.4), the function $N_0$ is split into four functions

$$\begin{aligned} B[\Xi,\mathcal{H}] &= (B[\Xi_1] + \frac{1}{4}B[\Xi_2])(\frac{1}{4}B[\mathcal{H}_1] + B[\mathcal{H}_2]) = \\ &= \frac{1}{4}B[\Xi_1]B[\mathcal{H}_1] + B[\Xi_1]B[\mathcal{H}_2] + \frac{1}{16}B[\Xi_2]B[\mathcal{H}_1] + \frac{1}{4}B[\Xi_2]B[\mathcal{H}_2] = \\ &= \frac{1}{4}\bar{N}_2 + \widehat{N}_0 + \frac{1}{16}\widehat{N}_3 + \frac{1}{4}\bar{N}_1. \end{aligned} \tag{5.10}$$

where $\Xi_1 = (\xi_1, \xi_2, \xi_3, \xi_4, \hat{\xi})$, $\Xi_2 = (\xi_2, \xi_3, \xi_4, \hat{\xi}, \xi_5)$, $\mathcal{H}_1 = (\eta_1, \hat{\eta}, \eta_2, \eta_3, \eta_4)$ and $\mathcal{H}_2 =$

$(\hat{\eta}, \eta_2, \eta_3, \eta_4, \eta_5)$. Now, to verify that $N_0 \in S_{\widehat{T}_2^e}$, we have to check that each of the four functions of (5.10) belongs to the new space $S_{\widehat{T}_2^e}$. The functions $\widehat{N}_0$ and $\widehat{N}_3$ are always blending functions of $\widehat{T}_2^e$ for any underlying mesh configuration. And the functions $\bar{N}_1$ and $\bar{N}_2$ can be directly blending functions of $\widehat{T}_2^e$ or a linear combination of those, depending on underlying mesh configuration. Namely, two different situations are possible.

- For the mesh shown in Fig. 5.A.10(b), the functions $\bar{N}_1$ and $\bar{N}_2$ are directly blending functions of $\widehat{T}_2^e$. So, we denote them as $\widehat{N}_1 = \bar{N}_1$ and $\widehat{N}_2 = \bar{N}_2$. Thus, from (5.10) we have

  $$N_0 = \widehat{N}_0 + \frac{1}{4}\widehat{N}_1 + \frac{1}{16}\widehat{N}_3 + \frac{1}{4}\widehat{N}_2.$$

  In this case the values of the knots $\eta_5$ and $\xi_1$ are irrelevant. Equivalent cases take place for other mesh configurations where $\eta_5 = \eta_4 + h$ or $\eta_5 = \eta_4$, and $\xi_1 = \xi_2 - h$ or $\xi_1 = \xi_2$. The linear combination for $N_0$ would be the same.

- For the mesh shown in Fig. 5.A.10(c), the functions $\bar{N}_1$ and $\bar{N}_2$ are linear combinations of the blending functions of $\widehat{T}_2^e$, namely $\bar{N}_1 = \frac{1}{4}\widehat{N}_4 + \widehat{N}_1$ and $\bar{N}_2 = \frac{1}{4}\widehat{N}_5 + \widehat{N}_2$. Therefore, from (5.10) follows

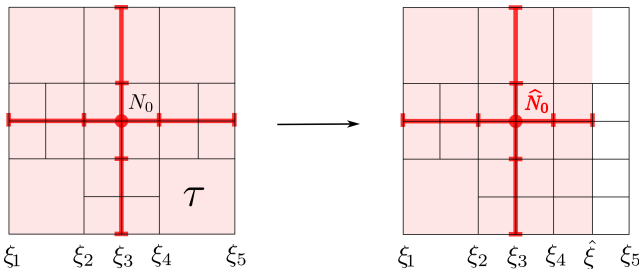
  $$N_0 = \widehat{N}_0 + \frac{1}{4}\widehat{N}_1 + \frac{1}{4}\widehat{N}_2 + \frac{1}{16}\widehat{N}_3 + \frac{1}{16}\widehat{N}_4 + \frac{1}{16}\widehat{N}_5,$$

  where
  $\widehat{N}_1 = B[\Xi_2]B[(\hat{\eta}, \eta_2, \eta_3, \eta_4, (\eta_4 + \eta_5)/2)]$,
  $\widehat{N}_4 = B[\Xi_2]B[(\eta_2, \eta_3, \eta_4, (\eta_4 + \eta_5)/2, \eta_5)]$,
  $\widehat{N}_2 = B[((\xi_1 + \xi_2)/2, \xi_2, \xi_3, \xi_4, \hat{\xi})]B[\mathcal{H}_1]$,
  $\widehat{N}_5 = B[(\xi_1, (\xi_1 + \xi_2)/2, \xi_2, \xi_3, \xi_4)]B[\mathcal{H}_1]$.

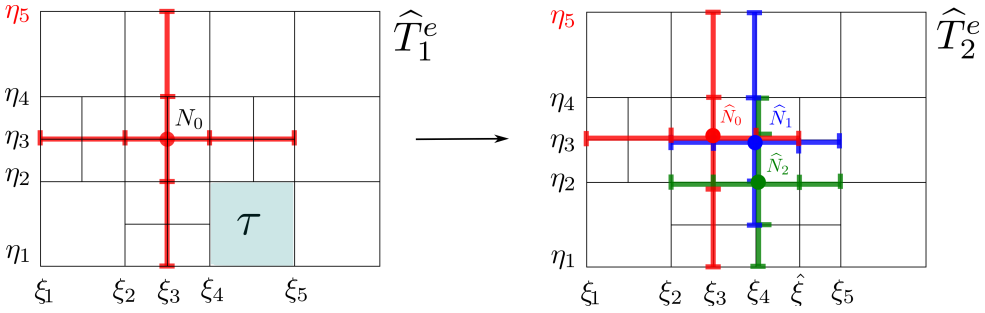
*Case 2.* Only one knot vector is changed. Suppose that for the function $N_0 = B[\Xi]B[\mathcal{H}]$ its knot vector $\Xi = (\xi_1, \xi_2, \xi_3, \xi_4, \xi_5)$ becomes $(\xi_1, \xi_2, \xi_3, \xi_4, \hat{\xi})$ after the refinement of $\tau$. That means that a new knot $\hat{\xi}$ is inserted between the fourth and the fifth knot of $\Xi$, see Fig. 5.A.11(a). Then, in virtue of (5.3) and (5.5), the function $N_0$ is split into the two functions

$$N_0 = B[\Xi]B[\mathcal{H}] = B[\Xi_1]B[\mathcal{H}] + \frac{1}{4}B[\Xi_2]B[\mathcal{H}] = \widehat{N}_0 + \frac{1}{4}\bar{N}_1,$$

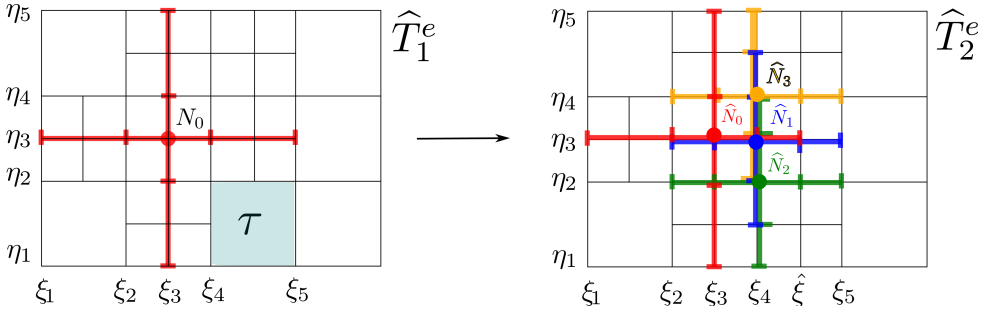
where $\Xi_1 = (\xi_1, \xi_2, \xi_3, \xi_4, \hat{\xi})$ and $\Xi_2 = (\xi_2, \xi_3, \xi_4, \hat{\xi}, \xi_5)$. The function $\widehat{N}_0$ is a blending function of $\widehat{T}_2^e$, and the other function $\bar{N}_1$ can be expressed as linear combination of blending functions of $\widehat{T}_2^e$. Two situations are possible.




(a) Modification of the function after the refinement.




(b) $N_0 = \widehat{N}_0 + \frac{1}{16}\widehat{N}_2 + \frac{1}{4}\widehat{N}_1$.




(c) $N_0 = \widehat{N}_0 + \frac{1}{16}\widehat{N}_2 + \frac{1}{4}\widehat{N}_1 + \frac{1}{16}\widehat{N}_3$.

**Figure 5.A.11:** *Type 2 function $N_0$, case 2: only one knot vector is changed.*

- For the mesh shown in Fig. 5.A.11(b), the function $\bar{N}_1$ is a linear combination of two blending functions of $\widehat{T}_2^e$, i.e., $\bar{N}_1 = \frac{1}{4}\widehat{N}_2 + \widehat{N}_1$, and, therefore, we have

$$N_0 = \widehat{N}_0 + \frac{1}{4}\left(\frac{1}{4}\widehat{N}_2 + \widehat{N}_1\right) = \widehat{N}_0 + \frac{1}{16}\widehat{N}_2 + \frac{1}{4}\widehat{N}_1,$$

where
$\widehat{N}_0 = B[\Xi_1]B[\mathcal{H}]$,
$\widehat{N}_2 = B[\Xi_2]B[\eta_1, (\eta_1+\eta_2)/2, \eta_2, \eta_3, \eta_4]$ and

$\widehat{N}_1 = B[\Xi_2]B[(\eta_1+\eta_2)/2, \eta_2, \eta_3, \eta_4, \eta_5]$, see Fig. 5.A.11(b).
In this case the value of the knot $\eta_5$ is irrelevant. Other possible values are $\eta_5 = \eta_4 + h$ or $\eta_5 = \eta_4$.

- For the mesh configuration shown in Fig. 5.A.11(c), the function $\bar{N}_1$ is a linear combination of three blending functions of $\widehat{T}_2^e$; and the linear combination for the function $N_0$ is as follows

$$N_0 = \widehat{N}_0 + \frac{1}{16}\widehat{N}_2 + \frac{1}{4}\widehat{N}_1 + \frac{1}{16}\widehat{N}_3,$$

where
$\widehat{N}_2 = B[\Xi_2]B[\eta_1, (\eta_1+\eta_2)/2, \eta_2, \eta_3, \eta_4]$,
$\widehat{N}_1 = B[\Xi_2]B[(\eta_1+\eta_2)/2, \eta_2, \eta_3, \eta_4, (\eta_4+\eta_5)/2]$,
$\widehat{N}_3 = B[\Xi_2]B[\eta_3, \eta_4, (\eta_4+\eta_5)/2, \eta_5]$, see Fig. 5.A.11(c).

**Type 3.** For this type only one of the knot vectors of $N_0$ can be modified after the refinement of the cell $\tau$. However, several different situations can arise: scenario 1 (the support is reduced after the refinement) and scenario 2 (the support is extended). Let us consider each of the two possible scenarios.

*Scenario 1.* One of the knot vectors of the function $N_0$ is reduced after the refinement. Suppose that for the function $N_0 = B[\Xi]B[\mathcal{H}]$ its knot vector $\mathcal{H} = (\eta_1, \eta_2, \eta_3, \eta_4, \eta_5)$ becomes $(\hat{\eta}, \eta_2, \eta_3, \eta_4, \eta_5)$ after the refinement of $\tau$, see Fig. 5.A.12(a). That means that a new knot $\hat{\eta}$ is inserted between the first and the second knot of $\mathcal{H}$, see Fig. 5.A.12(a). Then, in virtue of (5.3) and (5.6), the function $N_0$ is split into two functions

$$N_0 = B[\Xi]B[\mathcal{H}] = \frac{1}{4}B[\Xi]B[\mathcal{H}_1] + B[\Xi]B[\mathcal{H}_2] = \frac{1}{4}\bar{N}_1 + \widehat{N}_0,$$

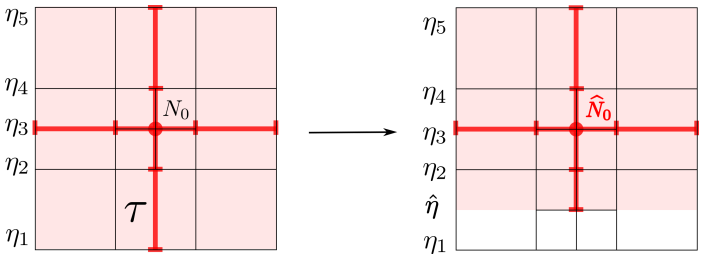
where $\mathcal{H}_1 = (\eta_1, \hat{\eta}, \eta_2, \eta_3, \eta_4)$ and $\mathcal{H}_2 = (\hat{\eta}, \eta_2, \eta_3, \eta_4, \eta_5)$.

The function $\widehat{N}_0$ is a blending function of $\widehat{T}_2^e$ for any mesh configuration. And, analogously to anterior cases, the function $\bar{N}_1$ can be a blending function of $\widehat{T}_2^e$ or can be expressed as linear combination of those. Three different situations are possible.

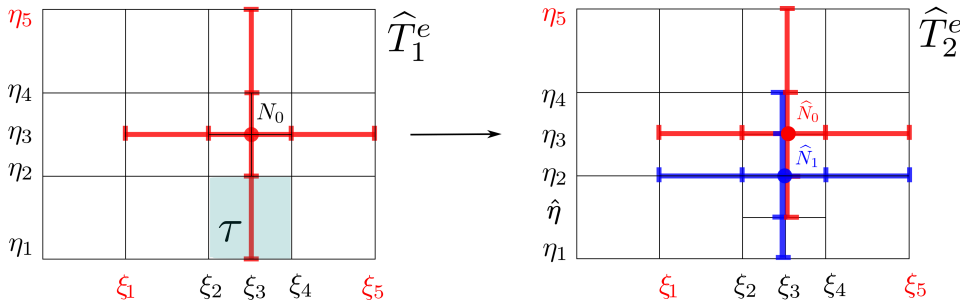
- For the mesh shown in Fig. 5.A.12(b) the function $\bar{N}_1$ is a blending function of $\widehat{T}_2^e$. So, $\widehat{N}_1 = \bar{N}_1$, and we have

$$N_0 = \widehat{N}_0 + \frac{1}{4}\widehat{N}_1.$$

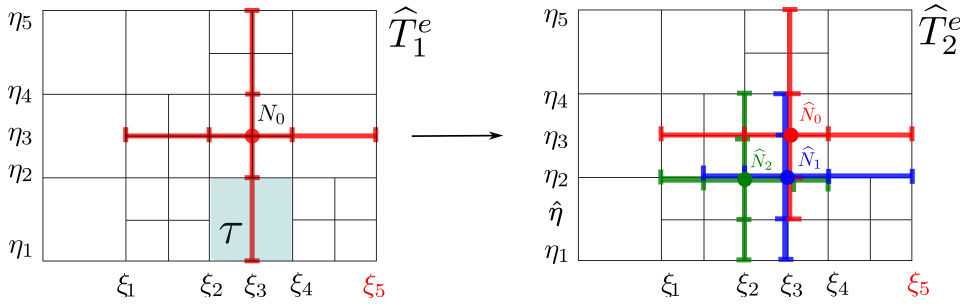
For this case the values of the knots $\xi_1$, $\xi_5$ and $\eta_5$ are irrelevant. Other possible values are $\eta_5 = \eta_4 + h$ or $\eta_5 = \eta_4$, and $\xi_1 = \xi_2 - h$ or $\xi_1 = \xi_2$, and $\xi_5 = \xi_4 + h$ or $\xi_5 = \xi_4$.



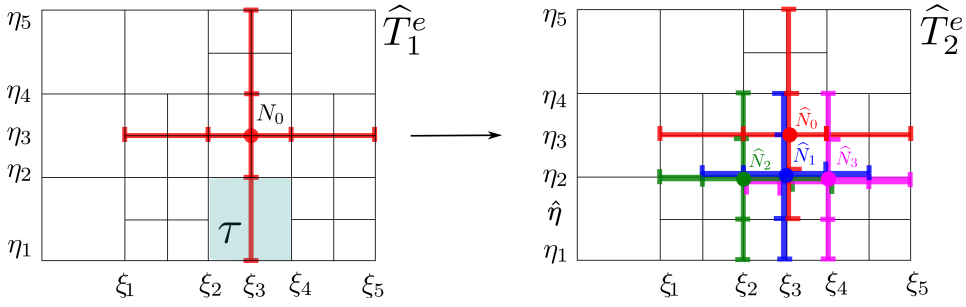

(a) Modification of the function after the refinement.




(b) $N_0 = \widehat{N}_0 + \frac{1}{4}\widehat{N}_1$.




(c) $N_0 = \widehat{N}_0 + \frac{1}{4}\widehat{N}_1 + \frac{1}{16}\widehat{N}_2$,




(d) $N_0 = \widehat{N}_0 + \frac{1}{4}\widehat{N}_1 + \frac{1}{16}\widehat{N}_2 + \frac{1}{16}\widehat{N}_3$,

**Figure 5.A.12:** *Type 3 function $N_0$, scenario 1. The knot vector $\mathcal{H}$ is reduced after the refinement.*

- For the mesh shown in Fig. 5.A.12(c), the function $\bar{N}_1$ is a linear combination of two blending functions of $\widehat{T}_2^e$: $\bar{N}_1 = \frac{1}{4}\widehat{N}_2 + \widehat{N}_1$. Then

$$N_0 = \widehat{N}_0 + \frac{1}{4}\widehat{N}_1 + \frac{1}{16}\widehat{N}_2,$$

  where
  $\widehat{N}_1 = B[((\xi_1+\xi_2)/2, \xi_2, \xi_3, \xi_4, \xi_5)]B[\mathcal{H}_1]$,
  $\widehat{N}_2 = B[(\xi_1, (\xi_1+\xi_2)/2, \xi_2, \xi_3, \xi_4)]B[\mathcal{H}_1]$.

- For the mesh shown in Fig. 5.A.12(d), the function $\bar{N}_1$ is a linear combination of three blending functions of $\widehat{T}_2^e$, namely $\bar{N}_1 = \frac{1}{4}\widehat{N}_2 + \widehat{N}_1 + \frac{1}{4}\widehat{N}_3$, then

$$N_0 = \widehat{N}_0 + \frac{1}{4}\widehat{N}_1 + \frac{1}{16}\widehat{N}_2 + \frac{1}{16}\widehat{N}_3,$$

  where
  $\widehat{N}_1 = B[((\xi_1+\xi_2)/2, \xi_2, \xi_3, \xi_4, \xi_5)]B[\mathcal{H}_1]$,
  $\widehat{N}_2 = B[(\xi_1, (\xi_1+\xi_2)/2, \xi_2, \xi_3, \xi_4)]B[\mathcal{H}_1]$
  $\widehat{N}_3 = B[(\xi_2, \xi_3, \xi_4, (\xi_4+\xi_5)/2, \xi_5)]B[\mathcal{H}_1]$.

*Scenario 2.* One of knot vectors of the function $N_0$ is extended after the refinement. Suppose that for the function $N_0 = B[\Xi]B[\mathcal{H}]$ its knot vector $\Xi = (\xi_1, \xi_2, \xi_3, \xi_4, \xi_5)$ becomes $(\hat{\xi}, \xi_2, \xi_3, \xi_4, \xi_5)$ after the refinement of $\tau$, see Fig. 5.A.13(a). Then, the function $N_0$ can be expressed as linear combination of two functions

$$N_0 = B[\Xi]B[\mathcal{H}] = \frac{1}{4}B[\Xi_1]B[\mathcal{H}] + B[\Xi_2]B[\mathcal{H}] = -\frac{1}{4}\bar{N}_1 + \widehat{N}_0,$$

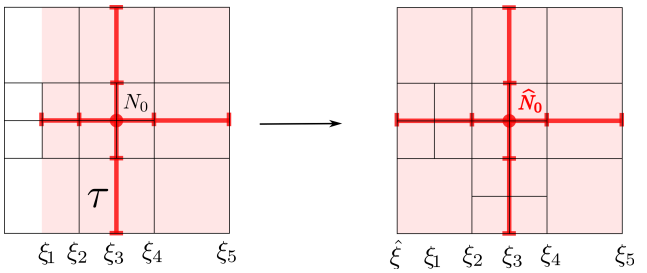
where $\Xi_1 = (\hat{\xi}, \xi_1, \xi_2, \xi_3, \xi_4)$ and $\Xi_2 = (\hat{\xi}, \xi_2, \xi_3, \xi_4, \xi_5)$.

The function $\widehat{N}_0$ is a blending functions of $\widehat{T}_2^e$ for any mesh configuration. And, analogously to anterior cases, the function $\bar{N}_1$ can be a blending function of $\widehat{T}_2^e$ or can be expressed as linear combination of those. Two different situations are possible.

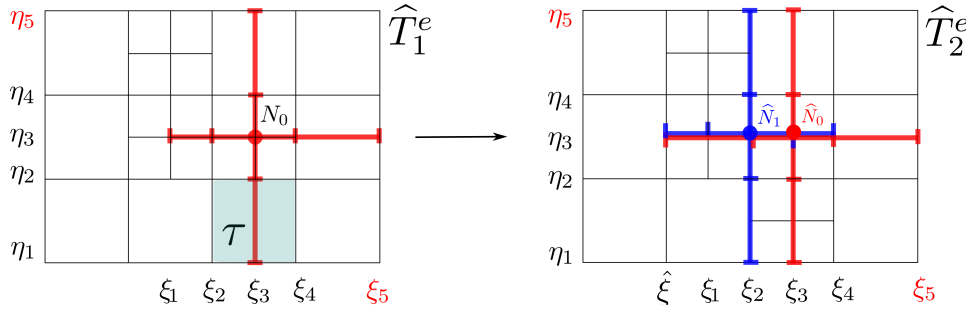
- For the mesh shown in Fig. 5.A.13(b), the function $\bar{N}_1$ is a blending function of $\widehat{T}_2^e$. So, we have

$$N_0 = \widehat{N}_0 - \frac{1}{4}\widehat{N}_1.$$

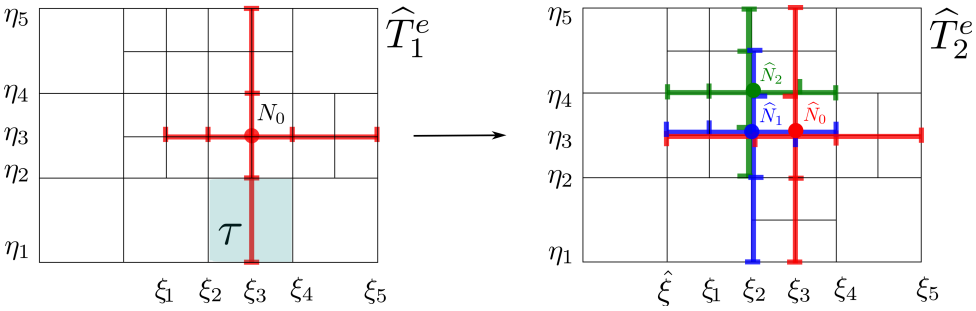
  For this case the values of the knots $\xi_5$ and $\eta_5$ are irrelevant. Several equivalent cases take place for other mesh configurations where $\eta_5 = \eta_4 + h$ or $\eta_5 = \eta_4$, and $\xi_5 = \xi_4 + h$ or $\xi_5 = \xi_4$.




(a) Modification of the function after the refinement.




(b) $N_0 = \widehat{N}_0 - \frac{1}{4}\widehat{N}_1$.



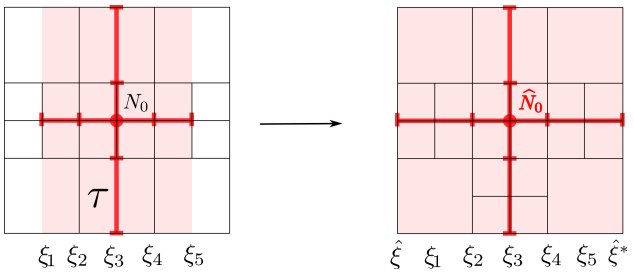

(c) $N_0 = \widehat{N}_0 - \frac{1}{4}\widehat{N}_1 - \frac{1}{16}\widehat{N}_2$,

**Figure 5.A.13:** *Type 3 function $N_0$, scenario 2. The knot vector $\Xi$ is extended in one direction after the refinement.*

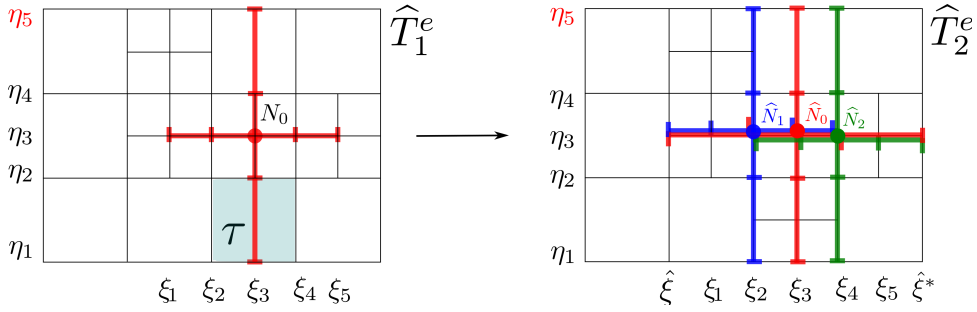
- For the mesh shown in Fig. 5.A.13(c), the function $\bar{N}_1$ is a linear combination of two blending functions of $\widehat{T}_2^e$ : $\bar{N}_1 = \frac{1}{4}\widehat{N}_2 + \widehat{N}_1$, then

$$N_0 = \widehat{N}_0 - \frac{1}{4}\widehat{N}_1 - \frac{1}{16}\widehat{N}_2,$$

where
$\widehat{N}_1 = B[\Xi_1]B[(\eta_1, \eta_2, \eta_3, \eta_4, (\eta_4 + \eta_5)/2)]$,
$\widehat{N}_2 = B[\Xi_1]B[(\eta_2, \eta_3, \eta_4, (\eta_4 + \eta_5)/2, \eta_5)]$.




(a) Modification of the function after the refinement.




(b) $N_0 = \widehat{N}_0 - \frac{1}{4}\widehat{N}_1 - \frac{1}{4}\widehat{N}_2$

**Figure 5.A.14:** *Type 3 function $N_0$, scenario 2. The knot vector $\Xi$ is extended in two directions after the refinement.*

Another possible extension of the support $N_0$ is for both sides of its knot vector $\Xi$, i.e., $\Xi = (\xi_1, \xi_2, \xi_3, \xi_4, \xi_5)$ becomes $(\hat{\xi}, \xi_2, \xi_3, \xi_4, \xi^*)$ after the refinement of $\tau$, see Fig. 5.A.14(a). Then, the function $N_0$ can be expressed as linear combination of three functions

$$N_0 = \widehat{N}_0 - \frac{1}{4}\widehat{N}_1 - \frac{1}{4}\widehat{N}_2,$$

where
$\widehat{N}_0 = B[(\hat{\xi}, \xi_2, \xi_3, \xi_4, \xi^*)]B[\mathcal{H}]$,
$\widehat{N}_1 = B[(\hat{\xi}, \xi_1, \xi_2, \xi_3, \xi_4)]B[\mathcal{H}]$,
$\widehat{N}_2 = B[(\xi_2, \xi_3, \xi_4, \xi^*)]B[\mathcal{H}]$, see Fig. 5.A.14(b).
For this case the value of the knot $\eta_5$ is irrelevant. It could be $\eta_5 = \eta_4 + h$ or $\eta_5 = \eta_4$.

**Remark 13.** *Any other mesh configuration for each type of functions near the cell $\tau$, not analysed here, is equivalent to one of the studied cases or it does not lead to any change of the function support under consideration. All these cases can be generated by giving to the irrelevant knots other possible values mentioned for each case.*

□

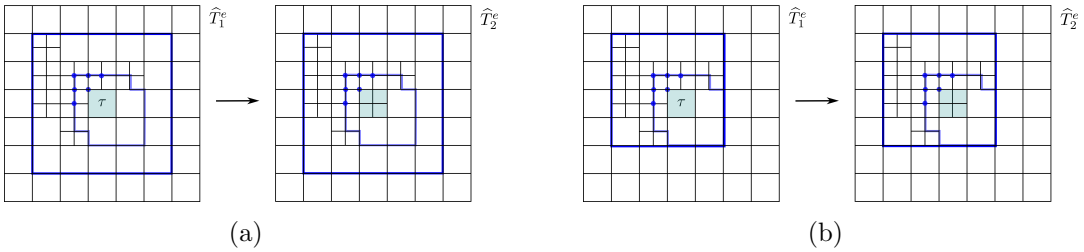



**Figure 5.A.15:** *Computational experiment to validate Proposition 1. (a) The area to study via computational experiment:* $25 = 5 \times 5$ *cells. (b) Reduced area to study, discarding symmetric function supports.*

### 5.A.2.2 Computational experiment to validate Proposition 1

Also, we have verified the statement of Proposition 1 by means of the following computational experiment. We have to show that for any simple elemental refinement $\widehat{T}_1^e \subset \widehat{T}_2^e$ all function supports of the one-ring of the cell $\tau$ can be recovered by means of the functions of the new space $S_{\widehat{T}_2^e}$.

For this, it is sufficient to analyse all possible mesh configurations in certain neighbourhood of the cell $\tau$. Namely, we can study a zone of neighbouring cells around the cell $\tau$, which includes the area occupied by $25 = 5 \times 5$ cells of level *(m-1)*, see Fig. 5.A.15(a). It is important to highlight that any function support (marked by blue circles), that can be affected by the refinement, are contained entirely in this area. Note that, discarding the symmetric function supports, we can reduce the study to the quadrant composed of $16 = 4 \times 4$ cells, see Fig. 5.A.15(b). We carry out the following simple computational experiment. We generate all possible $2^{15}$ mesh configurations (16 cells except the cell $\tau$ gives 15 cells) and verify for each mesh $\widehat{T}_1^e$ that all its functions of the one-ring neighbourhood can be expressed as linear combination of the blending functions of $\widehat{T}_2^e$. For that, we interpolate each blending function $N_0 \in S_{\widehat{T}_1^e}$ using the approximation space of the refined mesh $S_{\widehat{T}_2^e}$ and check if the interpolation is able to reproduce exactly the function under consideration (in the sense that the interpolation error $\|u - u_h\|_{L^1}$ is zero.) The experiment has confirmed that all functions of one-ring neighbourhood can be recovered with the new space. Thus, the experiment validates Proposition 1.

# CHAPTER 6

## Parameterization method for complex 2D and 3D geometries from representation of its boundary

In this chapter we expose briefly another result of our research: a method for spline parameterization of complex 2D and 3D geometries. CAD models usually provide only the boundary surface of a solid. But the application of Isogeometric Analysis requires a full volumetric representation of the geometry. "Surface-to volume parameterization" is still an open problem. For application of IGA it is essential to have an effective method to obtain volumetric parameterization of the geometry from the representation of its boundary. This problem is studied in depth and will be presented in another doctoral dissertation of the group. Here we include briefly some preliminary results about our parameterization method, which will be necessary for better understanding of some computational examples of the next chapter.

## 6.1 Meccano method. Volumetric parameterization of a solid

At the moment we have a method to obtain volumetric spline parameterization of a solid. The approach was developed previously in the group and described in [58, 59]. It is based on the idea of *Meccano* method [60] for automatic tetrahedral mesh generation, which, at the same time, obtains a one-to-one correspondence between the tetrahedral meshes of the parametric domain $\widehat{\Omega} = [0,1]^3$ and the physical domain $\Omega$, see Fig. 6.1. This correspondence is obtained by deforming isomorphically the adapted tetrahedral mesh of the parametric domain into the physical mesh of the solid. The deformation is performed by applying a tetrahedral mesh untangling and smoothing procedure, see [61]. The deformation only affects the positions of the nodes, that is, there is not any change in their connectivity: we say that both meshes are isomorphic. Given that any point is fully determined by the barycentric coordinates relative to the tetrahedron in which it is contained, we can define a one-to-one mapping between parametric tetrahedral mesh and the

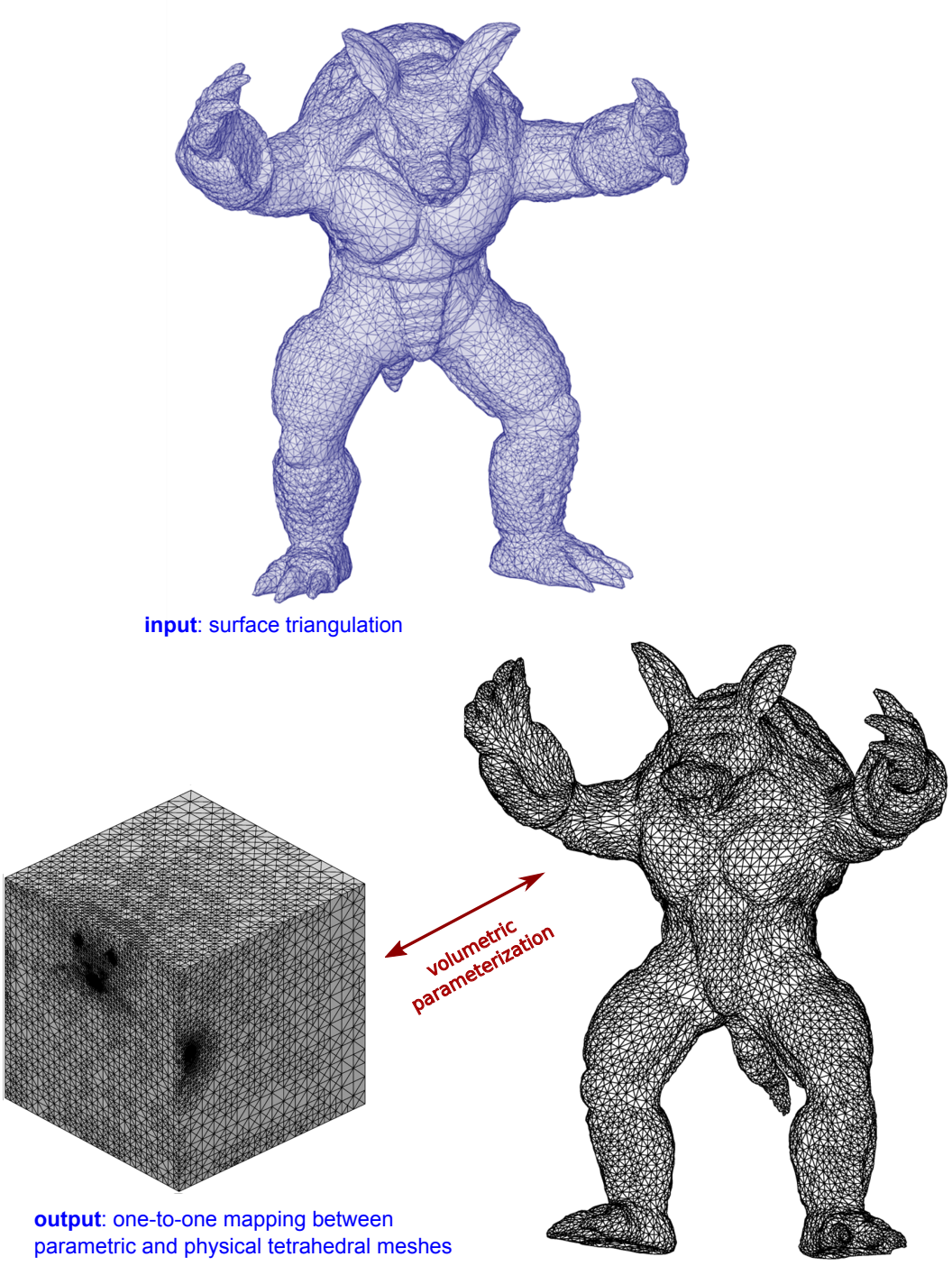


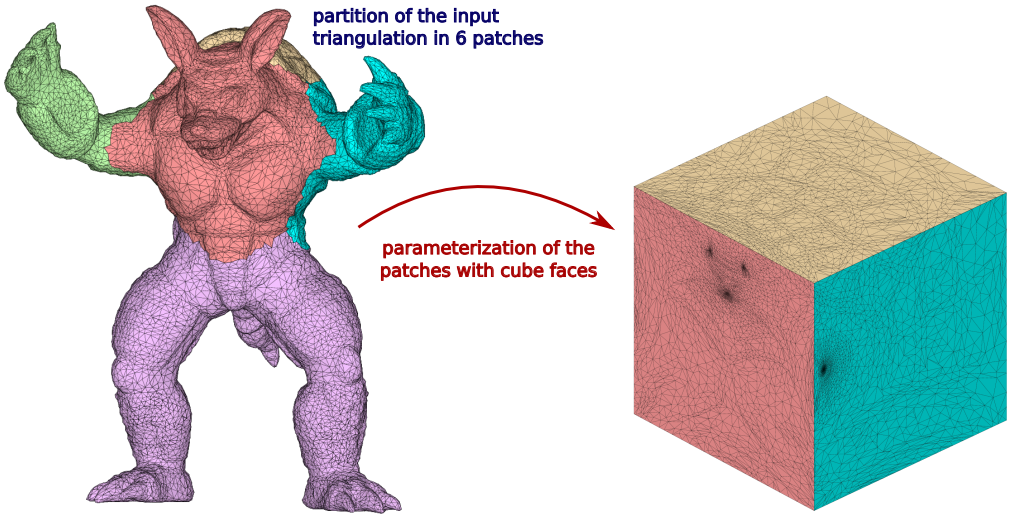

**Figure 6.1:** *Meccano method for automatic adaptive tetrahedral mesh generation.*

physical mesh of the solid, assuming that the barycentric coordinates are the same in both spaces. Thus, we obtain a volumetric parameterization of the solid. Figure 6.2 illustrates the main steps of Meccano method. Input triangulation of a solid is

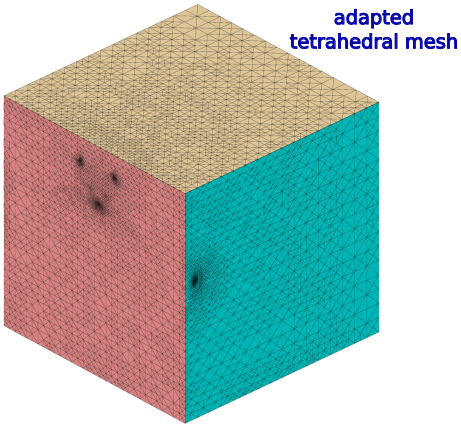
divided into six patches, see Fig. 6.2(a), that are mapped onto the cube faces via Floater parameterization [62]. Then, an adapted tetrahedral mesh of the cube is constructed using Kossaczký refinement until the geometry is approximated with a prescribed tolerance, see Fig. 6.2(b). As the result of this stage, the position of the boundary nodes in the physical domain are known and the position of the inner nodes will be defined by means of tetrahedral mesh optimization procedure. Figure 6.2(c) shows the adapted parametric mesh, the tangled physical mesh obtained after mapping the cube boundary to the *Armadillo* surface and the final optimized mesh. Optimization procedure pursues that each tetrahedron $T$ of the physical mesh is as similar as possible to its counterpart (target) tetrahedron $T_t$ in the parametric space.

Meccano method was originally designed for automatic tetrahedral mesh generation, however, at the same time it provides directly a volumetric parameterization of the solid, which can be used for construction of spline parametrization of the object. For that, a parametric T-mesh of $\widehat{\Omega}$, with a similar resolution that the tetrahedral mesh, is constructed. Trivariate spline mapping $\mathbf{S} : \hat{\Omega} \rightarrow \Omega$ is constructed by imposing interpolation conditions, where interpolation points are located in physical space via tetrahedral parameterization provided by Meccano method. Figure 6.3 illustrates the final result of the idea.

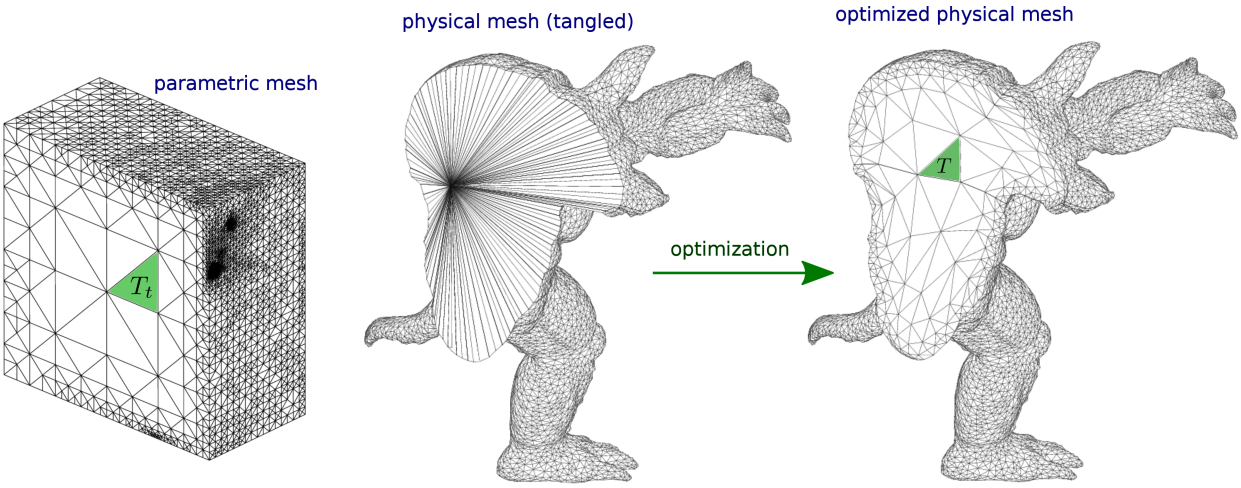
However, the procedure described below is able to obtain an acceptable spline parameterization quality only for slightly distorted geometries. Meccano method achieves a good quality tetrahedral mesh, but it may not be sufficient for constructing high quality trivariate spline mapping, since tetrahedral mesh does not take into account the necessities of tensor product structure of spline mapping. High quality spline mapping implies a good orthogonality and uniformity of isoparametric curves, which is not achieved using the tetrahedral volumetric parameterization. This has motivated the necessity to extend the method and develop an optimization procedure directly for T-mesh, instead of tetrahedral mesh. This approach allows to obtain a higher quality spline parameterization with good orthogonality and uniformity of isoparametric curves. As a first step, this idea was accomplished for 2D geometries and it is exposed briefly in next section.




(a) Surface paremeterization of the solid.




(b) Adapted tetrahedral mesh of the cube (Kossaczký refinement).




(c) Optimization of tetrahedral physical mesh.

**Figure 6.2:** *Main steps of the Meccano method.*

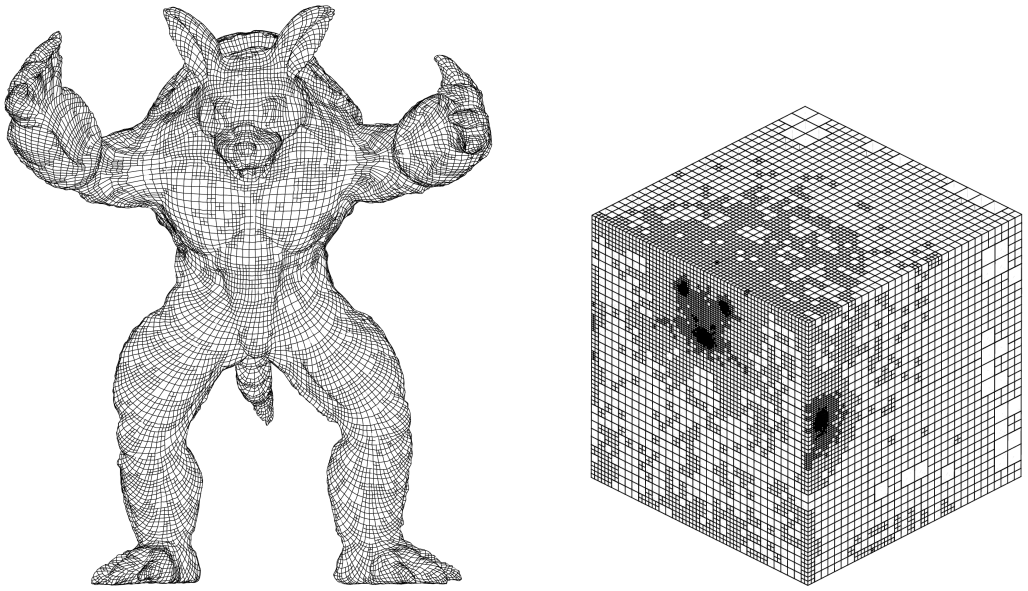


**Figure 6.3:** *T-spline representation of the solid obtained with Meccano method.*

## 6.2 Parameterization method for 2D geometries

Method for spline parameterization of 2D geometries is based on the idea of the Meccano method and a novel T-mesh optimization procedure. The proposed method only demands a boundary representation of the geometry as input data. The algorithm obtains, as a result, high quality parametric transformation between 2D objects and the parametric domain, the unit square. First, we define a parametric mapping between the input boundary of the object and the boundary of the parametric domain. Then, we build a T-mesh adapted to the geometric singularities of the domain in order to preserve the features of the object boundary with a desired tolerance. The key of the method lies in defining an isomorphic transformation between the parametric and physical T-mesh finding the optimal position of the interior nodes by applying a new T-mesh untangling and smoothing procedure. Bivariate spline representation is calculated by imposing the interpolation conditions on points sited both on the interior and on the boundary of the geometry. Next, we revise briefly the proposed method. A full version can be found in [63].

### 6.2.1 General scheme of the method

The algorithm includes the following stages:

1. *Boundary parameterization and construction of an adapted T-mesh:* A bijective correspondence between the input boundary of the object and the boundary of the parametric domain is defined. Then, an adapted T-mesh is

generated by refining the initial mesh in order to approximate the geometry with a prescribed tolerance. During this process, the boundary nodes of the parametric domain are mapped to the boundary of the object.

2. *T-Mesh optimization:* We relocate the inner nodes of the T-mesh by applying a mesh untangling and smoothing procedure.

3. *Construction of a T-spline representation of the geometry:* The T-spline mapping is obtained by imposing interpolation conditions. As interpolation points, we take the vertices of the physical T-mesh obtained after the optimization process.

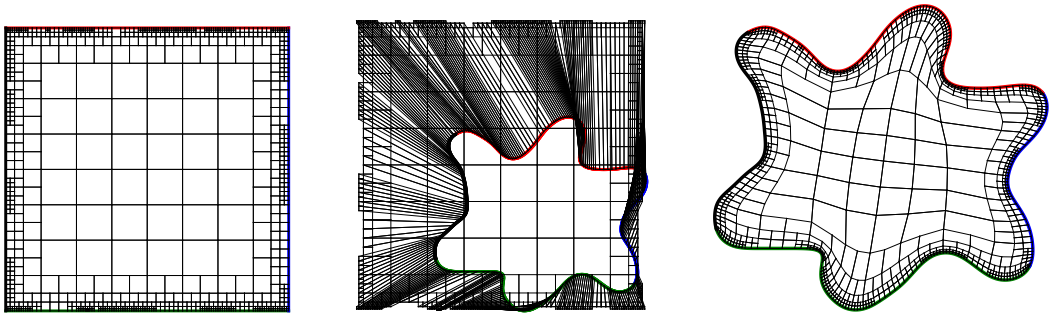
4. *Adaptive refinement to improve the mesh quality:* If the quality of the mesh is not satisfactory, we apply an adaptive refinement in order to increase the degree of freedom in the areas with high distortion. Then, we return to step 2 and repeat the process until reaching a good spline parameterization.

### 6.2.2 Boundary parameterization and construction of an adapted T-mesh

In order to define a parametric mapping between the input boundary polygonal of the object and the boundary of the parametric domain, the unit square, we have to select four points of the polygonal that will correspond to the four corners of the square. These points divide the input polygonal into four parts that are mapped via chord-length parameterization into its corresponding edge of the square. Next, we construct an adapted T-mesh that approximates the input boundary with a pre-defined tolerance $\epsilon$. To do that, an approximation error is calculated for each boundary cell and the cell is refined if this error is greater than $\epsilon$. A cell refinement produces a new boundary point that is projected over the input boundary polygonal, obtaining a more accurate approximation of the geometry. As result of this stage, the position of the boundary nodes in the physical domain are known and the position of the inner nodes will be defined by means of the T-mesh optimization procedure developed in next section. Figure 6.1(a) shows an example of the adapted parametric T-mesh constructed in this stage; the tangled mesh obtained after projecting the boundary of parametric domain to the input boundary polygonal is shown in Fig. 6.1(b); and Fig. 6.1(c) illustrates the resulting optimized T-mesh.

### 6.2.3 T-mesh optimization

The key of the proposed parameterization method lies in the optimization procedure that allows to obtain a high quality physical T-mesh, which is used to construct the T-spline representation of the object.



(a) (b) (c)

**Figure 6.1:** *Stages of T-mesh construction. (a) Parametric T-mesh adapted to the boundary of the geometry; (b) tangled physical mesh after projecting the boundary of parametric domain to the input boundary polygonal; (c) optimized physical T-mesh. The colors represent the correspondence between the parametric and physical boundaries.*

The mesh optimization process is carried out by iterative relocation of each inner node of the mesh in such a way that the new position of the node improves the quality of the local submesh corresponding to this node. A local submesh is the set of all the elements connected with the movable node which is called *free node*. The local objective function for a free node is based on algebraic shape quality metrics proposed by Knupp in [64, 65] for quadrilateral elements. Shape quality metric for a triangle is defined in terms of the Jacobian matrix of the affine mapping from the ideal triangle to the given one. This shape quality metric represents the deviation of the physical triangle from the ideal one. It attains its maximum value 1, if the triangle is similar to the ideal one, and it equals 0 if the triangle is degenerate. The distortion metric of an element is defined as the inverse of its quality metric. Shape quality metric for a quadrilateral element is based on the set of shape quality metric of its simplicial elements that compose the given quadrilateral. Therefore, in order to asses the quality of the cells, we have to decompose each cell into triangles and asses the quality of each triangle. For a T-mesh, this decomposition depends on the type of the free node. There are two types of free node: a regular node and a hanging node. A regular node is surrounded by four cells with equal or different sizes. Local submesh is decomposed in twelve triangles, three triangles per cell, whose qualities depends on the position of the free node. In a hanging node case, the free node is surrounded by three cells and the local submesh is decomposed in eleven triangles. The cell in which the node forms a T-junction is decomposed in five triangles, whose qualities depend on the position of the free node. Each one of the other two cells is decomposed in three triangles, as was described in the case of a regular node. Fig. 6.2 illustrates the decomposition of a local submesh for each case and the feasible region of the objective function. *Feasible region* is the subset of $\mathbb{R}^2$, where the free node can be placed, being the local submesh valid.

The optimal position of each free node is determined by minimizing a local

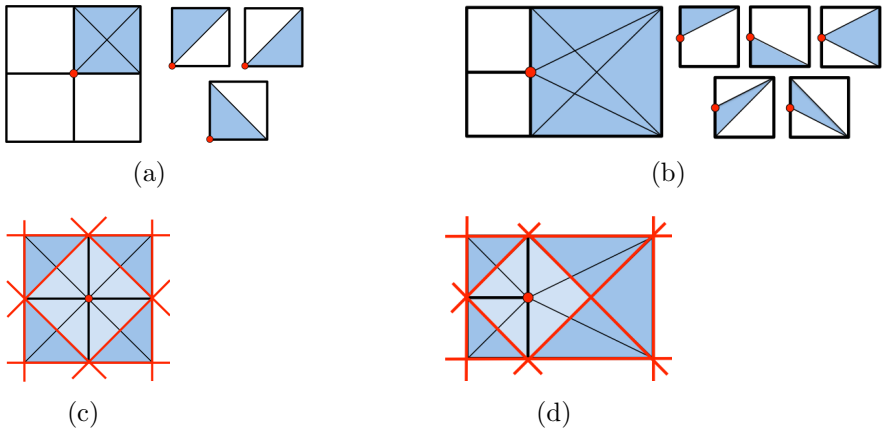



**Figure 6.2:** *Triangular decomposition of the local submesh. (a) Regular node case, where each cell is decomposed in three triangles; (b) hanging node case, where five triangles are formed in the cell where the node generates a T-juntion; (c) barriers (red lines) and feasible region (light blue) induced by the 12 triangles in the objective function for a regular node; (d) barriers and feasible region induced by the 11 triangles in the objective function for a hanging node.*

objective function. We define the objective function as a sum of shape distortion metrics of the triangles of the local submesh. For each triangle of the physical mesh, the corresponding triangle of the parametric mesh is used as its ideal element. Therefore, during optimization process, each cell of the physical mesh tends to have the same shape as its counterpart cell of the parametric mesh. Thus, repeating this procedure for all the inner nodes of the mesh, we achieve the physical mesh of the object as similar to the parametric one as possible.

Let $T$ be a triangle whose vertices are given by $\mathbf{x}_k = (x_k, y_k)^T \in \mathbb{R}^2$, $k = 0, 1, 2$. Consider that $T_I$ is our ideal or target triangle whose vertices are $\mathbf{v}_0$, $\mathbf{v}_1$ and $\mathbf{v}_2$. And let the matrix $S$ be the Jacobian matrix of the affine map that takes $T_I$ to $T$. Quality metrics of the triangle $T$ can be defined in terms of the matrix $S$. For example, the *mean ratio*, $q = \frac{2\sigma}{\|S\|^2}$, is an easily computable algebraic quality metric of $T$, where $\sigma = \det(S)$ and $\|S\|$ is the Frobenius norm of $S$. The maximum value of $q$ is the unity, and it is reached when $S = \mu R$, where $\mu$ is a scalar and $R$ is a rotation matrix. In other words, $q = 1$ if and only if $T$ and $T_I$ are similar. Besides, any flat triangle has quality measure zero. We can derive an optimization function from this quality metric. Let $\mathbf{x} = (x, y)^T$ be the position of the free node, and let $S_m$ be the Jacobian matrix of the $m$-th triangle connected to this free node. We define the distortion measure for $m$-th triangle as

$$\eta_m = \frac{\|S_m\|^2}{2\sigma_m}.$$

Then, the local objective function, used for mesh quality improvement, is defined

by means of distortion measures of each triangle of the local submesh

$$K(\mathbf{x}) = \sum_{m=1}^{M} \eta_m = \sum_{m=1}^{M} \frac{\|S_m\|^2}{2\,\sigma_m},$$

where $M$ is the number of triangles in the local submesh. The objective function $K$ becomes discontinuous when the area of any triangle goes to zero. Due to these singularities, the function $K$ improves the quality of valid elements but it does not work properly, when the mesh is tangled ($\sigma \leq 0$). In [61] we proposed a modification of $K$ by replacing $\sigma$ by the positive and increasing function $h(\sigma) = \frac{1}{2}(\sigma + \sqrt{\sigma^2 + 4\delta^2})$. Then, the modified distortion becomes

$$\eta_m^* = \frac{\|S_m\|^2}{2h\,(\sigma_m)}.$$

This modification eliminates the barriers associated with their singularities and the new objective function $K^* = \sum_{m=1}^{M} \eta_m^*$ becomes smooth in $\mathbb{R}^2$, so the unconstrained optimization problem can be easily solved with any standard method. In the feasible region the modified objective function $K^*$ approximates the original function $K$ as $\delta \to 0$, and then the minimum of the original and modified objective functions are nearly identical when $\delta$ is small. When this region does not exist, the minimum of the modified objective function is located in such a way that it tends to untangle the local mesh. Thus, the modified objective function allows the simultaneous untangling and smoothing of the mesh.

### 6.2.4 Construction of a spline representation of the geometry

We have to obtain a global one-to-one parametric transformation that maps the parametric domain into the physical domain $\mathbf{S} : \widehat{\Omega} = [0,1]^2 \to \Omega$. Spline representation of physical domain is build as lineal combination of spline blending functions defined over the adapted parametric T-mesh $T$

$$\mathbf{S}\,(\boldsymbol{\xi}) = \sum_{\alpha \in A_T} \mathbf{P}_\alpha\, N_\alpha\,(\boldsymbol{\xi})$$

where $\mathbf{P}_\alpha \in \mathbb{R}^2$ is the control point corresponding to the $\alpha$-th blending function.

Control points $\mathbf{P}_\alpha$ are found by imposing interpolation conditions. As interpolation points we use the anchors of the blending functions. Each anchor $\boldsymbol{\xi}_\alpha$ coincides with a T-mesh vertex and its position in the physical space $\mathbf{x}_\alpha$ was determined by the mesh optimization process, see Fig. 6.3. Note that for our EP-splines only regular vertices of the mesh are used for interpolation. For T-spline functions also hanging nodes (T-junctions) form a set of interpolation points.

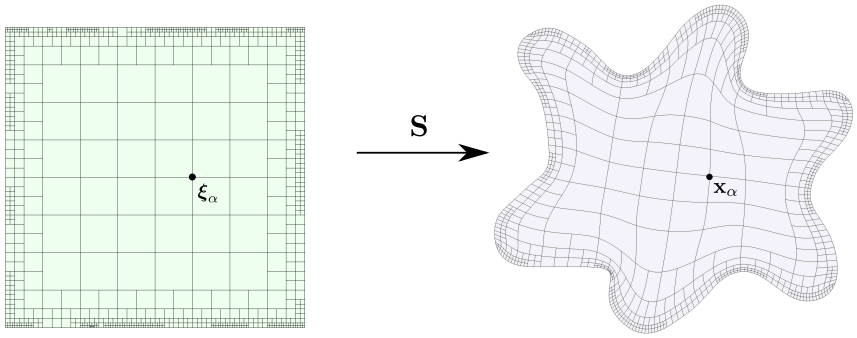



**Figure 6.3:** Spot *test model with* 844 *cells,* 1174 *control points. Parametric domain and spline representation of the physical domain.*

Finally we solve the linear system of equations

$$\mathbf{x}_\beta = \mathbf{S}\left(\boldsymbol{\xi}_\beta\right) = \sum_{\alpha \in A_T} \mathbf{P}_\alpha \, N_\alpha\left(\boldsymbol{\xi}_\beta\right), \quad \forall \boldsymbol{\xi}_\beta, \ \beta \in A_T$$

where $\boldsymbol{\xi}_\beta$ are interpolation points in parametric space and $\mathbf{x}_\beta$ are their images in the physical space.

Resulting spline parameterization of *Spot* test model is shown in Fig. 6.3.

## 6.2.5 Quality assessment and its improvement

### 6.2.5.1 Mean ratio Jacobian

Our objective is a high-quality geometry parameterization suitable for isogeometric analysis. High distortion of the geometry can produce a large variation of the Jacobian that can lead to a poor accuracy in the numerical results. A good uniformity and orthogonality of the isoparametric curves are desired for the parametric mapping $\mathbf{S}$. In order to assess the quality of the constructed parametric transformation we analyse the mean ratio Jacobian given by

$$J_r(\boldsymbol{\xi}) = \frac{2 \, \det\left(J\right)}{\|J\|^2},$$

where $J$ is the Jacobian matrix of the mapping $\mathbf{S}$ at the point $\boldsymbol{\xi} = (\xi, \eta)$ and $\|J\|$ is its Frobenius norm.

The value of the mean ratio Jacobian at any point $\mathbf{P}_0$ of the parametric domain is a shape quality metric for the infinitesimal triangle formed by two isoparametric curves of the physical domain passing through the point $\mathbf{P}'_0 = \mathbf{S}(\mathbf{P}_0)$, as illustrated in Fig. 6.4. In contrast to the scaled Jacobian, that represents a quality of the mapping $\mathbf{S}$ in the sense of the orthogonality of its isoparametric curves, the mean ratio Jacobian represents both: a quality of the mapping in the sense of the

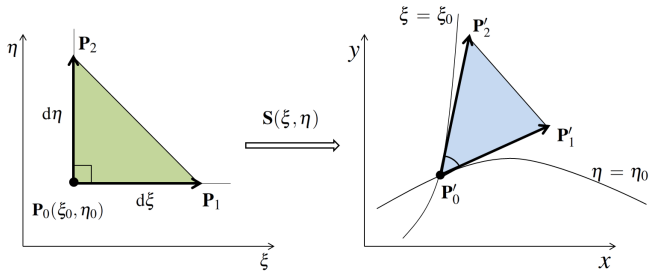
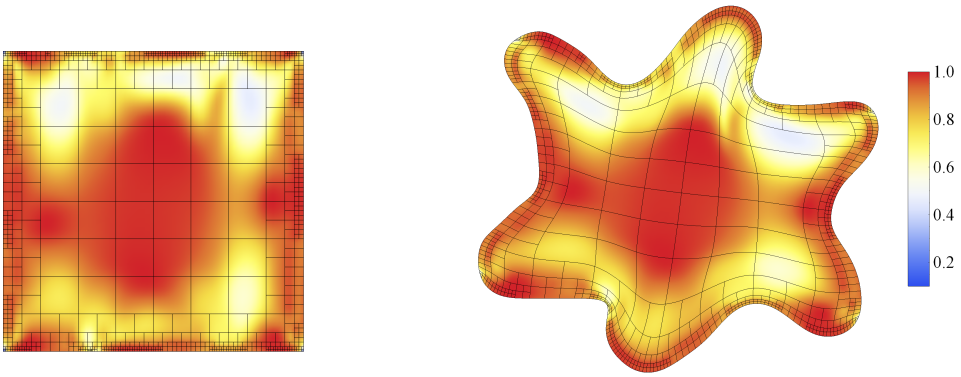



**Figure 6.4:** *Mean ratio Jacobian. A quality metric of the parametric mapping* $\mathbf{S}$ *at any point* $\mathbf{P}_0$ *in terms of the mean ratio of the triangle* $\mathbf{P}'_0\mathbf{P}'_1\mathbf{P}'_2$.

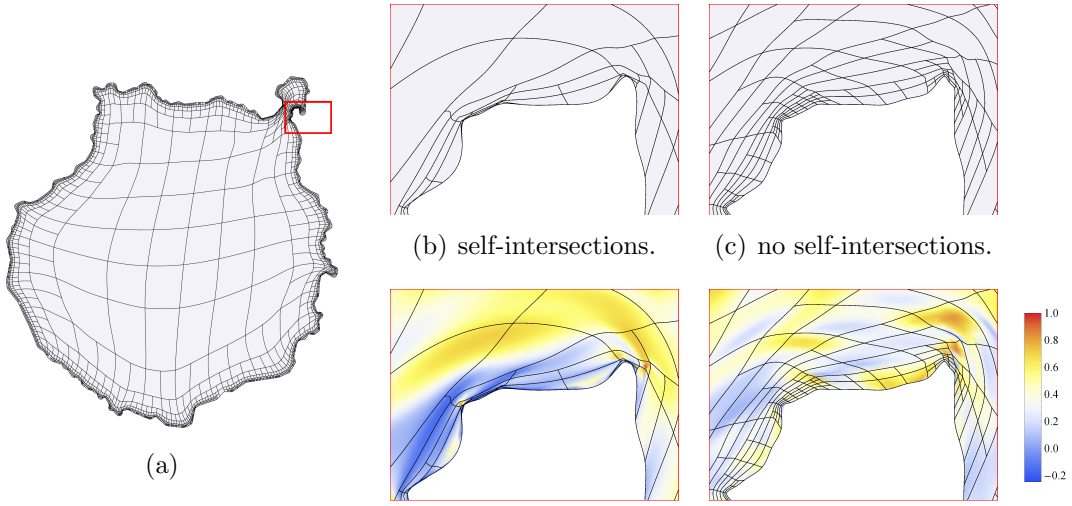
orthogonality and uniformity of its isoparametric curves. Mean ratio Jacobian is equal 1 at the point $\mathbf{P}_0$ if the mapping conserves orthogonality and produces the same length distortion in both parametric directions, i.e. the mapping is conformal at this point.

Figure 6.5 shows the resulting spline representation of the *Spot* test model and the colormap of the mean ratio Jacobian.




(a) parametric domain. (b) physical domain.

**Figure 6.5:** Spot *test model. Colormap of mean ratio Jacobian.*



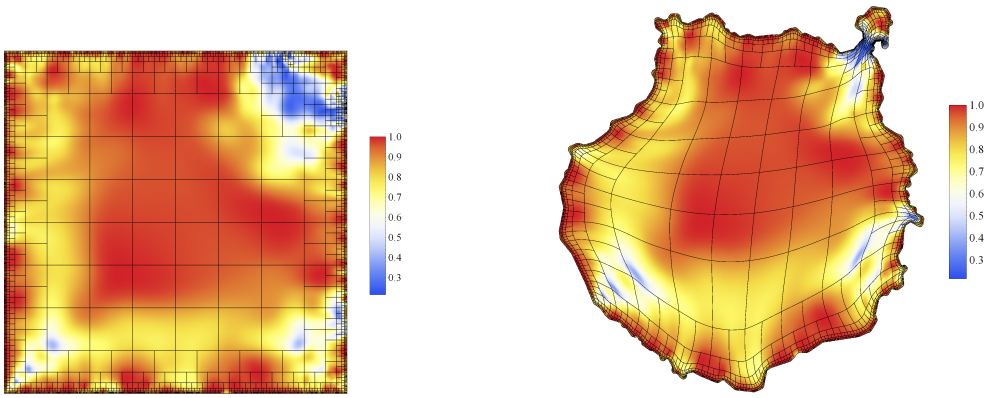

(a) (b) self-intersections. (c) no self-intersections. (d) negative Jacobian. (e) positive Jacobian.

**Figure 6.6:** *Adaptive refinement strategy to improve the parameterization quality in Gran Canaria Island domain. (a) Spline representation of the domain; (b) initial spline parametrization with negative Jacobian; (c) spline parameterization with no negative Jacobian after applying adaptive refinement; (d) mean ratio Jacobian of the initial parametrization; (e) mean ratio Jacobian of the final parametrization.*

#### 6.2.5.2 Adaptive refinement

Parameterization of complex geometries entails a severe distortion that can lead to low quality cells, even cells with negative Jacobian. This can be explained by the lack of degrees of freedom provided by the inner nodes. In order to improve the mesh quality in this case, we have to increase the number of degrees of freedom in the area. We propose an adaptive strategy that refines the cells with low quality. We proceed as follows. For each cell of the mesh, the mean ratio Jacobian is calculated at Gauss quadrature points. We use $16 = 4 \times 4$ quadrature points per cell. A cell $\hat{\Omega}_e$ is marked to refine if, at least, one of its quadrature points has mean ratio Jacobian less than a certain threshold $\delta$. The refined T-mesh is optimized again and the process is repeated until a satisfactory mesh quality is obtained. Figure 6.6 illustrates the proposed strategy. Additional refinements were applied to Gran Canaria Island domain with $\delta = 0.2$. The initial mesh with 3439 cells produces a spline parametric mapping with low quality in some areas and self-intersections in the North East part of the island. After adaptive refinement we obtain a mesh with 3577 cells and positive Jacobian in all the domain. Moreover, the minimum value of mean ratio Jacobian at the quadrature points is 0.21.

Figure 6.7 illustrates spline representation and the colormap of the mean ratio Jacobian of the Gran Canaria Island geometry.




(a) Parametric domain. (b) Physical domain.

**Figure 6.7:** *Gran Canaria Island geometry with* 3577 *cells and* 6054 *control points. Colormap of the mean ratio Jacobian.*

## 6.3 Parameterization method for 3D geometries

To extend the parameterization method to 3D it is necessary to have an optimization technique for 3D hexahedral T-mesh, which is not a trivial task. This issue is studied in another doctoral dissertation of the group.

For now we have implemented an optimization procedure for the regular conformal hexahedral meshes with no T-junctions. Using this procedure we can obtain volumetric parameterization for slightly distorted solids. For example, Fig. 6.1 shows a solid designed using *Rhinoceros* CAD software. The surface of the model is formed by six NURBS surfaces that can be exported and used as input data for construction of volumetric parameterization of the solid by means of our method. This slightly deformed shape can have one-patch volume parameterization using a unit cube as parametric domain. A uniform $8 \times 8 \times 8$ hexahedral mesh is sufficient to represent well all the details of the model, since the input boundary parameterization of each face is defined over $4 \times 4$ parametric grid. The inner nodes of the physical mesh is relocated by applying our optimization procedure for hexahedral meshes, see [66]. Analogously to 2D case, the optimization is based on mean ratio quality metric for a tetrahedron given by

$$q = \frac{3\sigma^{2/3}}{\|S\|^2},$$

where $\sigma = \det(S)$ and $\|S\|$ is the Frobenius norm of $S$, being $S$ the Jacobian matrix of the map from the target element to the physical one.

Trivariate spline mapping $\mathbf{S}$ is obtained by imposing interpolation conditions, where as interpolation points we take the vertices of the mesh. Figure 6.2 illustrates

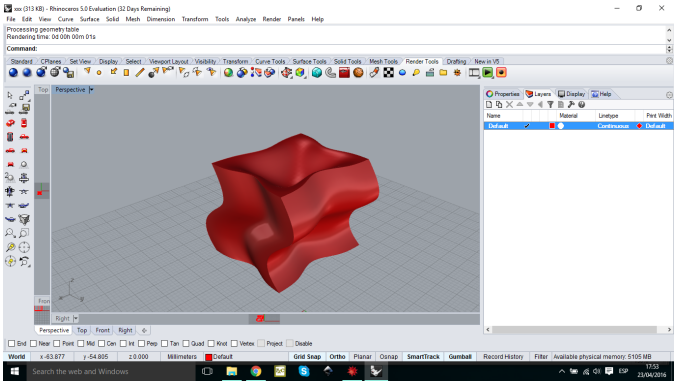


**Figure 6.1:** *Geometry designed in Rhinoceros CAD software.*

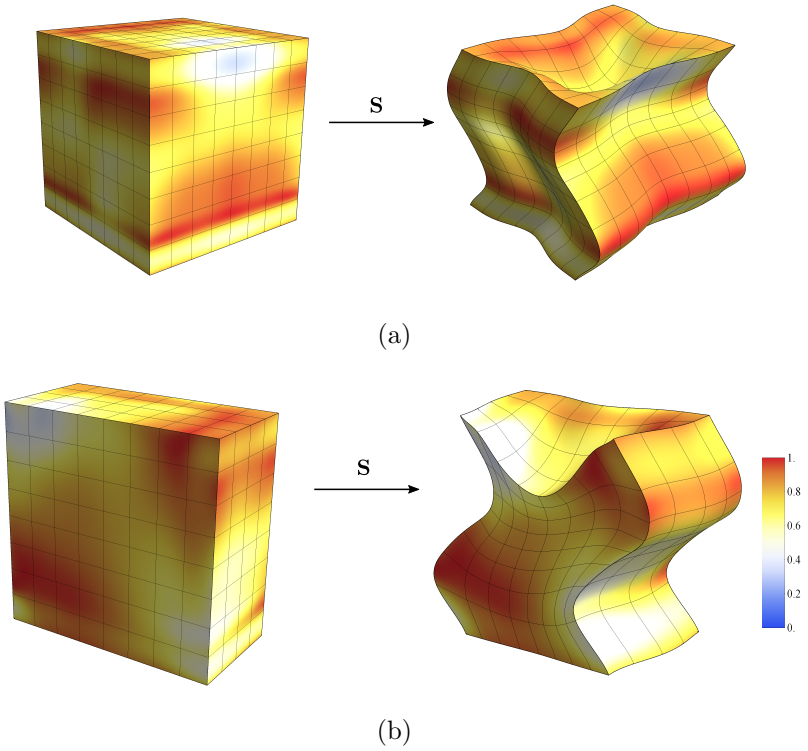



**Figure 6.2:** *Volumetric parameterization of the geometry designed in Rhinoceros. (a) Colormap of mean ratio Jacobian for the parametric mapping* **S**. *(b) A section of the parametric and physical domains.*

the resulting spline parameterization and its quality, that is, the colormap of its mean ratio Jacobian defined as $J_r = \frac{3(\det J)^{2/3}}{\|J\|^2}$, where $J$ is the Jacobian matrix of the parametric mapping $\mathbf{S} : \widehat{\Omega} \to \Omega$ .

# CHAPTER 7

## Testing of the strategy. Computational examples

In this section we test the proposed EP-spline spaces for their use in geometric modelling and Isogeometric Analysis. Approximation properties are tested in different type of problems with singularities, where an adaptive refinement is necessary to achieve an accurate numerical solution. For the analysis computational examples we use the second order elliptic problems with Dirichlet boundary condition, and the exact solution is known for all test problems. The Isogeometric Analysis and the parameterization method codes were implemented in *Mathematica* package. All the adaptive refinement examples include a comparison with uniform (global) refinement. Some of the test problems include a comparison with Finite Element Method, performed with *ALBERTA* FEM software, see `http://www.alberta-fem.de/` and [67]. Finite element discrete approximation space is formed by cubic Lagrange basis functions defined over triangulations, and the adaptive refinement is performed using Kossaczký refinement algorithm [68]. However, we would like to emphasize that this comparison cannot be considered a rigours comparative study of the two methods, because some details should be taken into account. First, Finite Element Analysis is performed over triangulations of computational domain, in contrast to quadrilateral T-meshes for IGA. FEM approximation space is formed by cubic Lagrange functions, meanwhile IGA space is composed by bicubic/tricubic functions. Secondly, the methods are performed with different software, which can influence the resulting accuracy of the numerical solution.

## 7.1 Some preliminaries for performing Isogeometric Analysis

First, let us revise briefly the procedure for solving a problem using Isogeometric Analysis. We consider the following Poisson model problem

$$\begin{cases} -\triangle u = f & \text{in } \Omega, \\ u = g & \text{on } \partial\Omega. \end{cases} \tag{7.1}$$

Its variational formulation consists in finding $u \in V_g(\Omega)$ such that

$$a(u,v) = F(v) \quad \forall v \in V_0(\Omega),$$

where

$$a(u,v) = \int_\Omega \nabla u \cdot \nabla v \, \mathrm{d}\Omega \quad \text{and} \quad F(v) = \int_\Omega f\, v \, \mathrm{d}\Omega.$$

We denote by $\widehat{\Omega} = [0,1]^2$ the parametric domain and by $\Omega$ the physical (computational) domain for our problem. Let $T$ be a T-mesh of $\widehat{\Omega}$, and $\widehat{V}_T = \operatorname{span}\{\widehat{N}_i\}_{i\in I}$ is the finite dimensional space spanned by the spline blending functions defined over the parametric T-mesh $T$. We construct the parametric mapping $\mathbf{S} : \widehat{\Omega} \to \Omega$ as $\mathbf{S} = \sum_{i\in I} \mathbf{P}_i \, \widehat{N}_i$. Then, the discrete approximation space $V_T$ in the physical domain is defined as follows: $V_T = \operatorname{span}\left\{ N_i \,:\, N_i = \widehat{N}_i \circ \mathbf{S}^{-1}, \text{ for all } \widehat{N}_i,\ i \in I \right\}$. Test function space is denoted by $V_{0,T} = \operatorname{span}\{N_i\}_{i\in I_0}$. $V_{g_h,T}(\Omega)$ is the subspace of functions of $V_T$ that are equal to $g_h$ at the boundary, where $g_h$ is an interpolant of $g$.

Isogeometric approximation consists in finding $u_h \in V_{g_h,T}$ such that

$$a(u_h, N_j) = F(N_j) \quad \forall N_j \in V_{0,T}. \tag{7.2}$$

Numerical solution $u_h$ is constructed as linear combination of the basis functions

$$u_h = \sum_{i\in I} c_i \, N_i = \sum_{i\in I_0} c_i \, N_i + \sum_{i\in I\setminus I_0} c_i \, N_i = \sum_{i\in I_0} c_i \, N_i + g_h.$$

Coefficients $\{c_i\}_{i\in I\setminus I_0}$ are found by interpolation of Dirichlet boundary condition and $\{c_i\}_{i\in I_0}$ are found from linear system derived from (7.2)

$$\sum_{i\in I_0} c_i \, a(N_i, N_j) + a(g_h, N_j) = F(N_j), \quad \forall j \in I_0.$$

### 7.1.1 Numerical integration

During the assembly process we accomplish the numerical integration using Gaussian quadrature on each element (cell) of the mesh. It should be taken into account that for exact integration (when possible) the numerical integration should be performed on each Bezier element, that is, on the sub-region of the cell where the blending functions are pure polynomials. For our balanced quadtree/octree meshes it implies to subdivide some cells in 2 or 4 sub-elements in 2D case, and 2, 4 or 8 sub-elements for 3D mesh. It worth mentioning, that taking into account the smoothness of the functions across the sub-elements boundary, more efficient quadrature rules can be used. For example, in [16] authors propose a numerical procedure to compute weights and points of quadrature rules on macro elements. These rules are exact for blending functions and efficient in the sense that requires less evaluations than classical Gauss rules on each element.

### 7.1.2 A posteriori error estimation

The adaptive refinement loop consists in

$$\textit{Solve} \rightarrow \textit{Estimate} \rightarrow \textit{Mark} \rightarrow \textit{Refine.}$$

To perform an adaptive refinement we are going to use the residual-based error estimator given by

$$\eta(\Omega_e)^2 = h^2 \| \left(f + \Delta u_h\right) \|^2_{L^2(\Omega_e)} = h^2 \int_{\Omega_e} \left(f + \Delta u_h\right)^2 \, \mathrm{d}\Omega,$$

where $h$ is the diameter of the cell $\Omega_e$. The estimator does not include the term of gradient jump through the cell interfaces because of the smoothness of the isogeometric approximation. Marking strategy used for adaptive process: a cell $\Omega_e$ is marked to be refined if $\eta(\Omega_e) > \gamma \max_i \{\eta(\Omega_i)\}$, being $\gamma \in [0, 1]$. Normally we use $\gamma = 0.5$.

### 7.1.3 A priori error estimates. Expected order of convergence

Some results about a priori error estimate for $p$-order NURBS-based Isogeometric Analysis, under h-refinement was presented in [13, 69]. For second order elliptic PDE, where the exact solution $u \in H^r(\Omega)$, holds:

$$\|u - u_h\|_{H^1(\Omega)} \leq C h^{\gamma - 1} \|u\|_{H^r(\Omega)}$$

and

$$\|u - u_h\|_{L^2(\Omega)} \leq C h^{\gamma} \|u\|_{H^r(\Omega)},$$

where $\gamma = \min\{p + 1, r\}$.

Thus, the optimal rates of convergence are the same as for Classical Finite Element Analysis with degree $p$ basis functions. For cubic NURBS basis this implies error convergence of order $3/d$ for $H^1$-norm and $4/d$ for $L^2$-norm in terms of degrees of freedom (DOF) for a $d$-dimensional problem ($d = 2$ or 3).

## 7.2 2D test problems

### 7.2.1 Geometric modelling. Spline representation of the surface from its triangulation

We start with the example of geometric modelling. Here we construct a spline representation of a surface given by its triangulation. First, a global parameterization of the surface triangulation is obtained by means of the method proposed by M. Floater in [62]. As a result, we have one-to-one mapping from planar triangulation of the parametric domain (unit square) to the surface triangulation. Then, in order to reproduce well all the features of the surface, a quadtree T-mesh, adapted to the planar triangulation, is constructed. For this purpose we have chosen the following simple criterion. We start from a coarse uniform mesh and refine it until each cell of the mesh contains no more than a certain number of nodes of the input triangulation (3 in our case). Having done so, we define a spline space over the adapted T-mesh $T$. Spline approximation of the surface is built as a linear combination of our bivariate basis functions

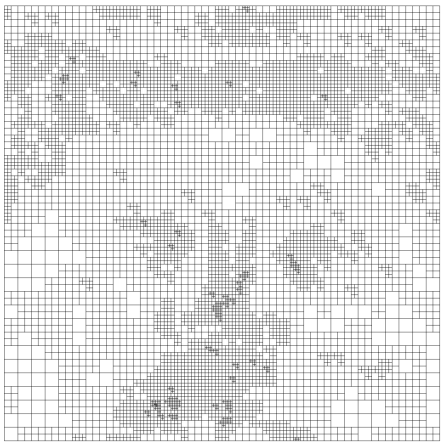
$$S(\boldsymbol{\xi}) = \sum_{\alpha \in A_T} \mathbf{P}_\alpha \, N_\alpha \left(\boldsymbol{\xi}\right).$$

The control points $\mathbf{P}_\alpha \in \mathbb{R}^3$ are found by imposing the interpolation conditions

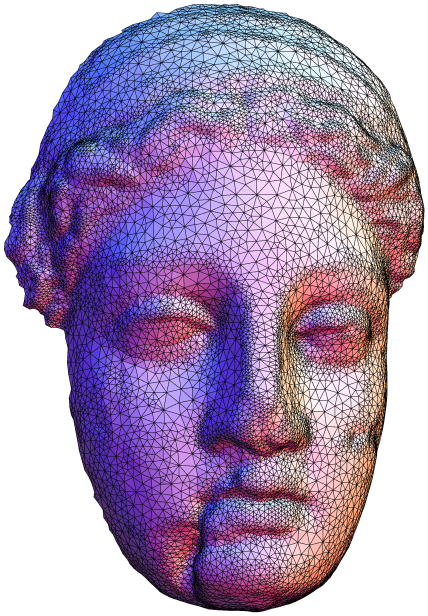
$$\boldsymbol{x}_\beta = \sum_{\alpha \in A_T} \mathbf{P}_\alpha \, N_\alpha \left(\boldsymbol{\xi}_\beta\right), \quad \forall \boldsymbol{\xi}_\beta, \ \beta \in A_T,$$

where $\boldsymbol{\xi}_\beta \in \mathbb{R}^2$ are interpolation points in the parametric domain and $\boldsymbol{x}_\beta \in \mathbb{R}^3$ are their images in the physical space determined by the triangular parameterization. As interpolation points we use the anchors of the functions, i.e., the regular nodes of the T-mesh, and some additional points associated to the functions whose local knot vectors contain repeated boundary knots. These additional points are situated at the midpoint of the edges that have contact with the boundary and coincide approximately with the maximum of the corresponding blending functions. See [63] for more details.

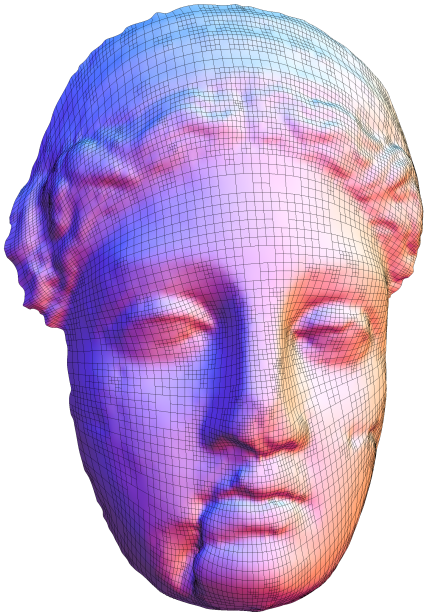
The resulting spline surface, parametric T-mesh and input surface triangulation are shown in Fig. 7.1. Spline representation gains to reproduce well the input surface. It can be appreciated in Fig. 7.2, that, even with less degrees of freedom, the spline representation improves considerably the appearance of input object,



(a) Parametric T-mesh.



(b) Input surface triangulation, 8650 nodes.



(c) Spline representation of the surface.

**Figure 7.1:** *Igea's face. Spline representation of the surface from its triangulation.*

since it obtains a visibly smoother surface.

(a) Input surface triangulation, 8650 nodes. (b) Spline representation of the surface.

**Figure 7.2:** *Igea's face. Comparison of the input triangulation and spline approximation of the surface.*

### 7.2.2 Adaptive refinement for interpolation problem

In this example we interpolate the function with strong singularity

$$u(r) = r^{\frac{1}{2}}, \tag{7.3}$$

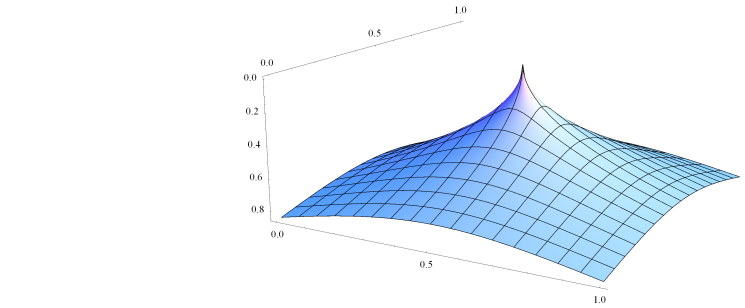
defined in the square domain $[0,1]^2$, being $r = \sqrt{(x-0.5)^2 + (y-0.5)^2}$. Spline approximation of the function (7.3) is built as a linear combination of our blending functions

$$u_h(\boldsymbol{\xi}) = \sum_{\alpha \in A_T} c_\alpha \, N_\alpha \left(\boldsymbol{\xi}\right).$$

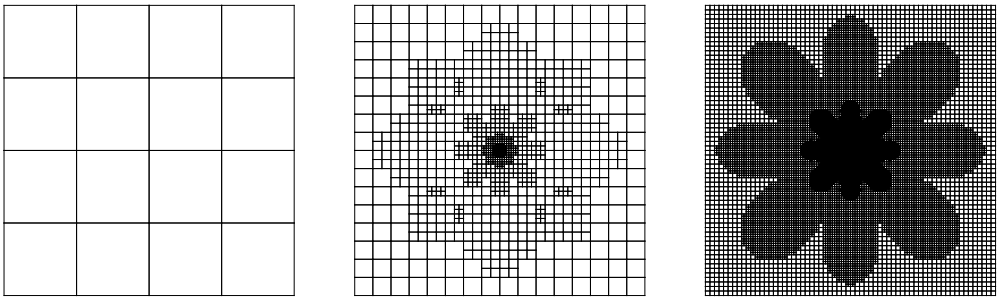
And the control values $c_\alpha$ are found by imposing the interpolation conditions

$$u(\boldsymbol{\xi}_\beta) = \sum_{\alpha \in A_T} c_\alpha \, N_\alpha \left(\boldsymbol{\xi}_\beta\right), \qquad \forall \boldsymbol{\xi}_\beta, \ \beta \in A_T,$$

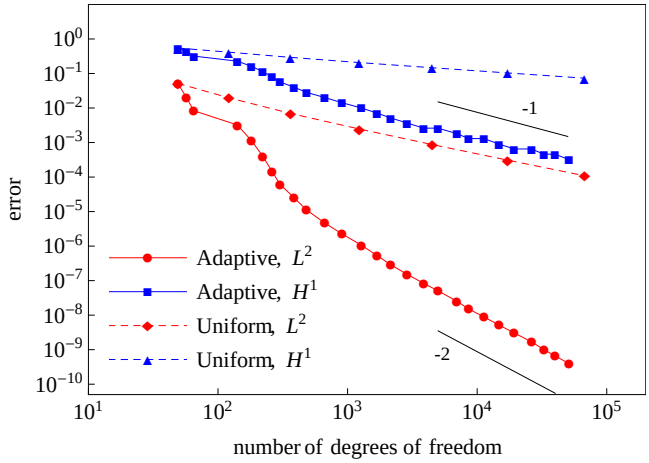
where the interpolation points $\boldsymbol{\xi}_\beta$ are the anchors of the functions, i.e., regular vertices of the mesh. A standard choice for an interpolation with B-splines is



(a)



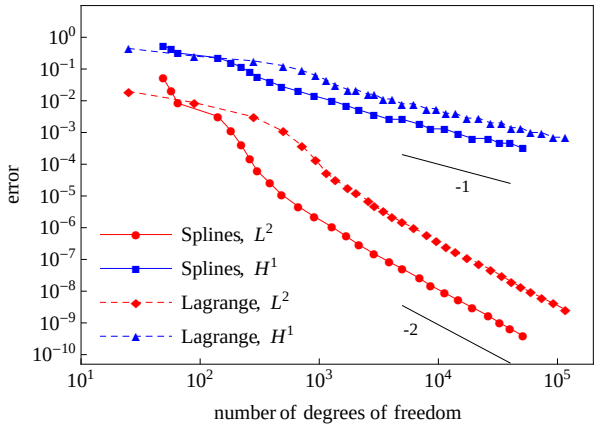
(b) (c) (d)



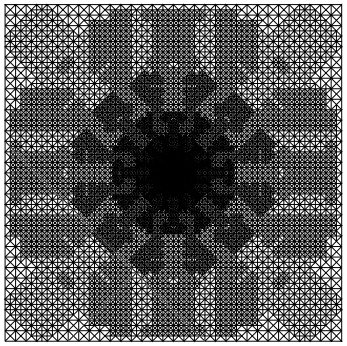

(e) $\|u_h - u\|_{L^2(\Omega)}$ and $|u_h - u|_{H^1(\Omega)}$.

**Figure 7.3:** *Adaptive refinement for interpolation with EP-splines. (a) Function to interpolate. (b) Initial mesh. (c) and (d) some steps of the adaptive refinement. (f) Error convergence of the adaptive refinement in $L^2$-norm and $H^1$-seminorm.*

Greville abscissae, which will be discussed in more details in the section 7.2.5. Generalization of this concept for T-mesh leads to so called Greville collocation points, which for our basis functions coincides with the anchors of the functions, except for the some functions with repeated boundary knots.




(a) $\|u_h - u\|_{L^2(\Omega)}$ and $|u_h - u|_{H^1(\Omega)}$.



(b) Final mesh.

**Figure 7.4:** *Comparison with the interpolation using cubic Lagrange basis functions. (a) Error convergence. (b) Final triangular mesh using Kossaczký refinement algorithm.*

Adaptive refinement is performed according to the indicator based on the exact $L^2$ interpolation error:

$$\eta(\Omega_e)^2 = \|u - u_h\|^2_{L^2(\Omega_e)} = \int_{\Omega_e} (u - u_h)^2 \, \mathrm{d}\Omega.$$

A cell $\Omega_e$ is marked to be refined if $\eta(\Omega_e) > \gamma \max_i \{\eta(\Omega_i)\}$, being $\gamma \in [0, 1]$.

Some steps of the adaptive refinement for the function interpolation (7.3) are shown in Fig. 7.3(b)-(d). Figure 7.3(e) illustrates the convergence in $L^2$-norm and $H^1$-seminorm for the uniform and adaptive refinement. Optimal rate of convergence is observed for $L^2$ error and suboptimal rate for $H^1$ error, which can be considered normal, since the error indicator, used for refinement, is based on the exact $L^2$ error. Figure 7.4(a) shows the error convergence for the interpolation performed using cubic Lagrange basis functions defined over triangulation, where the adaptive refinement is performed using Kossaczký refinement algorithm. The resulting convergence behaviour is similar to one performed with spline functions. Figure 7.4(b) shows the final triangular mesh of the adaptive refinement with Lagrange functions.

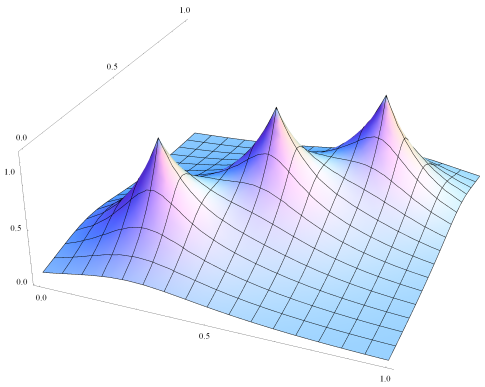
### 7.2.3 Poisson problem in a square domain

In this subsection we present an example of the resolution of a Poisson problem in a square domain $\Omega = [0,1]^2$

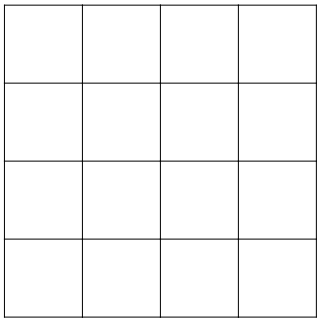
$$\begin{cases} -\Delta u = f & \text{in } \Omega, \\ u = g & \text{on } \partial\Omega. \end{cases} \tag{7.4}$$

In this case the parametric domain coincides with the physical one, and we use the parametric mapping $\mathbf{S}$ which is the identity.

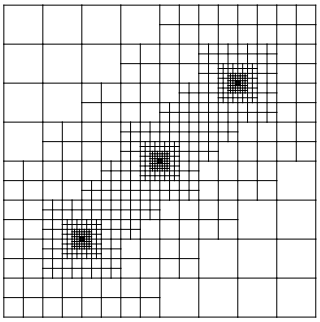
The problem (7.4) is set up in such a way that its exact solution is a function given by



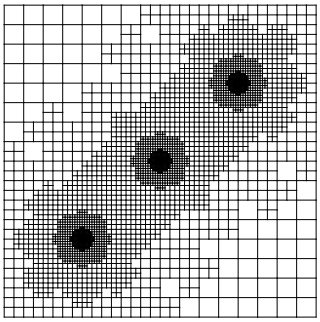

(a)



(b)

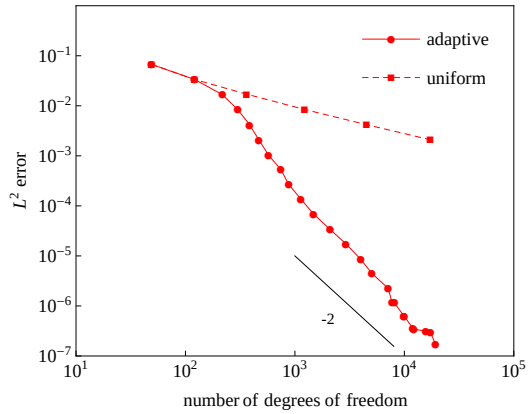


(c)

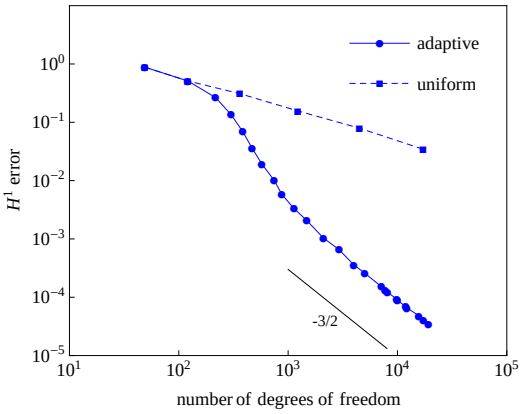


(d)

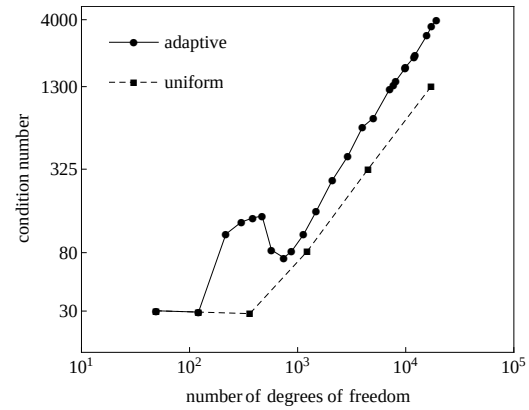
**Figure 7.5:** *Isogeometric analysis with EP-splines for the Poisson problem (7.4). (a) Numerical solution corresponding to the final refinement. (b) Initial mesh. (c), (d) Several steps of the adaptive refinement.*



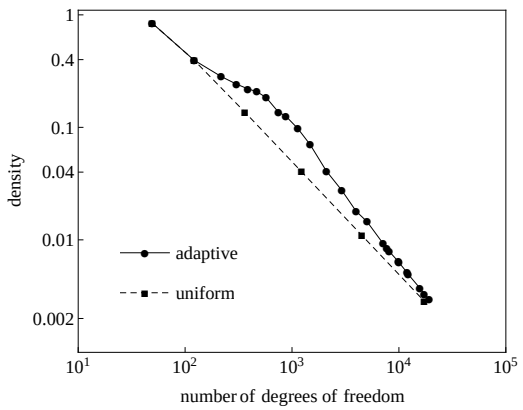

(a) $L^2$ error convergence, $\|u_h - u\|_{L^2(\Omega)}$.




(b) $H^1$ error convergence, $|u_h - u|_{H^1(\Omega)}$.




(c) Condition number of the stiffness matrix.




(d) Density of the stiffness matrix.

**Figure 7.6:** *Error convergence, condition number and density of the stiffness matrix for adaptive IGA for the Poisson problem (7.4).*

$$u(x,y) = \exp\left(-7\sqrt{(x-0.5)^2+(y-0.5)^2}\right) + \\ + \exp\left(-7\sqrt{(x-0.25)^2+(y-0.25)^2}\right) + \\ + \exp\left(-7\sqrt{(x-0.75)^2+(y-0.75)^2}\right).$$

Some steps of adaptive refinement for the problem (7.4) and the numerical solution corresponding to the final refinement iteration are shown in Fig. 7.5. As expected, the error estimator has refined near the three singularity points. The convergence behaviour of the adaptive refinement in $L^2$-norm and $H^1$-seminorm are shown in Fig. 7.6(a) and (b). Suboptimal rates of convergence are obtained for the uniform refinement due to the presence of singularities, and the adaptive refinement gains the optimal rates both for $L^2$ and $H^1$ errors. The evolution of

the density of the stiffness matrix $A$ and its condition number are represented in Fig. 7.6(c) and (d), respectively. The density of the matrix $A$ is the fraction of non-zero elements and the condition number is defined as $\kappa(A) = \| A \|_2 \| A^{-1} \|_2$. It can be observed that the density and the condition number during the adaptive refinement evolve similarly to the uniform refinement.

### 7.2.4 Poisson problem in a complex domain

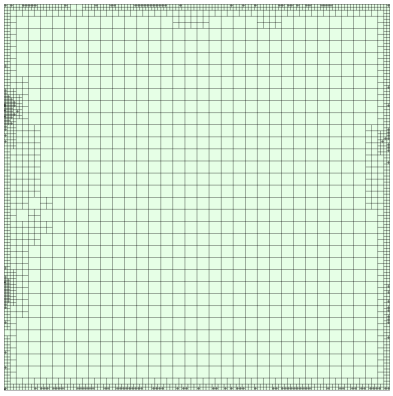
In the next example, we present the result of solving a Poisson problem in a complex domain $\Omega$, which is a puzzle piece, see Fig. 7.7. Parameterization of the computational domain is performed using the algorithm described in Section 6.2. This technique, based on a T-mesh optimization procedure, allows to obtain a good quality parametric mapping suitable for application of IGA. The mean ratio Jacobian $J_r(\boldsymbol{\xi}) = \frac{2 \det(J)}{\|J\|^2}$ is used to evaluate the quality of the parameterization in the sense of its orthogonality and uniformity. Figure 7.7 shows the resulting parameterization and the colormap of its mean ratio Jacobian.

Exact solution for the Poisson problem with Dirichlet boundary condition is a function with steep wave front given by

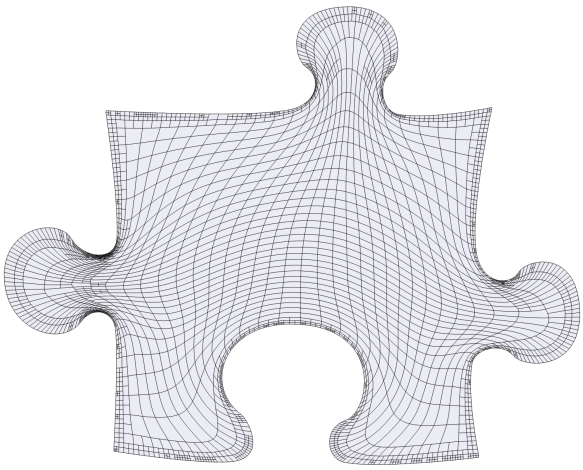
$$u(r) = \arctan(\alpha(r - r_0)),$$

where $r = \sqrt{(x - x_c)^2 + (y - y_c)^2}$. Parameter $\alpha$ determines the steepness of the wave front and $r_0$ is its location. In this example $\alpha = 200$ and $r_0 = 0.6$. The center of the wave front $(x_c, y_c) = (0, 0)$ is situated outside of the computational domain, so the function is smooth in $\Omega$.

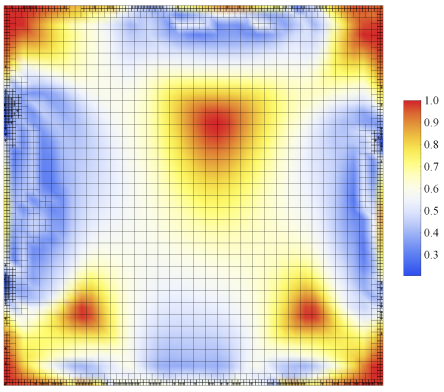
The numerical solution of the problem and the mesh corresponding to the final refinement iteration are shown in Fig. 7.8(a)-(d). As expected, the error estimator has marked for refinement the zone of the wave front. The evolution of the exact error in $L^2$-norm and $H^1$-seminorm are shown in Fig. 7.8(e). Comparison with the uniform refinement is given in Fig. 7.8(f). Optimal rates of convergence are obtained for both adaptive and uniform refinement, due to the smoothness of the exact solution.



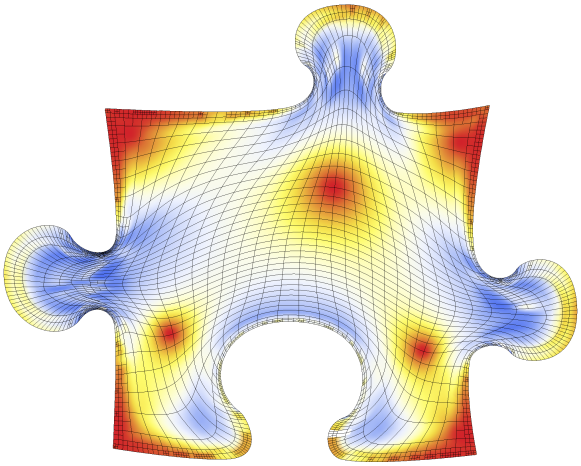
(a) Parametric domain.



(b) Physical domain.

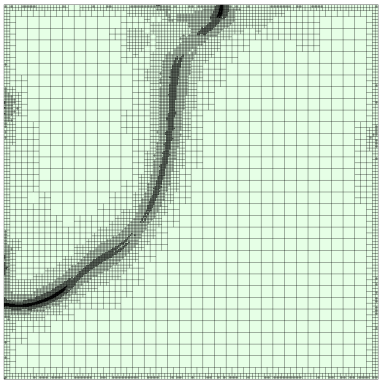



(c) Colormap of the mean ratio Jacobian.

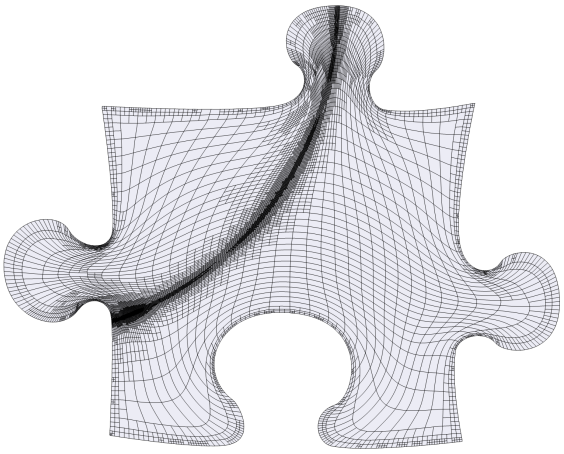


(d) Colormap of the mean ratio Jacobian.

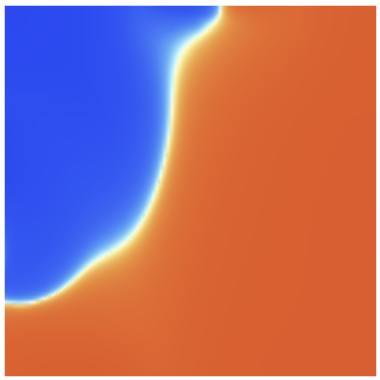
**Figure 7.7:** *Puzzle piece domain. Parameterization of the computational domain for the Poisson problem and its quality (mean ratio Jacobian).*



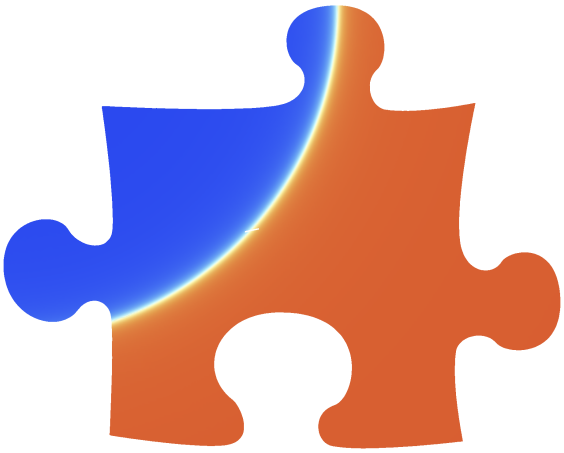
(a) Final refinement in the parametric domain.



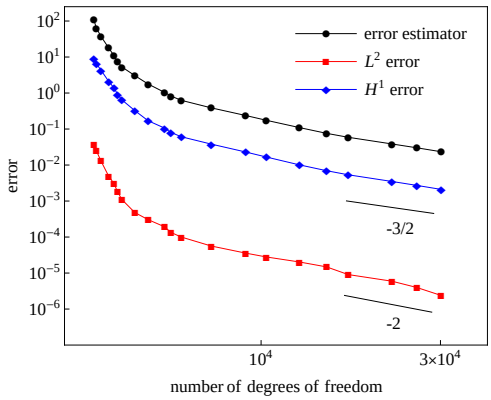
(b) Final refinement in the physical domain.



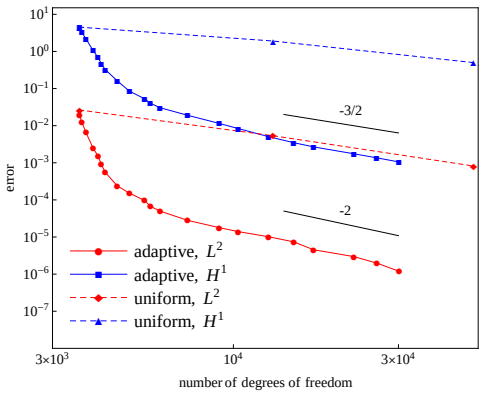
(c) Numerical solution in the parametric domain.



(d) Numerical solution in the physical domain.




(e) Error convergence for the adaptive refinement.




(f) Error convergence for the uniform and adaptive refinements.

**Figure 7.8:** *Results of the IGA adaptive refinement for the Poisson problem in the puzzle piece domain.*

### 7.2.5 Collocation method for Poisson problem in a complex domain

Another advantage of the higher regularity of spline basis functions is the possibility to apply collocation method based on the strong formulation of the problem. The major benefit of collocation approach is the reduced number of evaluation points compared with the Galerkin method. Besides, the bandwidth and matrix sparsity are significantly reduced compared with a Galerkin method, which improves the performance of iterative solvers. Recently IGA collocation method was investigated in several works. A theoretical background for 1D case and some computational examples for 1D, 2D and 3D problems using NURBS-based collocation method was given in [9]. The efficiency of the method in comparison with IGA and standard FEM Galerkin approach was analysed in [10]. The authors report that IGA collocation outperforms in efficiency both Isogeometric Galerkin and Finite Element Methods. Adaptive isogeometric collocation using hierarchical refinement scheme was shown in [10]. Application of isogeometric collocation using analysis-suitable T-splines for second- and fourth-order problems was presented in [11].

Here, we solve the same Poisson problem as in the previous section. So, we have the strong form of the Poisson problem with Dirichlet boundary condition

$$\begin{cases} -\Delta u = f & \text{in } \Omega, \\ u = g & \text{on } \partial\Omega, \end{cases} \tag{7.5}$$

where $\Omega$ is the puzzle piece domain from the previous section. Numerical solution $u_h$ is searched as linear combination of the basis functions

$$u_h = \sum_{i \in I} c_i \, N_i.$$

Collocation method consists in enforce that $u_h$ satisfies the differential equation and the boundary condition in a certain set of points $\{\mathbf{x}_i\}$ that are called *collocation points*. That is,

$$\begin{cases} -\Delta u_h(\mathbf{x}_i) = f(\mathbf{x}_i) & \forall \mathbf{x}_i \in \Omega, \\ u_h(\mathbf{x}_i) = g(\mathbf{x}_i) & \forall \mathbf{x}_i \in \partial\Omega. \end{cases} \tag{7.6}$$

Approximation space should be at least $C^2$ at collocation points, which is the case for our cubic splines.

Analogously to Galerkin method, we split the $u_h$ in two parts:

$$u_h = \sum_{i \in I} c_i \, N_i = \sum_{i \in I_0} c_i \, N_i + \sum_{i \in I \setminus I_0} c_i \, N_i = \sum_{i \in I_0} c_i \, N_i + g_h.$$

Coefficients $\{c_i\}_{i \in I \setminus I_0}$ are found by interpolating Dirichlet boundary condition, i.e.

from the system

$$\sum_{i\in I\setminus I_0} c_i\, N_i(\mathbf{x}_j) = g(\mathbf{x}_j) \quad \forall \mathbf{x}_j \in \partial\Omega.$$

And the coefficients $\{c_i\}_{i\in I_0}$ are found from the linear system

$$-\sum_{i\in I_0} c_i\, \Delta N_i(\mathbf{x}_j) - \Delta g_h(\mathbf{x}_j) = f(\mathbf{x}_j) \quad \forall \mathbf{x}_j \in \Omega, \quad j = 1, \ldots, m.$$

That is,

$$\mathbf{Kc} = \mathbf{f},$$

where
$\mathbf{K}_{j,i} = -\Delta N_i(\mathbf{x}_j), \quad j = 1, \ldots, m$ and $i = 1, \ldots, n = \#I_0$.
$\mathbf{f}_j = f(\mathbf{x}_j) + \Delta g_h(\mathbf{x}_j), \quad j = 1, \ldots, m$
$\mathbf{c} = \{c_1, c_2, \ldots, c_n\}$ are the unknown coefficients.

The system is determined if the number of collocation points coincides with the number of basis functions. Otherwise, if $m > n$, the system is overdetermined and can be solved in a least-squares sense.

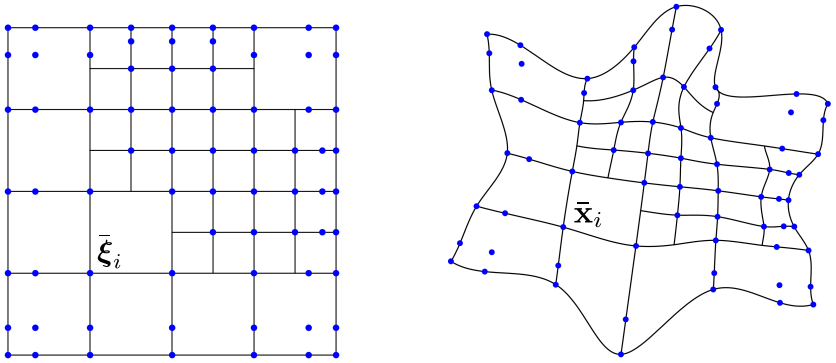
#### 7.2.5.1 Greville abscissae as collocation points

For a stability and good behaviour of the method is crucial to use a suitable set of collocation points. A standard option, working with spline, is Greville abscissae, which are the averages of the knots. Suppose we are given a global knot vector $\{\xi_1, \xi_2, \ldots, \xi_{n+p+1}\}$ that generates $n$ B-spline functions, then its Greville abscissae are defined as

$$\bar{\xi}_i = \frac{\xi_{i+1} + \xi_{i+2} + \cdots + \xi_{i+p}}{p}, \qquad i = 1, \ldots, n.$$

These points correspond approximately to the points, where the B-spline functions attain their maximum and coincide exactly with them for an uniform knot vector, except for the functions with repeated knots on the boundary. Other possible options for collocation points are quadrature points, the maxima of spline basis functions, and the Demko abscissae. Some comparison between Greville abscissae and Demko abscissae can be found in [9], which shows that Greville abscissae may not lead to a proper convergence only in very unusual cases and, in general, it is the simplest and reliable option. Here, we choose Greville abscissae as collocation points. As we deal with T-mesh and we do not have global knot vector, we obtain a set of Greville points in parametric space as the knot average of each local knot vector, that is, for the local knot vectors $\Xi_i = \{\xi_1, \xi_2, \xi_3, \xi_4, \xi_5\}$ and $\mathcal{H}_i = \{\eta_1, \eta_2, \eta_3, \eta_4, \eta_5\}$ of the function $\widehat{N}_i$ its Greville point is defined as $\bar{\boldsymbol{\xi}}_i = (\bar{\xi}_i, \bar{\eta}_i)$, where

$$\bar{\xi}_i = \frac{\xi_2 + \xi_3 + \xi_4}{3} \qquad \text{and} \qquad \bar{\eta}_i = \frac{\eta_2 + \eta_3 + \eta_4}{3}.$$




(a) Parametric domain. (b) Physical domain.

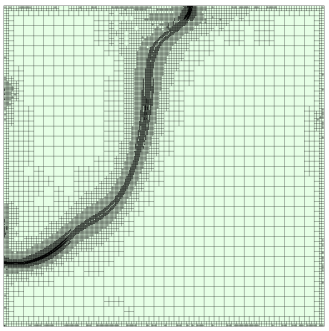
**Figure 7.9:** *Greville collocation points for EP-splines defined over a T-mesh.*

Then, collocation points $\bar{\mathbf{x}}_i$ in the physical space are defined as the images of Greville points $\bar{\boldsymbol{\xi}}_i$, see Fig. 7.9. Due to Condition 1, Greville points for our EP-spline functions coincide with their anchors and thus with the vertices of the T-mesh, except for the functions that contain three repeated knots on the boundary, see Fig. 7.9. Also Greville points coincide with the maximum point for the functions with symmetric knot vectors, and approximately with the maximum point for the rest of the functions.

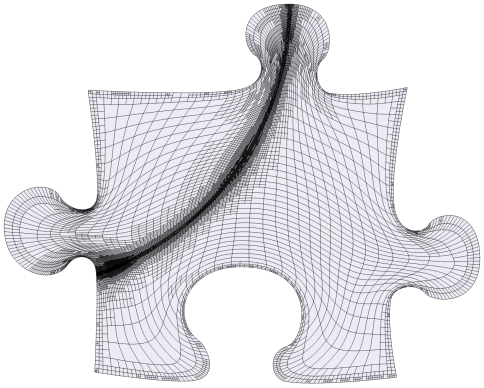
We perform adaptive refinement using the same as for Galerkin method residual-based error estimator. As the result of adaptive refinement we obtain a mesh very similar to the one obtained via Galerkin method for this problem, see Fig. 7.10(a),(b). Error convergence is shown in Fig. 7.10(c). In contrast to Galerkin method, $H^1$ convergence rate is lower than the estimator rate. As reported in [9, 10], there is suboptimal order of convergence for $L^2$ and $H^1$ norm and optimal rate for second derivatives norm, which is consistent with our results. Comparison with Galerkin Isogeometric method is shown in Fig. 7.10(d)-(f). As can be seen, Galerkin method outperforms the collocation method in accuracy per degree of freedom and in rates of convergence.

#### 7.2.5.2 Isogeometric collocation method using superconvergent points

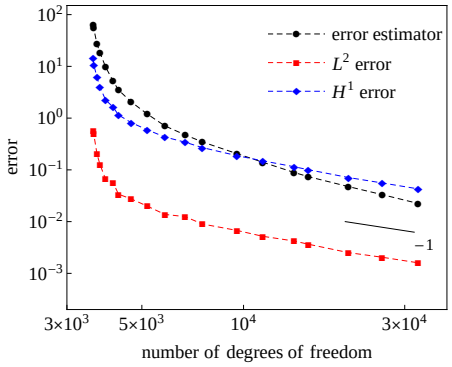
In [12] another option for collocation points, derived from orthogonal collocation and the superconvergence theory, was proposed. The authors report the improved convergence rate in energy norm for odd polynomial degrees compared with Greville collocation points and call them *superconvergent points*. Using these points, collocation solution behaves similarly to the standard Galerkin approximation. For odd degree B-splines and uniform mesh, the Greville abscissae are located at the knots, while the superconvergent points are sited in the interior of each knot span. For cubic uniform B-splines any knot span contains two collocation points that corresponds with the Gauss–Legendre quadrature points $\pm 1/\sqrt{3}$ of the interval. For



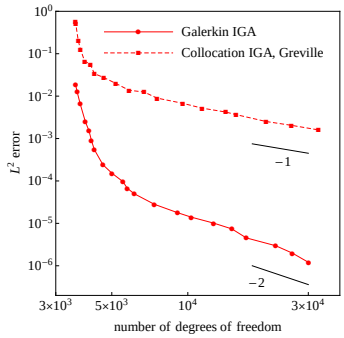
(a) Final refinement, parametric space.



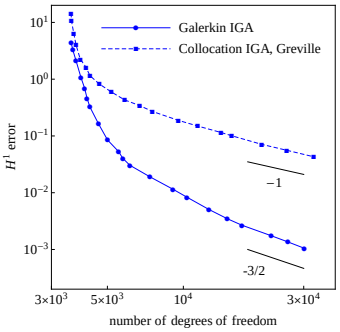
(b) Final refinement, physical space.



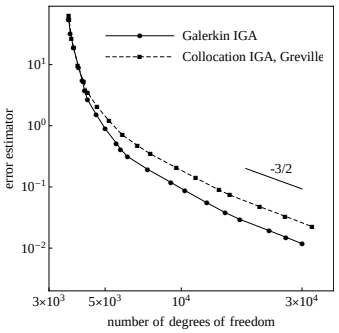

(c) Error convergence.




(d) $L^2$ error.



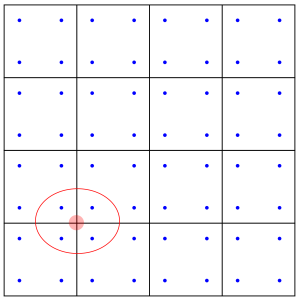

(e) $H^1$ error.



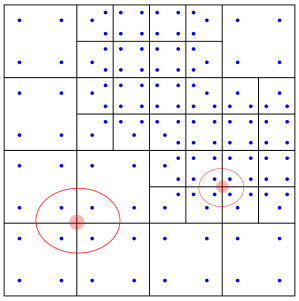

(f) Error estimator.

**Figure 7.10:** *Collocation method using Greville collocation points. Convergence behaviour (c) and its comparison with Galerkin method (d)-(f).*

bivariate uniform B-splines there are $2^2$ collocation points per knot span (2 values for each parametric direction), see Fig. 7.11(a). That leads to an over-determined linear system, which is solved in a least-squares sense.



(a) Superconvergent points for B-splines over uniform mesh.

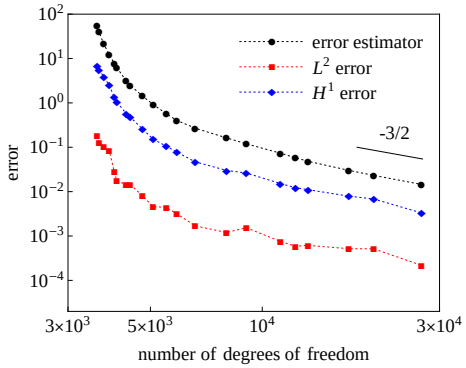


(b) Superconvergent points for EP-splines over T-mesh.

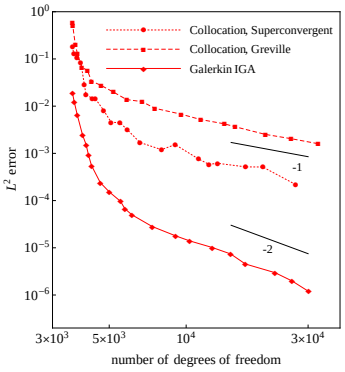
**Figure 7.11:** *(a) Superconvergent collocation points for uniform mesh. (b) Generalization of the idea for EP-splines over a T-mesh.*

In order to extend the idea for our spline spaces over T-mesh, we assign to each blending function 4 collocation points by combining 2 values from each parametric direction. Let us consider a blending function $\widehat{N}_i$ whose local knot vectors are $\Xi_i = \{\xi_1, \xi_2, \xi_3, \xi_4, \xi_5\}$ and $\mathcal{H}_i = \{\eta_1, \eta_2, \eta_3, \eta_4, \eta_5\}$. Then for $\Xi$ vector we take 2 values sited within the knot spans corresponding to $\Delta_2^\xi$ and $\Delta_3^\xi$ of the vector $\Xi$. That is, the point $\bar{\xi}_1 \in (\xi_2, \xi_3)$ is the Gauss quadrature point of the interval $(\xi_2, \xi_3)$ that corresponds to quadrature point $+1/\sqrt{3}$ of the reference interval $[-1, 1]$, and the value $\bar{\xi}_2 \in (\xi_3, \xi_4)$ is the Gauss quadrature point of the interval $(\xi_3, \xi_4)$ corresponding to the quadrature point $-1/\sqrt{3}$ of the reference interval. Analogously we define 2 values $\bar{\eta}_1$ and $\bar{\eta}_2$ for $\mathcal{H}$ knot vector of the function. Finally, we have 4 collocation points for the function $\widehat{N}_i$: $(\bar{\xi}_1, \bar{\eta}_1), (\bar{\xi}_1, \bar{\eta}_2), (\bar{\xi}_2, \bar{\eta}_1), (\bar{\xi}_2, \bar{\eta}_2)$ sited around the anchor of the function, see Fig. 7.11(b). Since for our blending functions $\Delta_2^\xi = \Delta_3^\xi = \Delta_2^\eta = \Delta_3^\eta$, collocation points are sited symmetrically around the anchor, similar to uniform case, see Fig. 7.11(b). However, unlike uniform case, where each cell contains 4 collocation points, in our case a cell can also contain only 3 or 2 points.

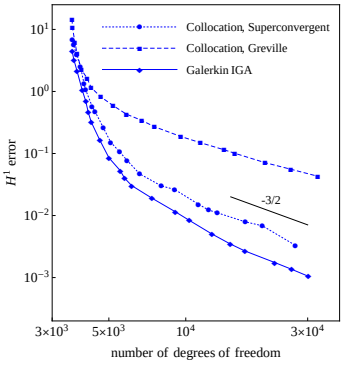
We perform adaptive refinement for the superconvergent collocation method. All conditions and parameters are the same as in the previous experiments for Greville collocation method and Galerkin method. Error convergence and its comparison with Greville collocation and Galerkin isogeometric method are shown in Fig. 7.12. As can be seen, superconvergent collocation gives a better accuracy and rates of convergence compared to Greville collocation. Error estimator, in this case, coincides completely with Galerkin method. Unlike Greville collocation, optimal order of convergence is observed for $H^1$ error and it is very close in accuracy to the Galerkin case. $L^2$ norm error is also considerably improved with respect to Greville case, but it is still inferior in accuracy to Galerkin method.



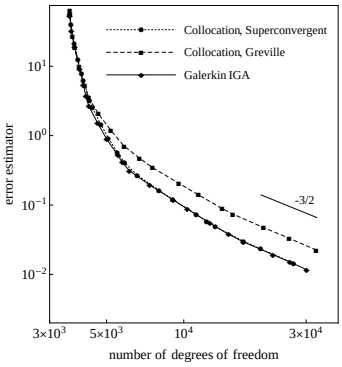

(a) Superconvergent collocation.




(b) $L^2$ error.




(c) $H^1$ error.




(d) Error estimator.

**Figure 7.12:** *Collocation method using superconvergent collocation points. (a) Convergence behaviour. (b)-(d) Comparison with Galerkin method and Greville collocation.*

**Remark 14.** *It is reasonable to suppose that collocation method using superconvergent points gives better results than Greville collocation due to the use of a larger set of collocation points. However, it should be mentioned, that it is not the only reason. The analogous experiments, with the same number of collocation points, equal to the number of superconvergent points, was carried out for some different positions of the points. The resulting convergence behaviour was worst than superconvergent collocation both in accuracy and convergence orders.*

### 7.2.6 Singularly perturbed elliptic equation

In this subsection we solve a singularly perturbed elliptic equation in a square domain $\Omega = [0,1]^2$. Let us consider the reaction-diffusion problem

$$\begin{cases} -\epsilon^2 \Delta u + u = f & \text{in } \Omega, \\ u(0,y) = u(x,0) = 1, \\ u(1,y) = u(x,1) = 0. \end{cases} \tag{7.7}$$

The exact solution of the problem (7.7) contains boundary layers along the two sides of the domain, and it is given by

$$u(x,y) = \left(1 - \frac{e^{-x/\epsilon} + e^{x/\epsilon}}{e^{-1/\epsilon} + e^{1/\epsilon}}\right)\left(1 - \frac{e^{-y/\epsilon} + e^{y/\epsilon}}{e^{-1/\epsilon} + e^{1/\epsilon}}\right), \tag{7.8}$$

where the parameter $\epsilon$ determines the strength of the boundary layer.

The variational formulation of the problem consists in finding $u \in V_g(\Omega)$ such that

$$a(u,v) = F(v) \quad \forall v \in V_0,$$

where

$$a(u,v) = \int_\Omega \left(\epsilon^2 \nabla u \cdot \nabla v + u\, v\right) \, \mathrm{d}\Omega \quad \text{and} \quad F(v) = \int_\Omega f\, v \, \mathrm{d}\Omega.$$

Numerical solution $u_h$ is searched as a linear combination of the basis functions $u_h = \sum_{i \in I} c_i N_i = \sum_{i \in I_0} c_i N_i + g_h$. Discrete weak formulation leads to the linear system

$$(\mathbf{K} + \mathbf{M})\, \mathbf{c} = \mathbf{f}, \qquad \text{where}$$

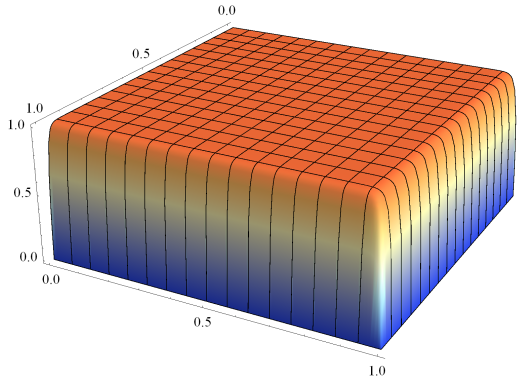
$\mathbf{K}_{j,i} = \int_\Omega \epsilon^2 \nabla N_i \cdot \nabla N_j \, \mathrm{d}\Omega$ is the stiffness matrix,
$\mathbf{M}_{j,i} = \int_\Omega N_i N_j \, \mathrm{d}\Omega$ is the mass matrix and
$\mathbf{f}_j = F(N_j) - a(g_h, N_j)$ is the load vector.

Here, we solve the problem for $\epsilon = 10^{-2}$. For adaptive refinement we use residual-type error estimator given by

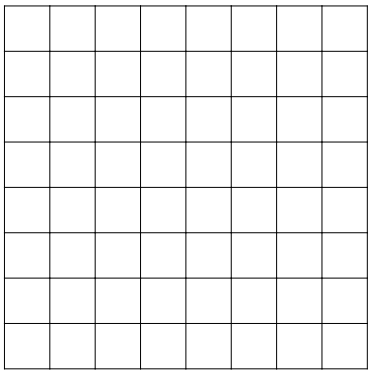
$$\eta(\Omega_e)^2 = h^2 \| \left(f + \epsilon^2 \Delta u_h - u_h\right) \|^2_{L^2(\Omega_e)} = h^2 \int_{\Omega_e} \left(f + \epsilon^2 \Delta u_h - u_h\right)^2 \, \mathrm{d}\Omega.$$

As expected, the estimator marks to refine the boundary layer zone, see Fig. 7.13(c). Error convergence for the adaptive refinement and its comparison with uniform global refinement is shown in Fig. 7.14(a). The optimal rate of convergence is obtained for both adaptive and uniform refinement, due to the regularity of the exact solution. Also we solve the problem with Finite Element Method using 3rd order Lagrange basis functions over triangular meshes. Final mesh of the adaptive refinement with FEM is given in Fig. 7.13(e). Figure 7.14(b)(c) illustrates the

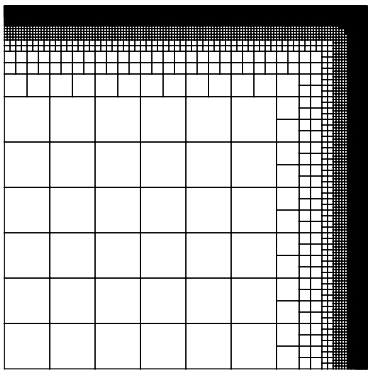
comparison of the error evolution between IGA and FEM results. It can be seen that both method offer the expected order of convergence. However, IGA gives a slightly better accuracy per degrees of freedom both for adaptive and uniform refinement.



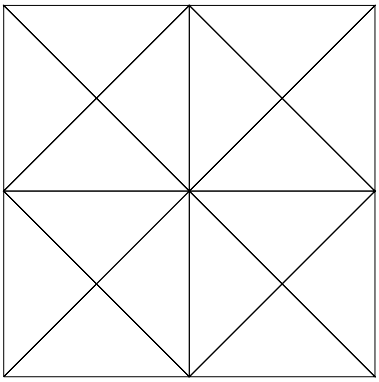

(a) Numerical solution.



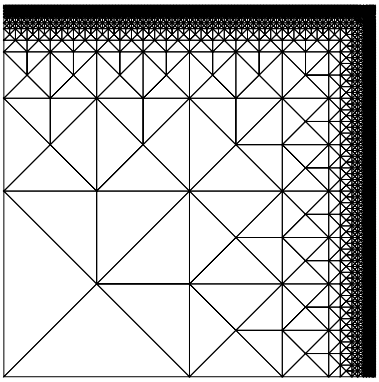
(b) Initial mesh, IGA.



(c) Final mesh, IGA.

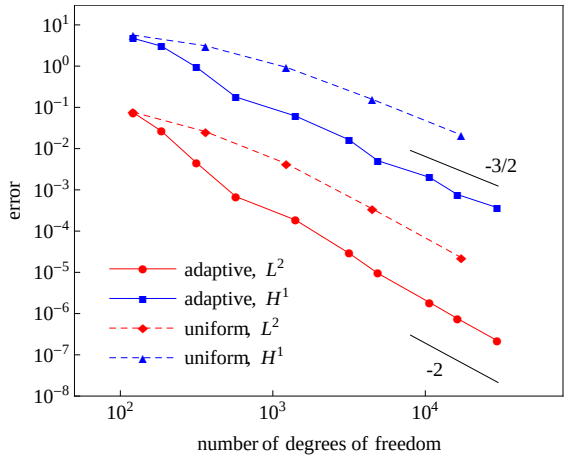


(d) Initial mesh, FEM.

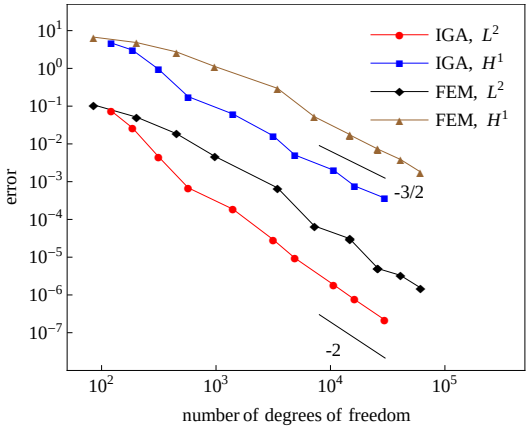


(e) Final mesh, FEM.

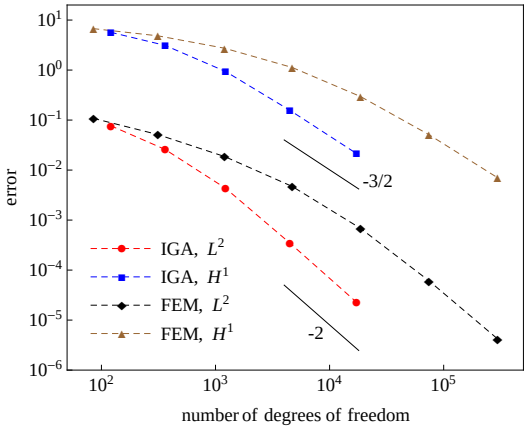
**Figure 7.13:** *Isogeometric analysis for the singularly perturbed elliptic equation (7.7).*




(a) Adaptive and uniform refinement, IGA.




(b) Adaptive refinement, IGA and FEM.




(c) Uniform refinement, IGA and FEM.

**Figure 7.14:** *Singularly perturbed elliptic equation (7.7). Error convergence and its comparison with FEM.*

### 7.2.7 Eigenvalue problem in a square domain

Eigenvalue problem arises in different context of physics and engineering, e.g. structural vibration analysis, acoustic and electromagnetic wave phenomena, optics. In this section we test the behaviour of Isogeometric Analysis for eigenvalue problem in comparison with Finite Element Method. Considerably superior accuracy of IGA, comparing with Finite Element Analysis, for spectrum analysis was reported in [4].

We consider an eigenvalue problem in the domain $\Omega = [0,1]^2$ that consist in finding $\lambda \in \mathbb{R}$ and $u \neq 0$ such that

$$\begin{cases} -\Delta u = \lambda u & \text{in } \Omega, \\ u = 0 & \text{on } \partial\Omega, \\ \|u\| = 1. \end{cases} \tag{7.9}$$

In this simple case the analytical solution of this problem is known. It has a countable sequence of eigenpairs given by

$$(\lambda_{n,m}, u_{n,m}) = \left(\pi^2(n^2+m^2), 2\sin(\pi n x)\sin(\pi m y)\right), \quad n, m = 1, 2, \ldots.$$

The variational formulation of the problem (7.9) consists in finding $\lambda \in \mathbb{R}$ and $u \in V_0$ such that

$$\begin{cases} a(u,v) = \lambda\,(u,v) & \forall v \in V_0, \\ \|u\| = 1. \end{cases} \tag{7.10}$$

where

$$a(u,v) = \int_\Omega \nabla u \cdot \nabla v \, \mathrm{d}\Omega, \quad (u,v) = \int_\Omega u\, v \, \mathrm{d}\Omega.$$

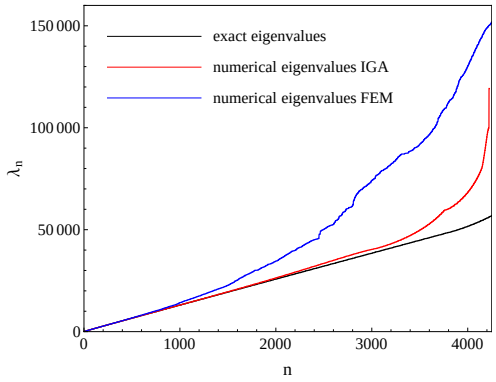
For the finite dimensional space $V_{0,T} = \{N_i\}_{i=1}^n$ the solution is searched as $u_h = \sum_{i=1}^n c_i N_i$. The discretized eigenvalue problem for (7.10) consists in determine $\lambda_h \in \mathbb{R}$ and $u_h \in V_{0,T}$ such that

$$\begin{cases} a(u_h, N_j) = \lambda_h (u_h, N_j) & \forall N_j \in V_{0,T}, \\ \|u\| = 1. \end{cases} \tag{7.11}$$

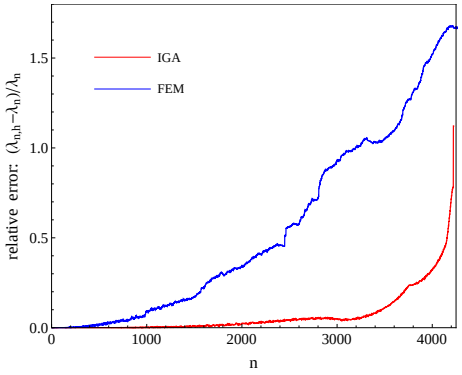
Problem (7.11) leads to a generalized algebraic eigenvalue problem

$$\mathbf{K}\mathbf{c} = \lambda_h \mathbf{M}\mathbf{c}, \tag{7.12}$$

being
$\mathbf{K}_{j,i} = a(N_i, N_j)$ the stiffness matrix,
$\mathbf{M}_{j,i} = (N_i, N_j)$ the mass matrix,
$\mathbf{c} = \{c_1, c_2, \ldots, c_n\}$ is a eigenvector that defines a numerical eigenfunction $u_h$, i.e.




(a) Sorted sequences of the exact eigenvalues $\lambda_n$ and numerical eigenvalues $\lambda_{n,h}$ obtained with IGA and FEA.




(b) Relative error of the numerical eigenvalues: $(\lambda_{n,h}-\lambda_n)/\lambda_n$ for IGA and FEM.

**Figure 7.15:** *Comparison of the numerical spectrum obtained via IGA and FEM with the exact spectrum of continuous problem.*

$u_h = \sum_{i=1}^{n} c_i\, N_i$. Matrices $\mathbf{K}$ and $\mathbf{M}$ are symmetric positive defined. The problem (7.12) have a finite sequence of real eigenvalues

$$0 < \lambda_{1,h} \leq \lambda_{2,h} \leq \lambda_{3,h} \leq \cdots \lambda_{n,h},$$

which correspond to eigenfunctions

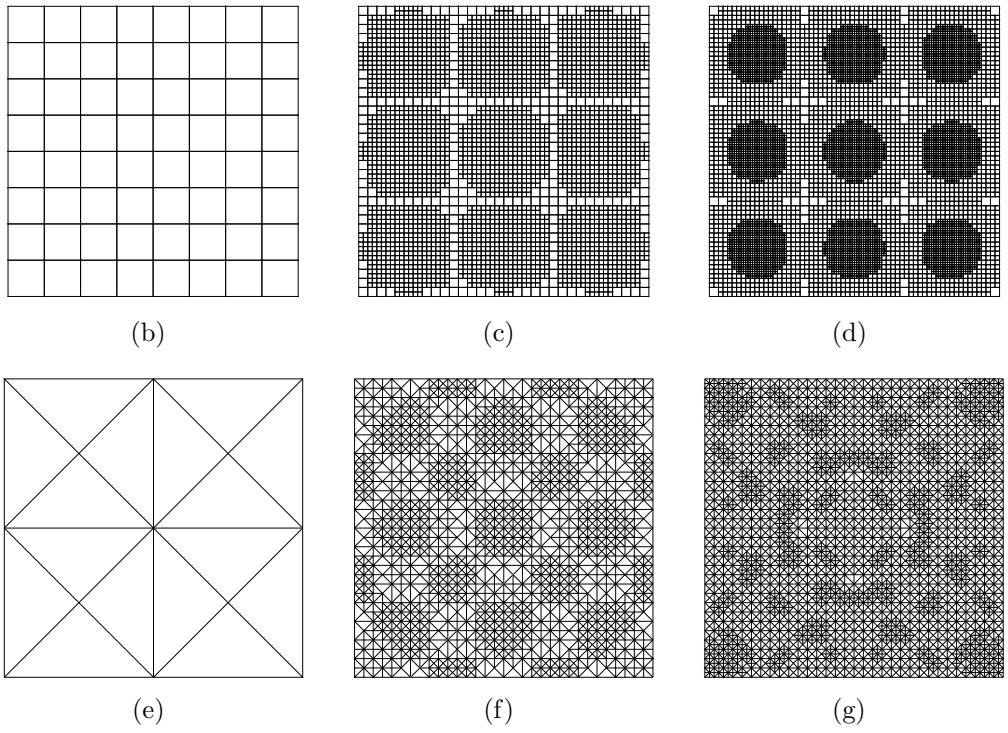
$$u_{1,h},\ u_{2,h},\ u_{3,h},\ \cdots\ , u_{n,h}.$$

Here we perform two numerical experiments.

The purpose of the first one is to compare the numerical spectrum obtained with IGA and FEA. We solve the eigenvalue problem (7.9) using Isogeometric Analysis over a uniform $64 \times 64$ mesh, which corresponds to 4225 basis functions, and using Finite Element Method with 3rd order Lagrange functions over a uniform triangular mesh with a similar number of degrees of freedom (4513). We compare the sequence of numerical eigenvalues $\lambda_{n,h}$, sorted in increasing order, with the exact values $\lambda_n$. The corresponding sequences of eigenvalues are plotted in Fig. 7.15(a). Isogeometric result includes 4225 eigenvalues and FEM sequences contains 4513 values. Abscissa of each point represents the index of the eigenvalue. Figure 7.15(b) shows relative error for the same sorted numerical spectrum: $(\lambda_{n,h}-\lambda_n)/\lambda_n$. As can be seen, the FEM eigenvalues have considerably superior error and the difference becomes bigger for the highest eigenvalues.

In the second experiment we perform an adaptive refinement in order to approximate well the 11-th eigenfunction $u_{11}$. The index 11 corresponds to a simple

(a) Numerical solution for the eigenfunction $u_{11} = u_{3,3} = 2 \sin 3\pi x \sin 3\pi y$.



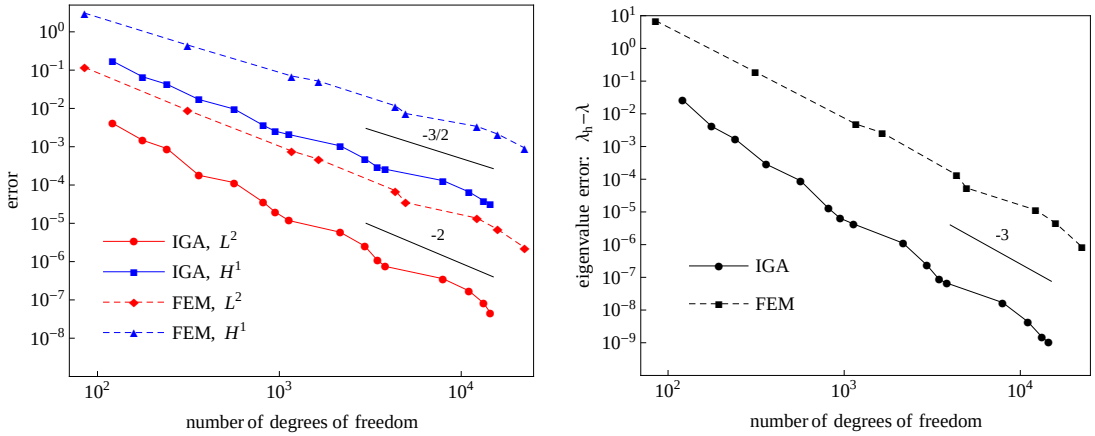
(b) (c) (d)

(e) (f) (g)

**Figure 7.16:** *Eigenvalue problem. Adaptive refinement to approximate the eigenfunction $u_{11}$ and its eigenvalue $\lambda_{11}$. (a) Numerical solution for the final refinement step. (b)-(d) some steps of adaptive refinement with IGA. (e)-(g) some steps of adaptive refinement with FEA.*

(no repeated) eigenvalue with indexes $n = 3, m = 3$:

$$(\lambda_{11}, u_{11}) = (\lambda_{3,3}, u_{3,3}) = (18\pi^2, 2 \sin 3\pi x \sin 3\pi y), \quad \text{see Fig. 7.16(a).}$$




(a) Error convergence for the eigenfunction: $\|u_{11,h} - u_{11}\|_{L^2}$ and $|u_{11,h} - u_{11}|_{H^1}$.

(b) Error convergence for the eigenvalue: $(\lambda_{11,h} - \lambda_{11})$.

**Figure 7.17:** *Eigenvalue problem. Adaptive refinement to approximate the eigenpair* $(\lambda_{11}, u_{11})$*. Convergence behaviour.*

Residual-type error estimator for $k$-th eigenpair is given by

$$\eta(\Omega_e)^2 = h^2 \| \left(\Delta u_{k,h} + \lambda_{k,h} u_{k,h}\right) \|^2_{L^2(\Omega_e)} = h^2 \int_{\Omega_e} \left(\Delta u_{k,h} + \lambda_{k,h} u_{k,h}\right)^2 \, \mathrm{d}\Omega.$$

Some steps of the adaptive refinement for Isogeometric Analysis and Finite Element Analysis are shown in Fig. 7.16. As can be observed, adaptive refinements lead to almost uniform refinement, since the exact solution is a smooth function. Error convergence for the eigenfunction $u_{11}$ and its eigenvalue are shown in Fig. 7.17. It can be seen that both methods offer the optimal rates of convergence. However, IGA obtains a considerably better accuracy of the numerical solution for the eigenpair $(\lambda_{11}, u_{11})$.

### 7.2.8 Kellogg function

This problem was designed by Kellogg while studying Poisson problems with intersecting interfaces. We consider elliptic boundary value problem with discontinuous diffusion coefficient in the domain $\Omega = [-1, 1]^2$

$$-\operatorname{div}(A(\mathbf{x})\nabla u) = f \qquad \text{in } \Omega, \tag{7.13}$$

where the coefficient $A$ is piecewise constant: $A = a_1$ in the first and third quadrant, and $A = a_2$ in the second and fourth quadrant.
For $f \equiv 0$ the solution is given in polar coordinates by

$$u(r, \theta) = r^a \mu(\theta),$$

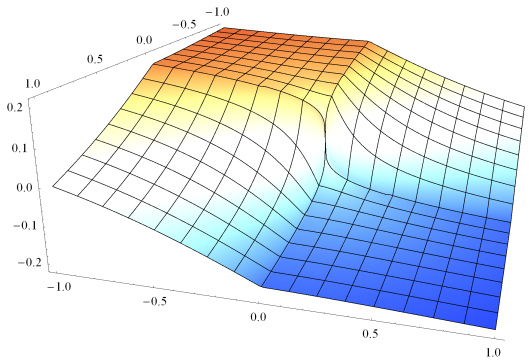



**Figure 7.18:** *Kellogg function for* $a = 0.25$.

where

$$\mu(\theta) = \begin{cases} \cos\left((\pi/2 - c)a\right)\cos\left((\theta - \pi/2 + b)a\right) & \text{if} \quad 0 \le \theta \le \pi/2, \\ \cos\left(ab\right)\cos\left((\theta - \pi + c)a\right) & \text{if} \quad \pi/2 \le \theta \le \pi, \\ \cos\left(ac\right)\cos\left((\theta - \pi - b)a\right) & \text{if} \quad \pi \le \theta \le 3\pi/2, \\ \cos\left((\pi/2 - b)a\right)\cos\left((\theta - 3\pi/2 - c)a\right) & \text{if} \quad 3\pi/2 \le \theta \le 2\pi, \end{cases}$$

and the numbers $a$, $b$ and $c$ satisfy the relations

$$\begin{cases} R := a_1/a_2 = -\tan\left((\pi/2 - c)a\right)\cot\left(ab\right), \\ 1/R := -\tan\left(ab\right)\cot\left(ac\right), \\ R = -\tan\left(ac\right)\cot\left(\pi/2 - b\right)a, \\ 0 < a < 2, \\ \max\{0, \pi a - \pi\} < 2ab < \min\{\pi a, \pi\}, \\ \max\{0, \pi - \pi a\} < -2ac < \min\{\pi, 2\pi - \pi a\}. \end{cases} \tag{7.14}$$

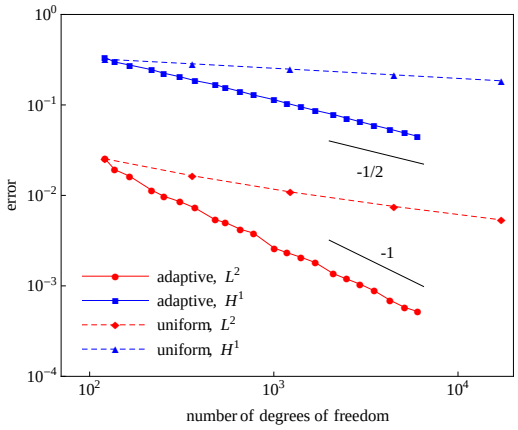
The exact solution has a discontinuous derivative along the axes $y = 0$, $x = 0$ and infinite derivative at the origin. It is known that the solution $u \in H^{1+a}$.

Here, we choose $a = 0.25$. Then, from (7.14), the following parameters for the solution of the problem (7.13) can be obtained: $a_1 \approx 25.27414236908818$, $a_2 = 1$, $b = \pi/4$ and $c \approx -5.49778714378214$. We have taken these data from [70]. The function $u$ for $a = 0.25$ is shown in Fig. 7.18.

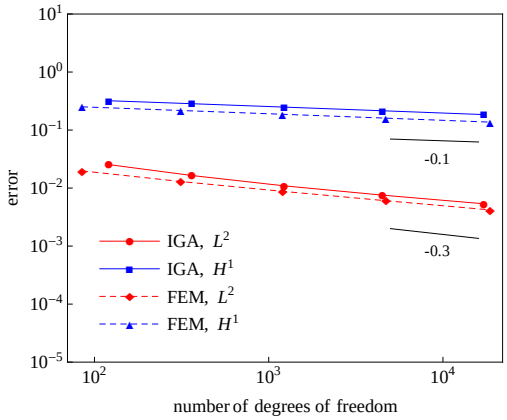
Due to the discontinuous diffusion coefficient, the error estimator for this problem includes the term for the gradient jump across the edges of the cell. Namely, the estimator is given by

$$\eta(\Omega_e)^2 = h^2 \int_{\Omega_e} \left(f + \operatorname{div}(A(\mathbf{x})\nabla u_h)\right)^2 \, \mathrm{d}\Omega + h \int_{\partial\Omega_e} \left[A(\mathbf{x})\frac{\partial u_h}{\partial n}\right]^2 \, \mathrm{d}\Gamma.$$

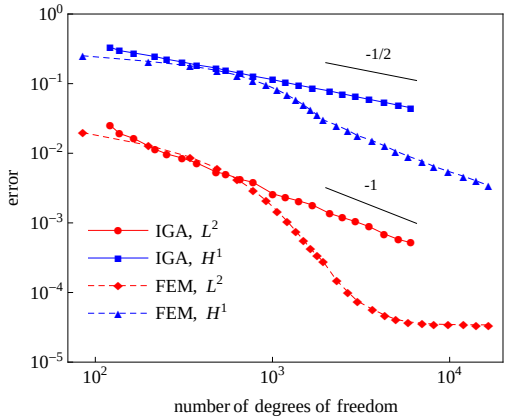
This test problem is a challenging test for two reasons.




(a) Uniform and adaptive refinement, $C^2$ IGA.




(b) Uniform refinement, FEM and $C^2$ IGA.




(c) Adaptive refinement, FEM and $C^2$ IGA.

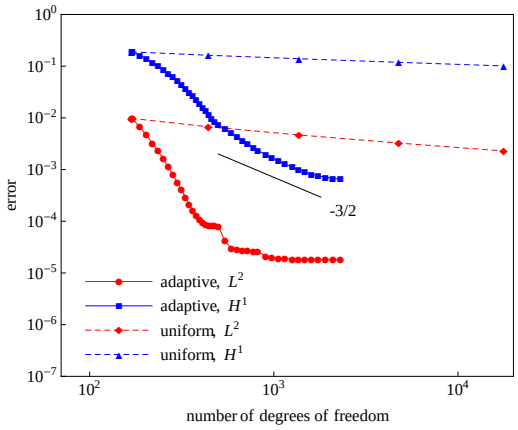
**Figure 7.19:** *Kellogg function. Poor error convergence for the adaptive refinement using $C^2$-continuous EP-spline spaces and its comparison with FEM solution.*

First, the problem has discontinuous diffusion coefficient, the solution has discontinuous derivative along the interfaces and infinite derivative at $(0, 0)$. The $C^2$-continuous space seems to be not appropriate for approximating the solution of this problem. Very poor convergence order was obtained for the adaptive refinement using standard $C^2$-continuous Isogeometric Analysis, see Fig. 7.19. Uniform refinement with $C^2$ spline spaces gives a similar to FEM error behaviour, Fig. 7.19(b); and for the adaptive refinement with $C^2$ spline spaces the convergence rates are considerably inferior to FEM's rates, Fig. 7.19(c). None of the methods gained optimal rates with adaptive refinement. To tackle this problem we repeat the knots in the interior of the domain to define spline spaces that are $C^0$-continuous along the axes. The resulting error convergence for $C^0$ IGA is given in Fig. 7.20. For the uniform refinement with $C^0$ spline spaces the order of convergence coincides with the order of the $C^2$ case, and the accuracy is improved

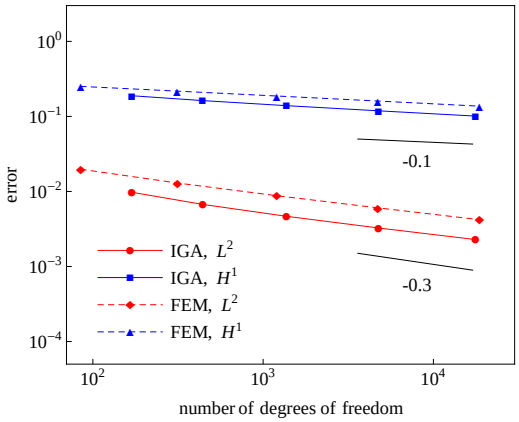
slightly with respect to the $C^2$ case. However, for the adaptive refinement using $C^0$-continuous spline space the resulting convergence is improved considerably, see Fig. 7.20(c). As can be seen, the convergence rates and accuracy are superior to Finite Element results. Suboptimal convergence order for $H^1$ error was obtained for the FEM adaptive refinement. Note that a stagnation of $L^2$ error, both for FEM and IGA method, takes place. The final meshes of adaptive refinement for IGA and FEM are shown in Fig. 7.20(d) and (e).

**Remark 15.** *Using $C^0$-continuous spline spaces was beneficial for this problem. For that it was necessary to use multiple interior knots along the axes. In this particular case it was easy to create a repeated knots structure along the entire knot line of the domain. This implies to change knot intervals of some functions (repeating the knots) and add some additional functions to obtain a complete polynomial space. However, in general case, it is not so straightforward to define functions with reduced regularity along a certain interval (or point) of the domain, which can be necessary for some problems. For that we need to extend the strategy for T-meshes with repeated interior knots.*

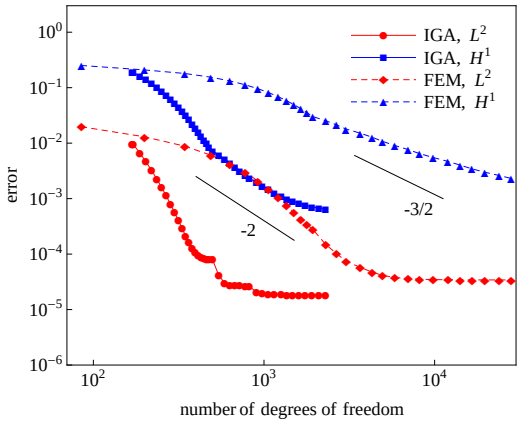
Second, due to the strong singularity at the origin, during adaptive refinement the estimator marks to refine at each iteration a very reduced zone around the singular point $(0, 0)$. That leads to excessive accumulation of function support in this zone, which affects the condition number of the matrix and its sparsity. To avoid this phenomenon, we can apply the strategy explained in Section 5.5. The idea is to impose some additional refinements in order to extend the refined zone and not allow an excessive support accumulation during adaptive refinement. The results of the adaptive refinement using this approach are shown in Fig. 7.21. As can be seen, the application of this strategy allows to "delay" the stagnation of the error convergence and arrive to a higher accuracy of numerical solution. Final adapted mesh, which is more refined compared with the standard adaptive process, is shown in Fig. 7.21(b). It is worth mentioning that IGA refinement process for this problem leads to a very high mesh discretization level. Namely, the depth of the final quadtree is 42. This depth corresponds to the cell size of the order $10^{-13}$, which is close to the working precision. This fact, probably, can explain the stagnation of the errors.



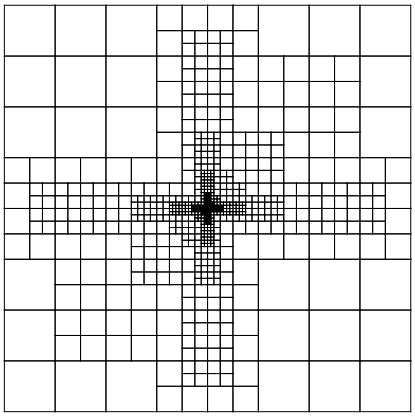

(a) Uniform and adaptive refinement, $C^0$ IGA.



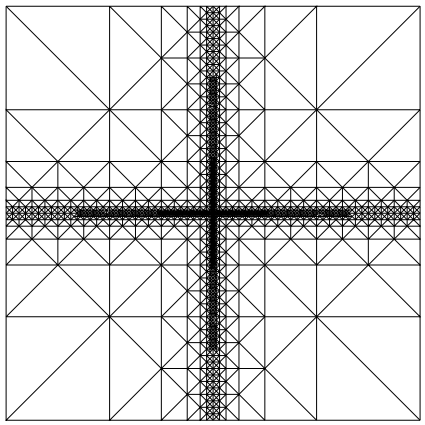

(b) Uniform refinement, $C^0$ IGA and FEM.




(c) Adaptive refinement, $C^0$ IGA and FEM.

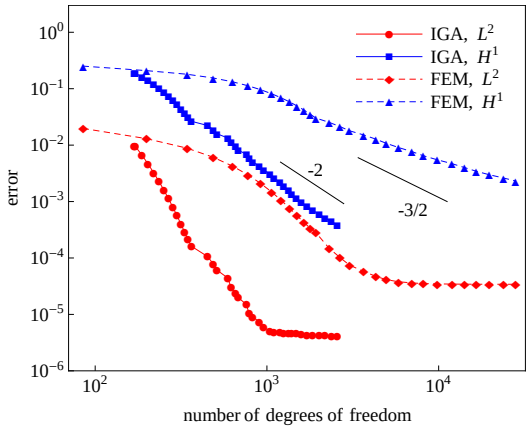


(d) Final mesh, IGA.

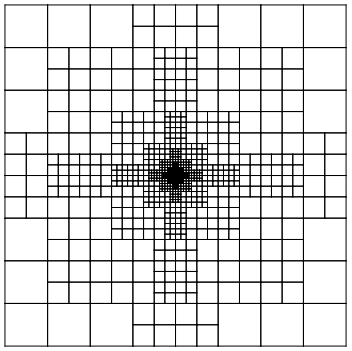


(e) Final mesh, FEM.

**Figure 7.20:** *Kellogg function. Isogeometric Analysis using $C^0$-continuous EP-spline spaces and its comparison with FEM solution.*




(a) Error convergence, $C^0$ IGA.



(b) Final mesh of the adaptive process.

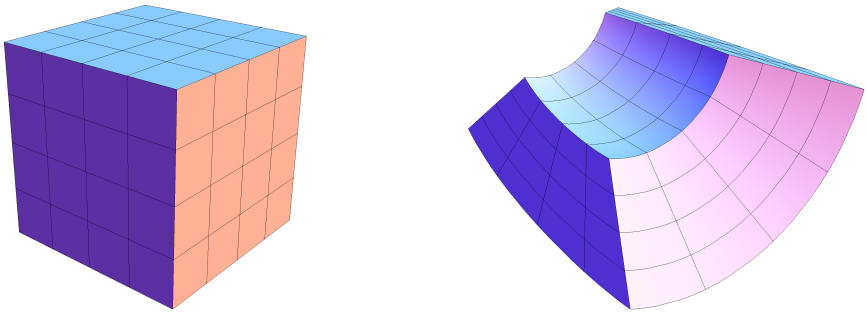
**Figure 7.21:** *Kellogg function. Isogeometric Analysis using $C^0$-continuous EP-spline spaces. Adaptive refinement imposing additional refinements to avoid excessive support accumulation.*

## 7.3 3D test problems

### 7.3.1 Poisson problem in a sphere portion domain

The first computational example in 3D is a Poisson problem over the geometry, which is a spline approximation of a sphere portion, see Fig. 7.1. The initial uniform mesh, used for domain parameterization, is composed by $4 \times 4 \times 4$ cells. The domain parameterization is constructed by interpolation procedure, where the images of the parametric points are mapped to the physical space via analytical expression for the sphere portion, namely $s(\xi, \eta, \zeta) = (x(\xi, \eta, \zeta), y(\xi, \eta, \zeta), z(\xi, \eta, \zeta))$, where

$$x(\xi, \eta, \zeta) = \frac{\zeta + 1}{2} \cos\left(\frac{\pi}{2}\eta\right) \sin\left(\frac{\pi}{4}(\xi + 1)\right),$$
$$y(\xi, \eta, \zeta) = \frac{\zeta + 1}{2} \sin\left(\frac{\pi}{2}\eta\right) \sin\left(\frac{\pi}{4}(\xi + 1)\right),$$
$$z(\xi, \eta, \zeta) = \frac{\zeta + 1}{2} \cos\left(\frac{\pi}{4}(2\xi + 1)\right).$$



(a) Parametric domain. (b) Physical domain.

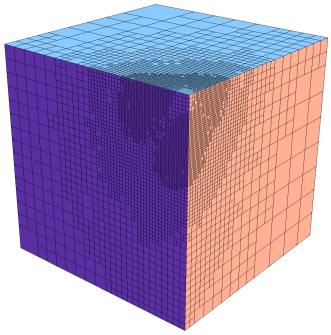
**Figure 7.1:** *Spline approximation of sphere portion domain.*

The Poisson problem with Dirichlet boundary condition is set up so that the analytical solution of the problem is

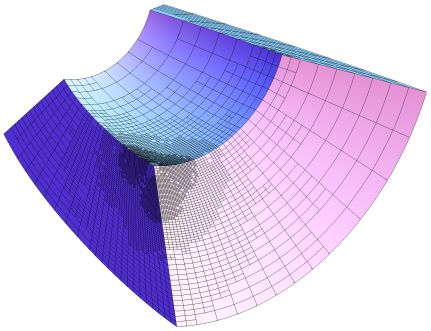
$$u(x, y, z) = \sin\left(\frac{1}{\alpha + (x^2 + y^2 + z^2)}\right),$$

where the parameter $\alpha = 1/10\pi$. This is a smooth function with an oscillation. The center of oscillation is situated on the boundary of the computational domain, see Fig. 7.3(b).

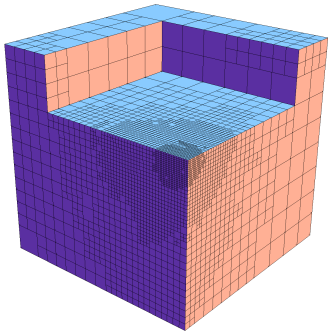
Final mesh of the adaptive refinement is shown in Fig. 7.2 and the corresponding numerical solution is shown in Fig. 7.3. The estimator has refined the



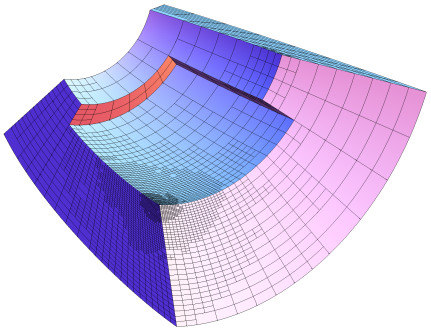
(a) Final adapted parametric mesh.



(b) Physical mesh.



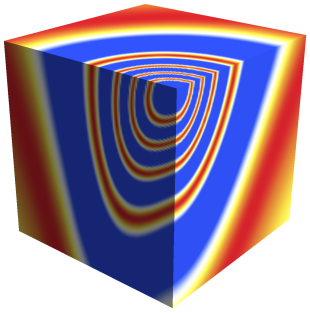
(c) A section of the parametric mesh.



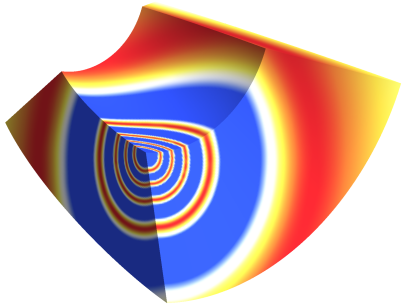
(d) A section of the physical mesh.

**Figure 7.2:** *Final mesh of the adaptive refinement for 3D Poisson problem in the sphere portion.*

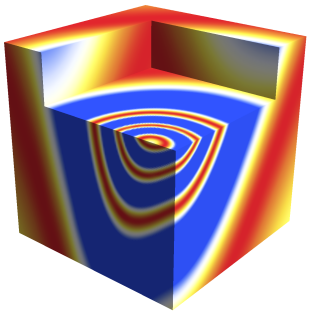
expected zone near the center of the oscillation. The convergence behaviour for the uniform and adaptive refinement are given in Fig. 7.3(e). Due to the regularity of the solution, optimal rates of convergence are obtained for both adaptive and uniform refinement.



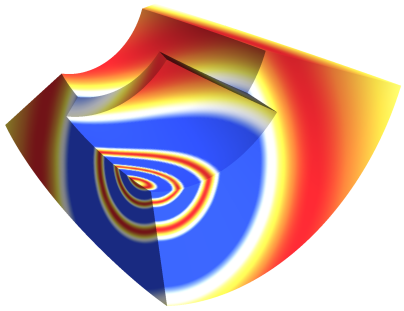
(a) Parametric domain.



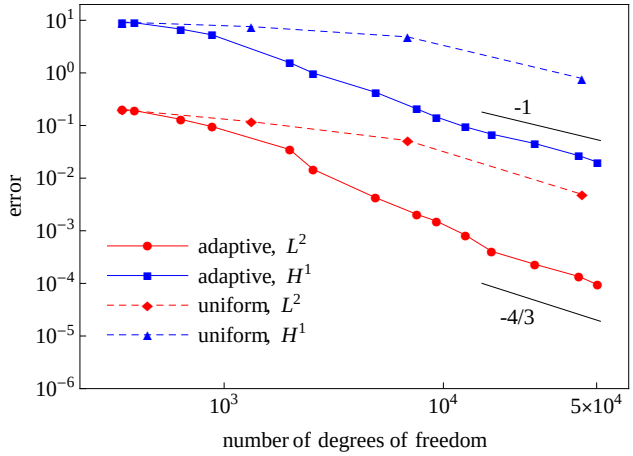
(b) Physical domain.



(c) A section of the parametric domain.



(d) A section of the physical domain.




(e) Convergence of $L^2$-norm and $H^1$-seminorm error for the adaptive and uniform refinement.

**Figure 7.3:** *Numerical solution of the 3D Poisson problem in the sphere portion domain.*

### 7.3.2 Poisson problem in a complex domain

Here we solve Poisson equation in the geometry designed in *Rhinoceros* CAD software and shown in Section 6.3. Its volumetric parameterization was obtained using our method exposed in Chapter 6. Spline representation of the geometry is constructed using a uniform $8 \times 8 \times 8$ hexahedral mesh, which is sufficient to represent well all the features of the geometry. The quality (mean ratio Jacobian) of the parametric mapping is shown in Fig. 7.4.

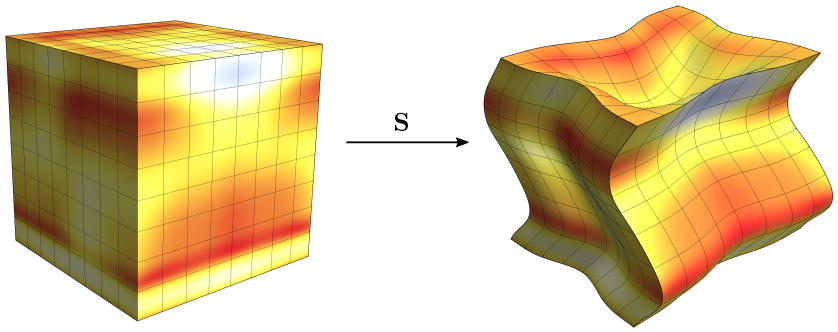


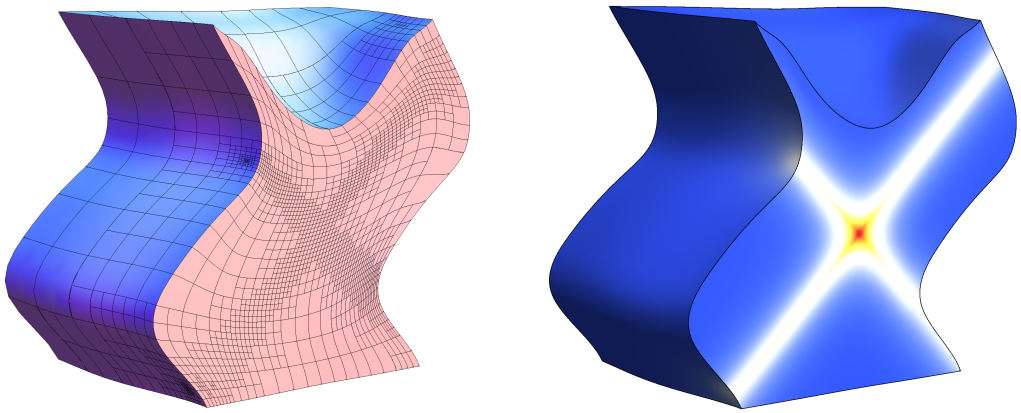

**Figure 7.4:** *Volumetric parameterization of the geometry designed in Rhinoceros. Colormap of mean ratio Jacobian for the parametric mapping* **S**.

The boundary value problem with Dirichlet boundary conditions is set up in such a way that its analytical solution is a function given by

$$u(x,y,z) = \exp\left(-\alpha\sqrt{(y-y_0-z+z_0)^2+(x-x_0)^2}\right)+ \\ +\exp\left(-\alpha\sqrt{(y-y_0+z-z_0)^2+(x-x_0)^2}\right).$$

This function has singularities along the two lines:
$\{x = x_0, y - y_0 = z - z_0\}$ and $\{x = x_0, y - y_0 = -(z - z_0)\}$, that intersect at the point $(x_0, y_0, z_0)$ of the domain. The gradient of $u$ is discontinuous along the lines and the source-term $f$ of the Poisson problem tends to infinity at the points of the lines. Here we take $\alpha = 20$, which make the function $u$ very "concentrated" near the singularity lines.

The final mesh of the adaptive refinement and the corresponding solution are shown in Fig. 7.5. As expected, the error estimator marked to refine the cells along the singularity lines. The convergence behaviour for the uniform and adaptive refinements are illustrated in Fig. 7.6. Suboptimal order of convergence is obtained for the uniform refinement due to the presence of singularities, and the adaptive refinement gains optimal convergence rates.



(a) Final refinement step.

(b) Numerical solution corresponding to the final refinement.

**Figure 7.5:** *Poisson problem in a complex domain designed in Rhinoceros. A section of the physical domain.*

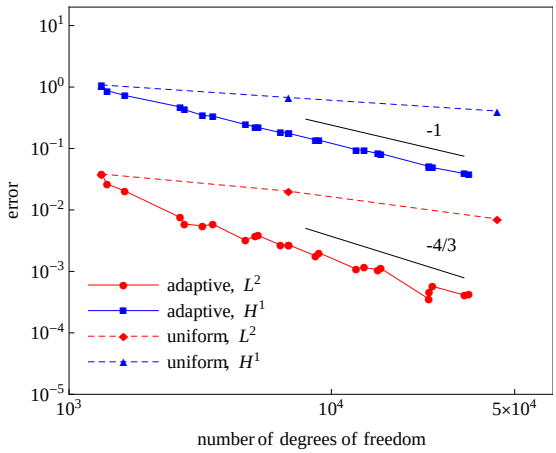



**Figure 7.6:** *Error convergence in $L^2$-norm and $H^1$-seminorm for the Poisson problem in the domain designed in Rhinoceros.*

### 7.3.3 Helmholtz equation with variable frequency

This test problem was taken from [71]. The problem was inspired by the wave function that satisfies a Schrödinger equation model of two interacting atoms. We have Helmholtz equation

$$\begin{cases} -\Delta u - \dfrac{1}{(\alpha + r)^4} u = f & \text{in } \Omega, \\ u = g & \text{on } \partial\Omega, \end{cases} \tag{7.15}$$

where $r = \sqrt{(x-x_0)^2 + (y-y_0)^2 + (z-z_0)^2}$ and $f = \dfrac{2\alpha \cos \frac{1}{\alpha+r}}{(\alpha+r)^3 r}$.
The exact solution of the problem is given by

$$u(r) = \sin\left(\frac{1}{\alpha+r}\right). \tag{7.16}$$

The function (7.16) is highly oscillatory near the point $(x_0, y_0, z_0)$ with an increasing wave frequency. The number of oscillations is determined by the parameter $\alpha$. Besides, it has discontinuous gradient at the point $(x_0, y_0, z_0)$.

The variational formulation of the problem consists in finding $u \in V_g(\Omega)$ such that

$$a(u, v) = F(v) \quad \forall v \in V_0,$$

where

$$a(u, v) = \int_\Omega \left(\nabla u \cdot \nabla v + k(r)^2 u\, v\right)\, \mathrm{d}\Omega \quad \text{and} \quad F(v) = \int_\Omega f\, v\, \mathrm{d}\Omega,$$

being $k(r) = \dfrac{1}{(\alpha+r)^2}$. Discrete weak formulation leads to the linear system

$$(\mathbf{K} + \mathbf{M})\, \mathbf{c} = \mathbf{f}, \qquad \text{where}$$

$\mathbf{K}_{j,i} = \int_\Omega \nabla N_i \cdot \nabla N_j \, \mathrm{d}\Omega$ is the stiffness matrix,
$\mathbf{M}_{j,i} = \int_\Omega k(r)^2\, N_i\, N_j \, \mathrm{d}\Omega$ is the mass matrix and
$\mathbf{f}_j = F(N_j) - a(g_h, N_j)$ is the load vector.

We solve the problem for the domain $\Omega = [0,1]^3$ with $x_0 = y_0 = z_0 = 0.5$ and $\alpha = 1/(5\pi)$. Figure 7.7 shows the analytical solution of the problem along the $z$-direction with $x = y = 0.5$.

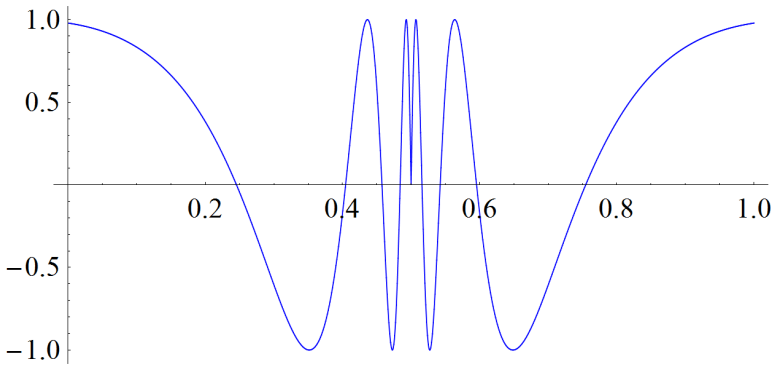
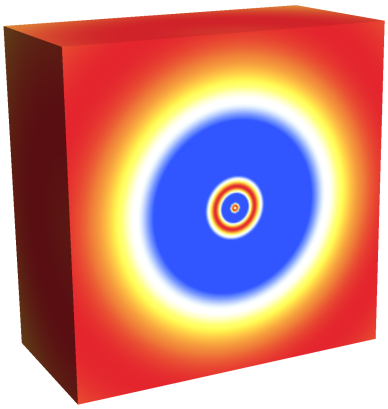
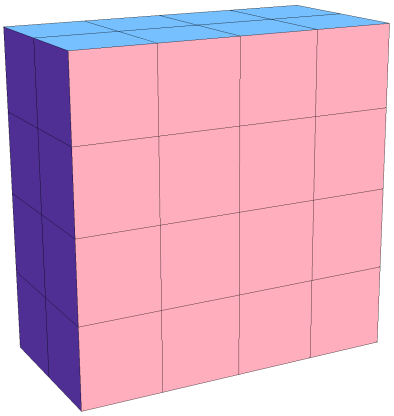



**Figure 7.7:** *Exact solution (7.16) for Helmholtz equation along the z-direction with* $x = y = 0.5$.

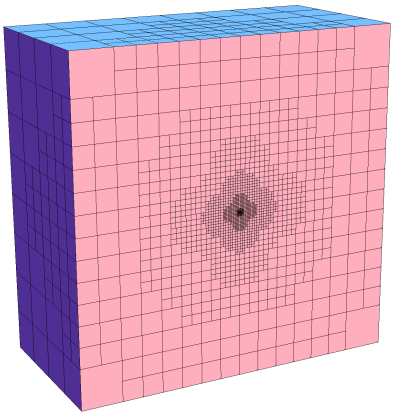
The adaptive refinement is performed, as usually, according to residual-type error estimator. The initial and the final adapted meshes are shown in Fig. 7.8. Error evolution for the adaptive and uniform refinement and its comparison with FEM results are shown in Fig. 7.9. An increase in the error is observed for Finite Element Analysis in the first three iterations for the adaptive and uniform refinement, see Fig. 7.9(b) and (c). This can be due to the fact that for a stable solution of a Helmholtz equation $kh$ should be sufficiently small, where $k$ is a wave number and $h$ is a mesh size. For Helmholtz equation the $H^1$ error is bounded by $C_1kh+C_2k^3h^2$, where for large $k$ the term $k^3h^2$ is leading [72]. Although this pollution effect cannot be avoided completely for two and more space dimensions [73], it can be reduced to minimal using Generalized Finite Element Method (GFEM) for the Helmholtz equation [74], which consists in enriching the Finite Element space via the Partition of Unity Method to create the GFEM space. However, for this particular case, no such phenomenon is observed for IGA results. Only a slight increase of the error takes place for the fourth iteration step during the adaptive and uniform refinement, see Fig. 7.9(a). IGA gives a better accuracy per degree of freedom for the adaptive and uniform refinement. Both IGA and FEM gained the optimal rates of convergence in asymptotic range of the adaptive refinement.



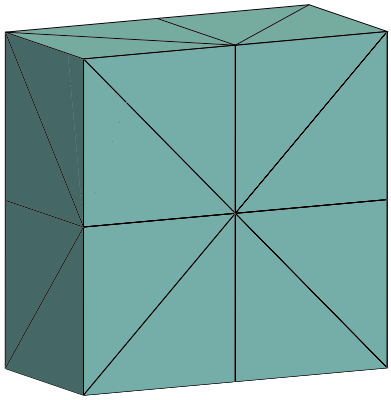
(a) IGA solution.



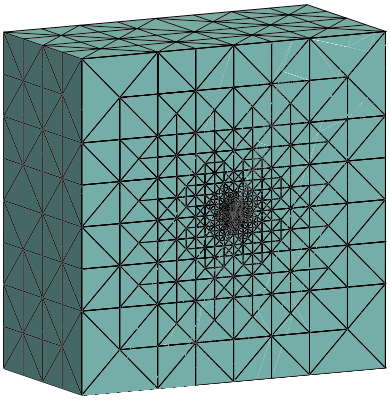
(b) Initial mesh, IGA.



(c) Final mesh, IGA.

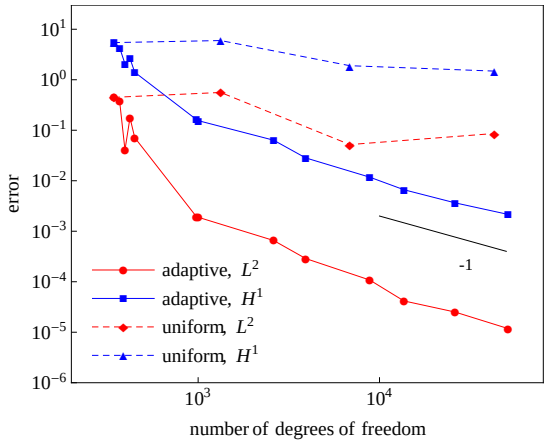


(d) Initial mesh, FEM.

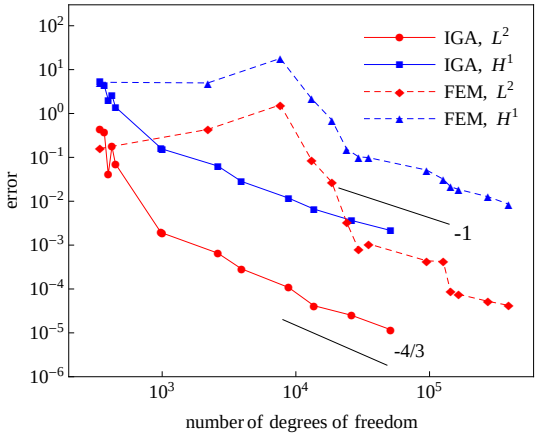


(e) Final mesh, FEM.

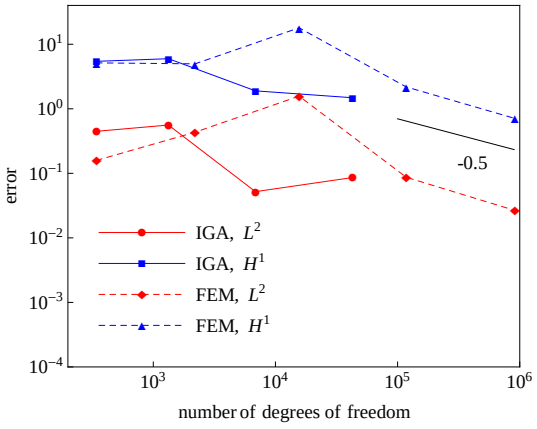
**Figure 7.8:** *Adaptive IGA for Helmholtz equation. (a) Solution of the problem on the section of the computational domain $x = x_0$. (b)-(c) Initial and final meshes for adaptive IGA. (d)-(e) Initial and final meshes for adaptive FEM.*




(a) Uniform and adaptive refinement for IGA.




(b) Adaptive refinement for IGA and FEM.




(c) Uniform refinement for IGA and FEM.

**Figure 7.9:** *Error convergence of the adaptive IGA for Helmholtz equation and its comparison with FEM.*

# CHAPTER 8

## Conclusions and future works

## 8.1 Summary and conclusions

In this thesis we have proposed a strategy for constructing spline spaces for their use in Isogeometric Analysis and CAD. The method permits to define easily $C^2$-continuous cubic spline functions over hierarchical quadtree and octree T-meshes. We only demand these meshes to be strongly balanced and this requirement can be easily satisfied by using a standard balancing procedure. The proposed strategy includes simple rules for inferring local knot vectors to define blending functions for a given T-mesh. The designed technique was motivated by the inefficiency of the tensor product B-splines to carry out a local refinement, which complicates their use in the analysis. Although currently there a few methods available to perform local refinements for spline spaces, all of them have some shortcomings, which was discussed in Chapter 3. Moreover, for some of them (analysis-suitable T-splines), it is not clear its extension to 3D. Our goal was to develop an alternative method for constructing spline spaces with all the properties necessary for analysis. In contrast to other techniques, we propose a simple, easy-to-implement method with low computational cost both for 2D and 3D cases. EP-spline spaces have the following characteristics:

- EP-splines are polynomials, linearly independent and $C^2$-continuous;
- nesting behaviour: spaces spanned by nested T-meshes are also nested;
- polynomial reproduction property;
- non-negative partition of unity;
- straightforward implementation, both for 2D and 3D;
- possibility to define a space for a given quadtree/octree T-mesh;
- the functions are defined independently from each other, so the process can be parallelized;
- each node of the mesh has only one basis function assigned, which facilitates

the interpolation and application of collocation method.

Thesis includes the following main parts:

- description of the rules used for inferring local knot vectors and implementation algorithms to define EP-splines in 2D and 3D;
- discussion and analysis of the properties and characteristics of the constructed spaces;
- proof of the nesting behaviour of spline spaces and the non-negative partition of unity for 2D;
- testing of the EP-spline functions for their use in geometric modelling and Isogeometric Analysis for problems involving adaptive refinement.

Approximation properties of the constructed spline spaces are tested in different type of problems with singularities, where an adaptive refinement is necessary to obtain an accurate numerical solution. In all the computational examples the error estimator refined the expected zone and optimal rates of convergence were obtained. Some examples include a comparison with the Finite Element Method, showing a slightly superior accuracy of IGA for some problems and a significant difference in others (eigenvalue problem, Kellog function, Helmholtz equation). We believe that the simplicity and the good approximation properties of EP-splines make it an attractive tool for IGA and CAD.

Besides, some preliminary results about a parameterization method developed in our research, have been briefly exposed in this thesis. The method is based on a novel T-mesh optimization procedure and obtains, as a result, a high quality parametric transformation between the object and the parametric domain. This technique was used for the parameterization of some complex domains in computational examples.

## 8.2 Future work

Regarding the proposed spline spaces, the following future works should be mentioned:

- First of all, a rigorous proof of the properties of EP-spline spaces is needed: nesting behaviour of the spaces for 3D case and linear independence for 2D and 3D cases.
- The proposed spline spaces possess the essential properties for their use in analysis; however, some improvements can be done. Namely, as was mentioned in Section 5.5, an excessive support overlapping can take place during the adaptive refinement for some problems, which can lead to increase of

the density of the matrix and worsen its condition number. Although this situation does not happen for most problems, it is preferable to deal with functions with more local supports. We believe that it is possible to obtain a little more selective rules for inferring local knot vectors to avoid an unnecessary support extension in some cases. On the other hand, more careful and selective inferring of spline basis is also necessary to obtain a positive partition of unity without the presence of redundant functions.

- Reduced function continuity in a specific zone can be beneficial for some problems, as was illustrated in Section 7.2.8. So, it is necessary to extend the method developed in this thesis for the parametric spaces with repeated interior knots. Also, it would be interesting to extend the strategy for higher order spline functions.

- In this work we define spline spaces over a square parametric domain. However, to facilitate the parameterization of very complex geometries it is essential to work with more complex, polygon type, parametric domains, that fit better the input geometry. This would allow to reduce the high distortion that appears when a square parametric domain is used to parametrize a geometry very different from a square. Hence, it is necessary to elaborate a procedure to construct spline spaces over this type of parametric domains, trying to conserve the regularity of the basis functions.

In a more general context of the Isogeometric Analysis paradigm, we consider it interesting to explore the following questions:

- IGA exhibits some advantages and a superior accuracy per degree of freedom over the Finite Element Method for some problems. However, as for any other technique, there are cases where, probably, it is not the best choice. It would be desirable to study and classify the problems where Isogeometric Analysis can offer a considerable advantages and the problems where it is not so significant, specially taking into account the efficiency of both methods.

- Also, we would like to explore the possibility to construct spline spaces over triangulations of computational domain. Complex domain parameterization using structured hexahedral meshes has considerable limitations. Splines over triangulation can offer more flexibility compared with rigid nature of tensor product splines. However, it is not a trivial task. Currently available options presents a high complexity. There is a necessity of a simple and easy-to-use scheme for constructing globally smooth spline functions over triangulations, able to compete with tensor product splines.

# List of Figures

# Bibliography


[1] L. Piegl, W. Tiller, The NURBS book, Springer, New York, 1997.

[2] G. Farin, Curves and Surfaces for CAGD: a Practical Guide, 5th Edition, Morgan Kaufmann Publishers, San Francisco, CA, USA, 1999.

[3] T. J. R. Hughes, J. A. Cottrell, Y. Bazilevs, Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement, Comput. Meth. Appl. Mech. Eng. 194 (2005) 4135–4195.

[4] J. Cottrell, A. Reali, Y. Bazilevs, T. Hughes, Isogeometric analysis of structural vibrations, Comput. Meth. Appl. Mech. Eng. 195, Issues 4143 (2006) 5257–5296.

[5] H. Gómez, V. Calo, Y. Bazilevs, T. Hughes, Isogeometric analysis of the Cahn–Hilliard phase-field model, Comput. Meth. Appl. Mech. Eng. 197, Issues 4950 (2008) 4333–4352.

[6] H. Gómez, T. J. Hughes, X. Nogueira, V. M. Calo, Isogeometric analysis of the isothermal Navier–Stokes–Korteweg equations, Comput. Meth. Appl. Mech. Eng. 199, Issues 2528 (2010) 1828–1840.

[7] J. Liu, L. Dede, J. A. Evans, M. J. Borden, T. Hughes, Isogeometric analysis of the advective Cahn–Hilliard equation: Spinodal decomposition under shear flow, Journal of Computational Physics 242 (2013) 321–350.

[8] M. Kästner, P. Metsch, R. de Borst, Isogeometric analysis of the Cahn–Hilliard equation – a convergence study, Journal of Computational Physics 305 (2016) 360–371.

[9] F. Auricchio, L. B. D. Veiga, T. J. R. Hughes, A. Reali, G. Sangalli, Isogeometric collocation methods, Mathematical Models and Methods in Applied Sciences Vol. 20, No. 11 (2010) 2075–2107.

[10] D. Schillinger, J. A. Evans, A. Reali, M. A. Scott, T. J. Hughes, Isogeometric collocation: Cost comparison with Galerkin methods and extension to adaptive hierarchical NURBS discretizations, Comput. Meth. Appl. Mech. Eng. 267 (2013) 170–232.

[11] H. Casquero, L. Liu, Y. Zhang, A. Reali, H. Gómez, Isogeometric collocation using analysis-suitable T-splines of arbitrary degree, Comput. Meth. Appl. Mech. Eng. 301 (2016) 164–186.

[12] C. Anitescu, Y. Jia, Y. J. Zhang, T. Rabczuk, An isogeometric collocation method using superconvergent points, Comput. Meth. Appl. Mech. Eng. 284 (2015) 1073–1097.

[13] Y. Bazilevs, L. Beirão Da Veiga, J. Cottrell, T. Hughes, G. Sangalli, Isogeometric analysis: Approximation, stability and error estimates for h-refined meshes, Mathematical Models and Methods in Applied Sciences 16 (7) (2006) 1031–1090.

[14] L. Beirão da Veiga, A. Buffa, R. J., G. Sangalli, Some estimates for h-p-k-refinement in Isogeometric Analysis, Numer. Math. 118 (2) (2011) 271–305.

[15] A. Tagliabue, L. Dedé, A. Quarteroni, Isogeometric Analysis and error estimates for high order partial differential equations in fluid dynamics, Computers and Fluids 102 (2014) 277–303.

[16] T. J. R. Hughes, A. Reali, G. Sangalli, Efficient quadrature for NURBS-based isogeometric analysis, Comput. Meth. Appl. Mech. Eng. 199, 5-8 (2010) 301–313.

[17] F. Auricchio, F. Calabró, T. J. R. Hughes, A. Reali, G. Sangalli, A simple algorithm for obtaining nearly optimal quadrature rules for NURBS-based isogeometric analysis, Comput. Meth. Appl. Mech. Eng. 249-252 (2012) 15–27.

[18] N. O. Collier, L. Dalcín, D. Pardo, V. M. Calo, The cost of continuity: performance of iterative solvers on isogeometric finite elements, SIAM Journal on Scientific Computing 35(2) (2012) A767–A784.

[19] N. O. Collier, D. Pardo, L. Dalcín, M. Paszynski, V. M. Calo, The cost of continuity: A study of the performance of isogeometric finite elements using direct solvers, Comput. Meth. Appl. Mech. Eng. 213-216 (2013) 353–361.

[20] S. Lipton, J. A. Evans, Y. Bazilevs, T. Elguedj, T. J. R. Hughes, Robustness of isogeometric structural discretizations under severe mesh distortion, Comput. Meth. Appl. Mech. Eng. 199 (2010) 357–373.

[21] T. W. Sederberg, J. Zheng, A. Bakenov, A. Nasri, T-splines and T-NURCCSs, ACM Trans. Graph. 22 (2003) 477–484.

[22] T. W. Sederberg, D. L. Cardon, G. T. Finnigan, J. Zheng, T. Lyche, T-spline simplification and local refinement, in: ACM Transactions on Graphics. Proceedings of ACM SIGGRAPH 2004 . Volume 23, Issue 3. Pages 276–283.

[23] M. A. Scott, X. Li, T. W. Sederberg, T. J. R. Hughes, Local refinement of analysis-suitable T-splines, Comput. Meth. Appl. Mech. Eng. 213-216 (2012) 206–222.

[24] X. Li, J. Zheng, T. W. Sederberg, T. J. R. Hughes, M. A. Scott, On linear independence of T-spline blending functions, Comput. Aid. Geom. Design 29 (2012) 63–76.

[25] D. R. Forsey, R. H. Bartels, Hierarchical B-spline refinement, Computer Graphics 22(4) (1988) 205–212.

[26] R. Kraft, Surface Fitting and Multiresolution Methods, Vol. 2, Vanderbilt University Press, 1997, Ch. Adaptive and linearly independent multilevel B-splines, pp. 209–218.

[27] A. V. Vuong, C. Giannelli, B. Jüttler, B. Simeon, A hierarchical approach to adaptive local refinement in isogeometric analysis, Comput. Meth. Appl. Mech. Eng. 200 (2011) 3554–3567.

[28] D. Schillinger, L. Debé, M. Scott, J. A. Evans, M. J. Borden, E. Rank, T. J. R. Hughes, An isogeometric design-through-analysis methodology based on adaptive hierarchical refinement of nurbs, immersed boundary methods, and T-spline CAD surfaces, Comput. Meth. Appl. Mech. Eng. 249–252 (2012) 116–150.

[29] P. B. Bornemann, F. Cirak, A subdivision-based implementation of the hierarchical B-spline finite element method, Comput. Meth. Appl. Mech. Eng. 253 (2013) 584–598.

[30] C. Giannelli, B. Jüttler, H. Speleers, THB-splines: The truncated basis for hierarchical splines, Computer Aided Geometric Design 29 (2012) 485–498.

[31] J. Deng, F. Chen, X. Li, C. Hu, W. Tong, Z. Yang, Y. Feng, Polynomial splines over hierarchical T-meshes, Graphical Models 70 (2008) 76–86.

[32] T. Dokken, T. Lyche, K. F. Pettersen, Polynomial splines over locally refined box-partitions, Comput. Aid. Geom. Design 30(3) (2013) 331–356.

[33] C. de Boor, Splines as linear combinations of B-splines. A Survey (1976).

[34] M. Neamtu, Mathematical Methods for Curves and Surfaces, Vanderbilt University, Nashville, TN, USA, 2001, Ch. What is the Natural Generalization of Univariate Splines to Higher Dimensions?, pp. 355–392.

[35] M. Neamtu, Delaunay Configurations and Multivariate Splines: A Generalization of a Result of B. N. Delaunay, Transactions of the American Mathematical Society 359, No. 7 (2007) 2993–3004.

[36] M. Neamtu, Bivariate simplex B-Splines: A New Paradigm, in: In SCCG 01: Proceedings of the 17th Spring conference on Computer graphics, 2001, pp. 71–78.

[37] C. de Boor, K. Höllig, S. Riemenschneider, Box Splines, Springer-Verlag New York, Inc., New York, NY, USA, 1993.

[38] M. J. D. Powell, M. A. Sabin, Piecewise quadratic approximations on triangles, ACM Trans. Math. Softw. 3 (4) (1977) 316–325.

[39] H. Speleers, P. Dierckx, S. Vandewalle, Numerical solution of partial differential equations with powell-sabin splines, J. Comput. Appl. Math. 189 (1-2) (2006) 643–659.

[40] H. Speleers, C. Manni, F. Pelosi, From NURBS to NURPS geometries, Computer Methods in Applied Mechanics and Engineering 255 (2013) 238–254.

[41] N. Jaxon, X. Qian, Isogeometric analysis on triangulations., Computer-Aided Design 46 (2014) 45–57.

[42] J. A. Cottrell, T. J. R. Hughes, Y. Bazilevs, Isogeometric analysis: Toward integration of CAD and FEA, 1st Edition, Wiley Publishing, 2009.

[43] G. Xu, B. Mourrain, R. Duvigneau, A. Galligo, Parametrization of computational domain in isogeometric analysis: Methods and comparison, Comput. Meth. Appl. Mech. Eng. 200 (2011) 2021–2031.

[44] G. Xu, B. Mourrain, R. Duvigneau, A. Galligo, Optimal analysis-aware parameterization of computational domain in 3D isogeometric analysis, Comput. Aided Des. 45 (4) (2013) 812–821.

[45] G. Xu, B. Mourrain, R. Duvigneau, A. Galligo, Variational harmonic method for parameterization of computational domain in 2D isogeometric analysis, in: 12th International Conference on Computer-Aided Design and Computer Graphics, IEEE, Jinan, 2011, pp. 223–228.

[46] G. Xu, B. Mourrain, R. Duvigneau, A. Galligo, Constructing analysis-suitable parameterization of computational domain from CAD boundary by variational harmonic method, Journal of Computational Physics 252 (2013) 275–289.

[47] X. Li, X. Guo, H. Wang, Y. He, X. Gu, H. Qin, Harmonic volumetric mapping for solid modeling applications, in: Proc. of ACM Solid and Physical Modeling Symposium, Association for Computing Machinery, Inc., 2007, pp. 109–120.

[48] T. Martin, E. Cohen, R. Kirby, Volumetric parameterization and trivariate B-spline fitting using harmonic functions, Comput. Aid. Geom. Design 26 (2009) 648–664.

[49] Y. Zhang, W. Wang, T. J. R. Hughes, Solid T-spline construction from boundary representations for genus-zero geometry, Comput. Meth. Appl. Mech. Eng. 249-252 (2012) 185–197.

[50] W. Wang, Y. Zhang, L. Liu, T. J. R. Hughes, Trivariate solid T-spline construction from boundary triangulations with arbitrary genus topology, Computer-Aided Design 45 (2013) 351–360.

[51] H. Samet, Foundations of Multidimensional and Metric Data Structures, Morgan Kaufmann Publishers, Burlington, Massachusetts, 2006.

[52] C. De Boor, A practical guide to splines, Applied mathematical sciences, Springer-Verlag, New York, 2001.

[53] A. Buffa, D. Cho, G. Sangalli, Linear independence of the T-spline blending functions associated with some particular T-meshes, Comput. Meth. Appl. Mech. Eng. 199 (2010) 1437–1445.

[54] A. Wang, G. Zhao, Y. Li, Linear independence of the blending functions of T-splines without multiple knots, Expert Systems with Applications 41 (2014) 3634–3639.

[55] L. A. Piegl, W. Tiller, K. Rajab, It is time to drop the "r" from nurbs, Eng. with Comput. 30 (4) (2014) 703–714.

[56] D. Moore, The cost of balancing generalized quadtrees, in: Proceedings of the Third ACM Symposium on Solid Modeling and Applications, SMA '95, ACM, New York, NY, USA, 1995, pp. 305–312. doi:10.1145/218013.218078.

[57] C. Giannelli, B. Jüttler, H. Speleers, Strongly stable bases for adaptively refined multilevel spline spaces, Adv. Comp. Math. 40 (2014) 459–490.

[58] J. M. Escobar, J. M. Cascón, E. Rodríguez, R. Montenegro, A new approach to solid modeling with trivariate T-splines based on mesh optimization, Comput. Meth. Appl. Mech. Eng. 200 (2011) 3210–3222.

[59] J. M. Escobar, R. Montenegro, E. Rodríguez, J. M. Cascón, The meccano method for isogeometric solid modeling and applications, Engineering with Computers 30 (3) (2014) 331–343.

[60] R. Montenegro, J. M. Cascón, E. Rodríguez, J. M. Escobar, G. Montero, The meccano method for automatic three-dimensional triangulation and volume parametrization of complex solids, in: Developments and Applications in Engineering Computational Technology, Saxe-Coburg Publications, Stirling, 2010, pp. 19–48.

[61] J. M. Escobar, E. Rodríguez, R. Montenegro, G. Montero, J. M. González-Yuste, Simultaneous untangling and smoothing of tetrahedral meshes, Comput. Meth. Appl. Mech. Eng. 192 (2003) 2775–2787.

[62] M. S. Floater, Parametrization and smooth approximation of surface triangulations, Comput. Aid. Geom. Design 14 (1997) 231–250.

[63] M. Brovka, J. I. López, J. M. Escobar, J. M. Cascón, R. Montenegro, A new method for T-spline parameterization of complex 2D geometries, Engineering with Computers 30 (4) (2014) 457–473.

[64] P. M. Knupp, Algebraic mesh quality metrics, SIAM J. Sci. Comput. 23 (2001) 193–218.

[65] P. M. Knupp, A method for hexahedral mesh shape optimization, Int. J. Num. Meth. Eng. 58 (2) (2003) 319–332.

[66] J. I. López, M. Brovka, J. M. Escobar, R. Montenegro, G. V. Socorro, Strategies for optimization of hexahedral meshes and their comparative study, Engineering with Computers (available online May 2016) 1–11.

[67] A. Schmidt, K. Siebert, Design of Adaptive Finite Element Software. The Finite Element Toolbox ALBERTA, Springer Springer-Verlag Berlin Heidelberg, 2005.

[68] I. Kossaczky, A recursive approach to local mesh refinement in two and three dimensions, J. Comput. Appl. Math. 55 (1994) 275–288.

[69] A. Tagliabue, L. Dedé, A. Quarteroni, Isogeometric Analysis and error estimates for high order partial differential equations in fluid dynamics, Computers & Fluids 102 (2014) 277–303.

[70] J. M. Cascón, Estimación a-posteriori del error y adaptación de mallado para formulaciones mixtas de problemas elípticos y parabólicos, Ph.D. thesis, Universidad de Salamanca (2004).

[71] W. F. Mitchell, A Collection of 2D Elliptic Problems for Testing Adaptive Algorithms, Mathematical and Computational Sciences Division National Institute of Standards and Technology.

[72] F. Ihlenburg, I. Babuška, Finite element solution of the Helmholtz equation with high wave number Part I: The h-version of the FEM, Computers & Mathematics with Applications 30 (9) (1995) 9–37.

[73] I. M. Babuška, S. A. Sauter, Is the Pollution Effect of the FEM Avoidable for the Helmholtz Equation Considering High Wave Numbers?, SIAM Rev. 42 (3) (2000) 451–484.

[74] I. Babuška, F. Ihlenburg, E. T. Paik, S. A. Sauter, A Generalized Finite Element Method for solving the Helmholtz equation in two dimensions with minimal pollution, Comput. Meth. Appl. Mech. Eng. 128 (3-4) (1995) 325–359.

# Appendices

# APPENDIX A

## Published works derived from Ph. D. Thesis

### Publications

- A new method for T-spline parameterization of complex 2D geometries.
  M. Brovka, J.I. López, J.M. Escobar, J.M. Cascón, R. Montenegro.
  *Engineering with Computers*, 30 (4), 2013, pp. 457–473.

- A simple strategy for defining polynomial spline spaces over hierarchical T-meshes.
  M. Brovka, J.I.López, J.M. Escobar, R. Montenegro, J.M. Cascón.
  *Computer-Aided Design*, 72, 2016, pp. 140–156.

- Strategies for optimization of hexahedral meshes and their comparative study.
  J.I. López, M. Brovka, J.M. Escobar, R. Montenegro and G. V. Socorro.
  *Engineering with Computers*, available online May 2016, DOI: 10.1007/s00366-016-0454-1.

### Conference contributions

- Some advances in open problems of isogeometric analysis.
  M. Brovka, J.I. López, R. Montenegro, J.M. Escobar.
  *Proceedings of the XXIV Congress on Differential Equations and Applications/ XIV Congress on Applied Mathematics*, 8-12 July, 2015, Cadiz, Spain, pp. 815–820.

- Construction of polynomial spline spaces over quadtree and octree T-meshes.
  M. Brovka, J.I. López, J.M. Escobar, J.M. Cascón, R. Montenegro.
  *Proceedings of the 23nd International Meshing Roundtable*, 12-15 October, 2014, London, UK, pp. 21–33.

- Advances on T-Spline parametrization based on the Meccano Method.
  J.I. López, M. Brovka, J.M. Escobar, J.M. Cascón R. Montenegro.

*Proceedings del 11th World Congress on Computational Mechanics (WCCM XI), 5th European Conference on Computational Mechanics (ECCM V) y 6th European Conference on Computational Fluid Dynamics (ECFD VI)*, 20-25 July, 2014, Barcelona, Spain.

- T-spline parameterization of 2D geometries based on the meccano method with a new T-mesh optimization algorithm.
  J.I. López, M. Brovka, J.M. Escobar, J.M. Cascón, R. Montenegro.
  *Proceedings of the 22nd International Meshing Roundtable*, Springer, 13-16 October, 2013, Orlando, USA, pp. 57–74.

- Extension del Método del Mecano para Análisis Isogeométrico con T-splines.
  M. Brovka, J.I. López, J.Ramírez, R. Montenegro, J.M. Escobar, J.M. Cascón, E. Rodríguez.
  *Congress on Numerical Methods in Engineering*, 25-28 June, 2013, Bilbao, Spain.

- Application to Complex Solids of Adaptive Isogeometric Analysis using T-splines.
  R. Montenegro, J.M. Cascón, E. Rodríguez, J.M. Escobar, M. Brovka, J.I. López, J. Ramírez.
  *10th WCCM*, 8-13 July, 2012, Sao Paulo, Brazil.

# APPENDIX B

Summary of the dissertation in Spanish

# Construcción de espacios spline polinómicos sobre T-meshes para su aplicación en Análisis Isogeométrico

**Construcción de espacios spline polinómicos sobre T-meshes para su aplicación en Análisis Isogeométrico**

## Resumen

En esta tesis se aborda la construcción de espacios spline para su utilización en el Análisis Isogeométrico. Nuestro trabajo está motivado por la incapacidad inherente de los espacios B-spline con estructura tensorial para realizar refinamientos locales, lo cual dificulta su uso en el análisis. En esta tesis proponemos una nueva forma de definir funciones spline cúbicas capaces de generar espacios con propiedades adecuadas para el análisis. Nuestra estrategia está específicamente diseñada para T-meshes jerárquicas generadas a partir de subdivisiones tipo quadtree y octree. Esta clase de mallas pueden ser implementadas de manera muy eficiente con estructuras de datos de tipo árbol, frecuentemente utilizadas en ingeniería.

La construcción práctica de espacios spline sobre T-meshes es actualmente un problema abierto, especialmente en 3D. Por ello, uno de nuestros principales objetivos consiste en diseñar una estrategia sencilla y con un bajo coste computacional para la generación de espacios spline sobre T-meshes, tanto en 2D como en 3D. Nuestro método permite definir funciones spline cúbicas capaces de generar espacios con propiedades adecuadas para el análisis: independencia lineal de las funciones, continuidad $C^2$, encaje de espacios y, además, facilidad de implementación. Los knot vectors locales de nuestras funciones spline se infieren a partir de un conjunto de reglas sencillas definidas para una T-mesh dada, con el único requerimiento de ser 0-balanceada.

En esta tesis damos una descripción detallada del procedimiento utilizado para la construcción de los espacios spline y analizamos sus propiedades de aproximación sobre diferentes problemas, donde es necesario un refinamiento adaptativo para conseguir una alta precisión de la solución numérica.

# Índice general

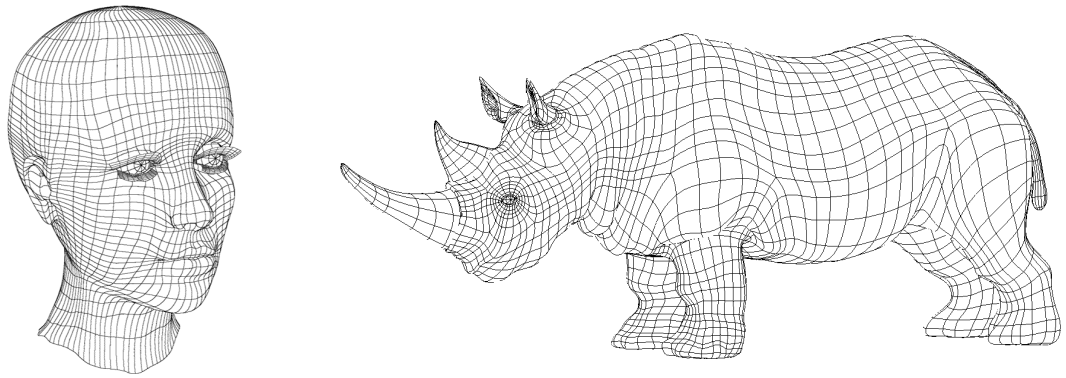
# CAPÍTULO 1

# Introducción

## 1.1. Conceptos generales sobre Análisis Isogeométrico

El diseño asistido por ordenador o CAD (Computer-Aided Design), es el proceso de diseño gráfico mediante programas informáticos. El CAD ha reemplazando al lápiz y al papel para crear objetos geométricos de la industria y de la ingeniería. Hoy en día el CAD es una tecnología ampliamente utilizada en numerosas aplicaciones en automoción, construcción naval, industria aeroespacial, diseño industrial y arquitectónico, la animación por ordenador para efectos especiales en películas. La herramienta matemática utilizada más comúnmente para la representación de curvas y superficies en CAD son las B-splines y las NURBS (Non uniform rational B-splines) [1, 2]. Las NURBS tienen la particularidad de ser capaces de representar secciones cónicas de forma exacta y, además, poseen características adicionales interesantes como la alta suavidad de las curvas construidas con estas funciones y la conocida propiedad de la envolvente convexa.

Por otro lado, otra importante técnica computacional que ha sido desarrollada desde 1950-1960 es el Método de Elementos Finitos (FEM). Este método numérico para resolver ecuaciones diferenciales en derivadas parciales ha recibido una gran cantidad de esfuerzo de investigación y se ha convertido en una herramienta estándar en la industria de la ingeniería. Los componentes básicos del método son la formulación variacional del problema físico y la discretización del espacio utilizado para aproximar su solución. El espacio de aproximación se define por sus funciones de base, típicamente de Lagrange o Hermite. Cada función de base se define de forma local en su elemento. Para realizar el análisis, el dominio computacional del problema se debe descomponer en elementos disjuntos simples como tetraedros o hexaedros. Este paso se conoce como proceso de generación de malla. Teniendo en cuenta que hoy en día la geometría está definida a través del CAD, se plantea la necesidad de generar una malla de análisis directamente a partir de su representación CAD. No es una tarea trivial debido a la creciente complejidad exigida en el proceso de diseño de ingeniería. Ambas tecnologías, FEM y CAD, han evolucio-



(a) (b)

**Figura 1.1:** *(a) Cabeza modelada con NURBS, (b) Rinoceronte diseñado con NURBS. Images tomadas de* `http://www.3drender.com/`

nado por separado y utilizan diferentes herramientas geométricas. Esto impide la comunicación entre ellas y hacen que el proceso de transferencia de datos consuma mucho tiempo. La creación de un modelo apto para el análisis y la generación de la malla son los procesos que invierten la mayor cantidad de tiempo. La estrecha comunicación entre el modelo CAD y la geometría FEM también es crucial para hacer posible la optimización del diseño. Además, la malla de elementos finitos no reproduce exactamente el modelo diseñado por el CAD. Para algunos problemas la imprecisión en la geometría puede conducir a un error importante en la solución numérica. Para aproximar una frontera curva con una precisión deseada se requiere un refinamiento adaptativo de la malla. Esto es posible sólo con una interacción automática entre la geometría CAD exacta y la malla de elementos finitos. Con todos estos inconvenientes, se hizo evidente la necesidad de unificar el diseño y análisis en un proceso de ingeniería único. Así, el CAD tiene que proporcionar directamente un modelo geométrico adecuado para el análisis mediante FEM.

La idea fue propuesta en 2005 por Tom Hughes y sus colaboradores [3]. El concepto recibió el nombre de *Isogeometric Analysis* (IGA). La idea es usar para el análisis las mismas funciones de base (originalmente NURBS ) que se han utilizado para la construcción del modelo de CAD. El Análisis Isogeometrico puede ser visto como una generalización de FEM que usa funciones de base (NURBS) de mayor regularidad.

Desde su introducción, IGA ha atraído mucha atención y ha sido objeto de numerosos trabajos de investigación en los últimos años. En la actualidad se considera una herramienta prometedora para cerrar la brecha entre el CAD y el FEM. Además, puede ofrecer algunas propiedades y posibilidades beneficiosas adicionales en comparación con el método de elementos finitos clásico. Estas son algunos de ellas:

- Trabajar con la geometría exacta conduce a una mayor precisión en los problemas sensibles a la aproximación de geometría (problemas de contacto, análisis de la pandeo de lámina, problemas de capa límite en aerodinámica e hidrodinámica).
- Incluso si no se consigue la geometría exacta, la aproximación suave es necesaria para algunos problemas que requieren continuidad de curvatura de la frontera.
- Una mayor precisión de la solución numérica debido a una mayor regularidad de las funciones de base [4].
- El superior grado de continuidad también permite el uso de IGA para resolver ecuaciones diferenciales parciales de orden superior a dos sin la necesidad de cambiar la formulación variacional, como por ejemplo, la ecuación de cambio de fase de Cahn-Hilliard [5, 6, 7, 8].
- La posibilidad de implementar métodos de colocación usando la formulación fuerte [9, 10, 11, 12].

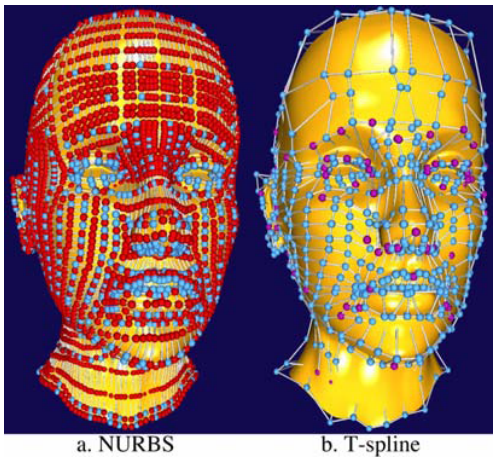
Sin embargo, para llevar a cabo esta ambiciosa idea del Análisis Isogeométrico y hacer que funcione en la práctica, hay que resolver algunas cuestiones abiertas. Esto se analizará en la siguiente sección.

## 1.2. Estado del arte

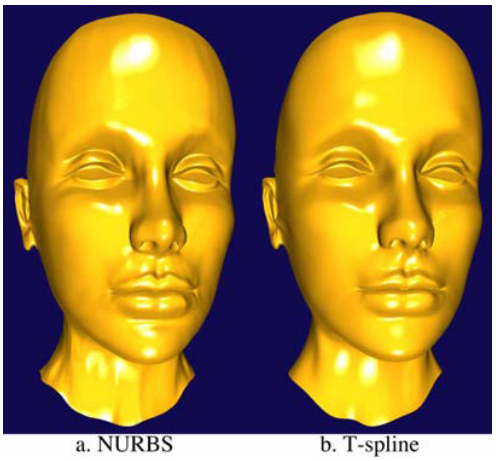
IGA es un método relativamente nuevo que necesita un marco teórico para ser desarrollado. Quedan cuestiones abiertas, como por ejemplo: una teoría de la estimación del error para IGA [13, 14, 15], reglas de cuadratura eficientes [16, 17], *solvers* directos e iterativos para los sistemas que surgen en el método [18, 19], una implementación eficiente de las estructuras y procedimientos que surgen en el método, la imposición de condiciones de contorno, la influencia de la calidad de parametrización en la precisión de la solución [20] . Sin embargo, los problemas más importantes y urgentes del IGA son el refinamiento local y parametrización del dominio computacional.

### 1.2.1. Refinamiento local

Originalmente el concepto de Análisis Isogeométrico fue propuesto y probado usando funciones de base NURBS ya que estas eran la herramienta más común utilizada por el software de CAD en ese momento. Sin embargo, estas funciones tienen un importante inconveniente: su estructura de producto tensorial no permite




(a) *"Head modeled (a) as a NURBS with* 4712 *control points and (b) as a T-spline with* 1109 *control points. The red NURBS control points are superfluous"*




(b) *"NURBS head model, converted to a T-spline"*

**Figura 1.2:** *Imagenes tomadas del trabajo original Sederberg et al. [22]: "T-spline simplification and local refinement".*

el refinamiento local. Una superficie NURBS se define por un conjunto de puntos de control que se encuentran, topologicamente, en una malla rectangular. La inserción de un nuevo punto de control provoca la inserción de toda una fila de puntos de control extendidos por todo el dominio. Esto conduce a un gran número de puntos de control superfluos. La estructura de producto tensorial las hace ineficientes para la representación de características locales requiriendo el uso de varios parches NURBS unidos entre sí. Con frecuencia, esta unión es discontinua, esto es, hay huecos y solapamientos entre diferentes parches que hacen que el modelo geométrico no sea adecuado para el análisis. Las T-splines fueron propuestas por Sederberg et al. [21] como una alternativa a las NURBS que permite el refinamiento local. Las T-splines pueden ser consideradas como una generalización de NURBS donde las filas o las columnas asociadas a los puntos de control pueden quedar truncadas, produciendo lo que se conoce como una T-junction. Esta característica permite el refinamiento local.

Programas de CAD basados en NURBS como Maya y Rhino incorporan aplicaciones (plug-in) de T-splines. Las T-splines ofrecen una herramienta flexible para crear superficies sin huecos ni pliegues capaces de representar características locales detalladas con un menor número de puntos de que las NURBS, ver Fig. 1.2.

Sin embargo, las T-splines inicialmente introducidas para CAD no son adecuadas para su uso en el IGA, ya que carecen de algunas propiedades esenciales para el análisis y la convergencia del IGA, como son la independencia lineal, el encaje de espacios y la consistencia de orden tres sobre todo el dominio (capacidad de representar polinomios cúbicos en todo el dominio). Además, el hecho de ser

funciones racionales complican los cálculos y aumentan el coste computacional.

Por lo tanto, sigue siendo un problema abierto en el contexto del Análisis Isogeométrico encontrar una alternativa a las NURBS que permita refinamiento local y pueda ser utilizado para el análisis. Para que las *blending functions* de las splines puedan ser utilizadas en el análisis numérico y que la convergencia sea adecuada, estas funciones deben cumplir algunos requisitos: independencia lineal, consistencia de orden tres, encaje de espacios de aproximación y la posibilidad de realizar refinamiento local. Este tema ha sido objeto de numerosos trabajos de investigación en los últimos años.

Las T-splines aptas para el análisis (analysis-suitable) propuestas por Scott et al. en [23], son una clase de T-splines definidas sobre T-meshes que cumplen con ciertas restricciones topológicas formuladas en términos de las extensiones de las T-junctions. Las *blending functions* definidas sobre una *analysis-suitable* T-mesh son linealmente independientes [24] y son consistentes. El algoritmo de refinamiento permite lograr mejoras muy localizadas y construir espacios de aproximación anidados, sin embargo presenta una elevada complejidad de implementación y, por lo que sabemos, su generalización a 3D es todavía una cuestión abierta.

Otro enfoque para el problema de enriquecimiento local del espacio de aproximación es el refinamiento jerárquico. Introducido originalmente por Forsey y Bartels en [25] y más tarde desarrollado en [26]. La técnica de refinamiento jerárquico, en el contexto del Análisis Isogeométrico, se describe en [27, 28, 29]. Este enfoque se basa en una idea simple y natural para construir espacios multinivel mediante la sustitución de las funciones de niveles más groseros con funciones de niveles más finos. A partir de una malla inicial uniforme, el esquema de refinamiento jerárquico conduce a la construcción secuencial de espacios anidados de splines con funciones de base linealmente independientes. La fácil generalización a 3D hace de esta técnica una opción atractiva para el refinamiento local. Sin embargo, un inconveniente de esta estrategia es la imposibilidad de definir espacios de splines sobre una T-mesh arbitraria, así como la presencia de funciones base redundantes y un excesivo solapamiento de los soportes de estas funciones. Un interesante enfoque de este último problema se analiza en [30]. Ellos utilizan una técnica de truncamiento de los soportes de las funciones base para reducir su solapamiento. Sin embargo, esta técnica tiene el inconveniente de elevar el coste computacional.

Otras estrategias para realizar un refinamiento local son las $C^1$-continuas PHT-splines [31] y las Locally refined B-splines (LR-splines) [32].

Vale la pena mencionar otro grupo de splines que no tienen la estructura de producto tensorial. A saber, splines definidas sobre triangulaciones [33], y en particular, splines definidas sobre una triangulación de Delaunay [34, 35, 36], box splines [37] y $C^1$ splines definidas sobre un refinamiento de Powell-Sabin del triángulo [38]. Ejemplos recientes de la aplicación del IGA utilizando splines sobre las triangulaciones se pueden encontrar en [39, 40, 41]. Las splines sobre triangulaciones ofrecen mayor flexibilidad y capacidad de adaptación a dominios irregulares que las splines basadas en producto tensorial, no obstante, quedan muchas cuestiones por resolver

en este campo, como por ejemplo su extensión a 3D.

### 1.2.2. Parametrización del dominio.

Los modelos CAD solo proporcionan la superficie que delimita a un sólido. Sin embargo, la aplicación del Análisis Isogeométrico requiere una representación volumétrica de la geometría. Por lo tanto, otro problema abierto en IGA es como generar una representación spline de un sólido a partir de la descripción CAD de su frontera. Como señala Cottrell et al. en [42], *"the most significant challenge facing isogeometric analysis is developing three-dimensional spline parameterizations from surfaces"*. Una parametrización es adecuada para el análisis si no tiene autointersección, es decir, si es invertible. Por otra parte, con el fin de conseguir una buena precisión en los resultados numéricos, es necesario que la parametrización sea de buena calidad, esto es, sin gran distorsión. La ortogonalidad y la uniformidad de las curvas isoparamétricas son deseables para una buena parametrización. Para dominios complejos no es trivial obtener una parametrización de buena calidad para la aplicación de IGA. Es esencial disponer de un método eficaz para su construcción.

En [43], la parametrización se determina resolviendo un problema de optimización con restricciones para encontrar las posiciones de los puntos de control de una superficie plana B -spline. La optimización consiste en la minimización de un funcional de energía que impone condiciones de ortogonalidad y uniformidad, donde además, se imponen restricciones para que la parametrización sea inyectiva. Esta idea se ha extendido a 3D en [44]. Otra técnica similar fue propuesta por estos autores en [45, 46]. Ellos utilizan un mapeo armónico obtenido a través de la resolución de un problema de optimización para los puntos de control. El uso de un mapeo armónico es una característica común de varios trabajos sobre métodos de parametrización en 2D y 3D. Por ejemplo, Li et al. [47] construyen un mapeo volumétrico utilizando el método de elementos de contorno. El algoritmo, que es complejo, puede ser aplicado a cualquier género de datos, pero requiere un proceso de prueba y error que debe ser guiado por la experiencia humana.

Martin et al. [48] presentar una metodología basada en funciones armónicas discretas para parametrizar un sólido, en donde el dato de entrada es una malla de tetraedros del sólido. El método implica la resolución de varias ecuaciones de de tipo Laplace mediante FEM, primero sobre la superficie y después sobre el dominio completo 3D. Se calculan dos funciones ortogonales armónicas para construir una malla de hexaedros del sólido que después se suaviza para eliminar irregularidades. El usuario debe elegir la posición de varios puntos críticos para establecer la superficie de parametrización y para fijar una semilla para generar el esqueleto.

Zhang et al. proponen en [49] un procedimiento para la construcción de sólidos de género cero a partir de la triangulación definida por la frontera de una malla de tetraedros. En primer lugar, se establece un mapeo entre la triangulación y la frontera del dominio paramétrico, el cubo unitario, utilizando para ello la parame-

trización de Floater. A continuación, se lleva a cabo una división octree del cubo que continúa hasta que el error entre la triangulación de entrada y la T-mesh de la superficie está por debajo de una tolerancia. Finalmente, los nodos del interior son recolocados mediante un proceso de optimización que maximiza el peor Jacobiano. El método fue extendido a sólidos de genero arbitrario en [50] .

## 1.3. Objetivo y contenido de la tesis

Esta tesis es el resultado del trabajo de nuestro grupo sobre Análisis Isogeométrico. Como se mencionó anteriormente, este es un método relativamente nuevo con numerosas cuestiones abiertas. Hemos estudiado y descubierto muchas cuestiones interesantes acerca de IGA desde que nos embarcamos en este proyecto. Pero, sobre todo, nuestra atención se ha centrado en dos problemas abiertos de IGA: la construcción de espacios de splines con propiedades adecuadas para el análisis y el problema de la parametrización volumétrica del dominio computacional. Algunos resultados preliminares sobre el método de parametrización desarrollado por nuestro grupo se incluyen brevemente en este trabajo, sin embargo, este problema va a ser estudiado en profundidad en otra tesis doctoral que se presentará próximamente.

En esta tesis se aborda el tema de la construcción de espacios de splines para su aplicación en el Análisis Isogeométrico. La principal contribución de esta tesis es el desarrollo de una nueva estrategia para la definición de bases spline que generan espacios con propiedades adecuadas para el análisis. La técnica está diseñada para T-meshes jerárquicas (mallas de varios niveles) generadas mediante una división quadtree/octree. Este tipo de mallas se puede implementar de manera eficiente con estructuras de datos de tipo árbol [51] que se utilizan con frecuencia en ingeniería. Debido a la elevada complejidad de todas las estrategias actuales, el objetivo principal que perseguimos aquí es la simplicidad y el bajo coste computacional, tanto en 2D como en 3D. Para poder definir bases de splines cúbicas tenemos que suponer una restricción sobre la T-mesh, esto es, que sea fuertemente balanceada.

La tesis se organiza como sigue:

- En el Capítulo 2 se presenta el núcleo de la tesis. Se describe el procedimiento para la construcción de un espacio de splines cúbicas sobre una T-mesh dada. La clave de nuestra técnica recae en ciertas reglas sencillas que se usan para inferir las bases del espacio spline. Se proporcionan los algoritmos para la construcción de estas bases, tanto para 2D como 3D.

- El Capítulo 3 presenta una breve descripción de otro resultado de nuestra investigación: la parametrización de splines en 2D y 3D para geometrías complejas.

- En el Capítulo 4 probamos las propiedades aproximadoras de las funciones propuestas. Aplicamos IGA para resolver diferentes problemas que implican refinamiento adaptativo y se analiza su orden de convergencia.
- En el Capítulo 5 se presenta conclusiones y se exponen las líneas futuras de investigación.

# CAPÍTULO 2

## Estrategia para la construcción de espacios spline polinómicos sobre T-meshes jerárquicas

En este capítulo se muestra el resultado principal de la tesis, esto es, una novedosa estrategia para definir funciones spline de producto tensorial sobre T-meshes. Debido a la alta complejidad de las estrategias existentes actualmente, el principal objetivo que se persigue aquí es conseguir una estrategia sencilla y de bajo coste computacional, tanto en 2D como en 3D. Para ello, tenemos que asumir una restricción en la T-mesh. Para nuestra estrategia, la malla debe ser una T-mesh 0-balanceada.

Asumiendo esta restricción sobre la T-mesh, podemos definir fácilmente funciones spline cúbicas que generan espacios con las propiedades deseadas: independencia lineal, continuidad $C^2$, propiedad de reproducción de polinomios cúbicos, encaje de espacios y facilidad de implementación. La clave de la estrategia recae en un conjunto sencillo de reglas para inferir los vectores de knots de cada función de base.

## 2.1. Principales etapas de la estrategia

La estrategia que proponemos tiene algunas similitudes con T-splines ya que las funciones de base se definen a partir de su vector de knots local, el cual se infiere atravesando las aristas de la T-mesh. En nuestra estrategia, además imponemos imponemos algunas reglas adicionales para los vectores de knot locales con la finalidad de obtener espacios spline con las propiedades deseadas. Estas reglas adicionales fueron obtenidas mediante el estudio y análisis de diferentes situaciones donde las reglas de T-spline no generaban espacios spline polinómicos. Además, algunas de características de nuestras reglas están inspiradas en el esquema de refinamiento jerárquico.

La estrategia está motivada por la idea de preservar el encaje de espacios. Para ello, tenemos que evitar las situaciones cuando una blending function no se puede reproducir con las bases del nuevo espacio T-spline que surgen tras un

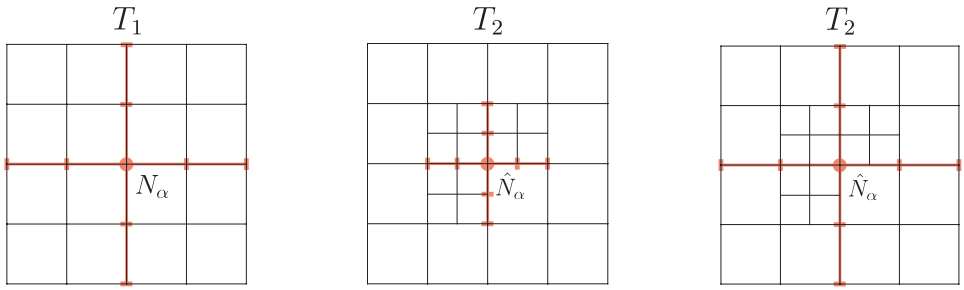



**Figura 2.1:** *Motivación de la estrategia. (a) Malla inicial y función de base $N_\alpha$. (b) Malla refinada, donde el nuevo espacio T-spline no es capaz de reporoducir la función original $N_\alpha$. (c) Intentamos preservar la función de base original $N_\alpha$ en el nuevo espacio cuando ésta no puede ser recuperada.*

refinamiento. Esto garantizaría el encaje de los espacios spline. Por ejemplo, en la malla inicial de la Fig. 2.1(a), definimos una función de base $N_\alpha$ asociada al vértice $\alpha$. Luego se refinan algunas celdas de la malla como se muestra en la Fig. 2.1(b). Se puede comprobar fácilmente que el nuevo espacio T-spline obtenido no es capaz de reproducir la función de base original $N_\alpha$. Sin embargo, si definimos la función $\hat{N}_\alpha$ como se muestra en la Fig. 2.1(c), podemos garantizar el encaje de espacios ya que en este caso la base inicial se mantiene sin modificar en el nuevo espacio. Esta es una de las situaciones donde necesitamos inferir para conseguir encaje de espacios. Las reglas que hemos desarrollado para inferir el soporte de las funciones tratan de conservar el comportamiento de encaje en todas las configuraciones posibles de una malla 0-balanceada.

En nuestra estrategia, el proceso de construcción del espacio spline para una T-mesh dada se puede dividir en las siguientes tres etapas:

1. *Pretratamiento de la malla (balanceo)*
2. *Inferencia de vectores de knot locales*
3. *Modificación de los vecotores de knots*

Cada una de estas etapas se describen a continuación.

## 2.2. Pretratamiento de la malla. 0-balanceo de T-meshes tipo quadtree y octree

La técnica que presentamos aquí está diseñada para T-meshes jerárquicas (mallas multinivel) con un esquema de subdivisión tipo quadtree y octree. Este

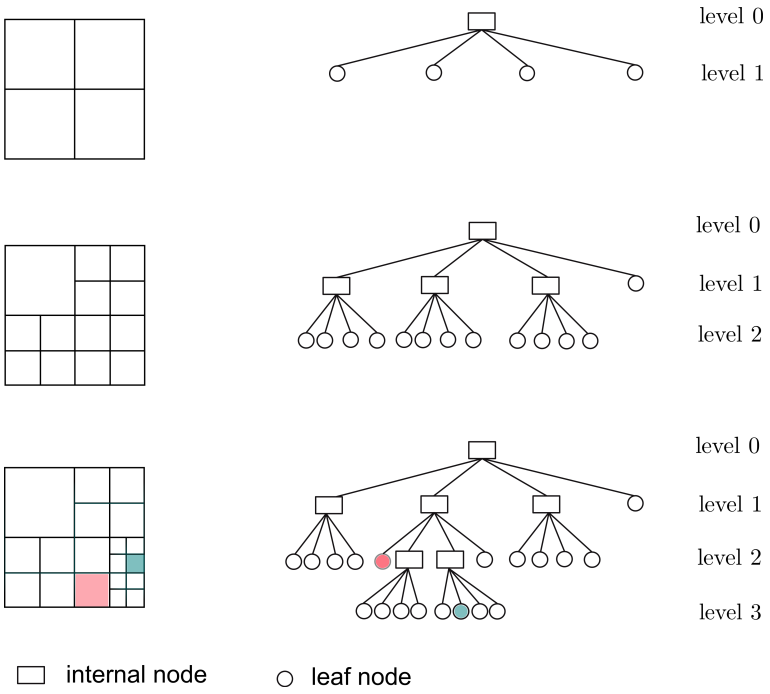



**Figura 2.2:** *Implementación de T-mesh jerárquica utilizando una estructra de datos quadtree.*

tipo de mallas se puede implementar eficientemente mediante estructuras de datos *tree* [51], las cuales son muy utilizadas en ingeniería.

La estructura de datos quadtree es una estructura de datos espacial utilizada para la descomposición recursiva del espacio. Un región bidimensional, normalmente un cuadrado, se divide en cuatro cuadrantes denominados *hijos*. Cada uno de estos nuevos cuadrantes puede volver a ser dividido y así sucesivamente. Los elementos del quadtree se denominan nodos del árbol. Cuando un nodo tiene hijos se le denomina nodo interno, y cuando no tiene hijos se le llama nodo hoja. Cada nodo hoja del quadtree representa una celda de la T-mesh, ver Fig. 2.2. Cada nodo del árbol pertenece a un cierto nivel $k$ y sus hijos pertenecen al nivel $k+1$. El nodo root tiene nivel $k = 0$. Cuando una hoja del árbol del nivel $k$ se refina, esta pasa a ser un nodo del árbol de nivel $k$. Se dice que una malla quadtree $T$ es del nivel $k$ si su celda más pequeña es del nivel $k$. El concepto de T-mesh quadtree se muestra en la Fig. 2.2.

Debido a su simplicidad, las mallas quadtree y octree son una herramienta atractiva para llevar a cabo refinamientos adaptativos en IGA y en el modelado geométrico. Para garantizar una buena calidad del espacio de aproximamión construído sobre una malla, es preferible tener una transición gradual desde la parte más gruesa de la malla hasta las zonas más refinadas. Por ello es común trabajar con mallas quadtree/octree balanceadas.

La estrategia que proponemos está diseñada exclusivamente para T-meshes

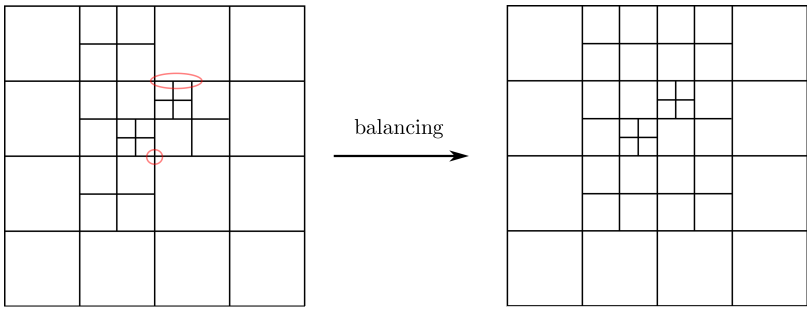



**Figura 2.3:** *Ejemplo del proceso de 0 balanceo. La Fig. de la izquierda muestra una malla quadtree no balanceada. La Fig. de la derecha corresponde a la malla resultante tras un procedimiento de balanceo.*

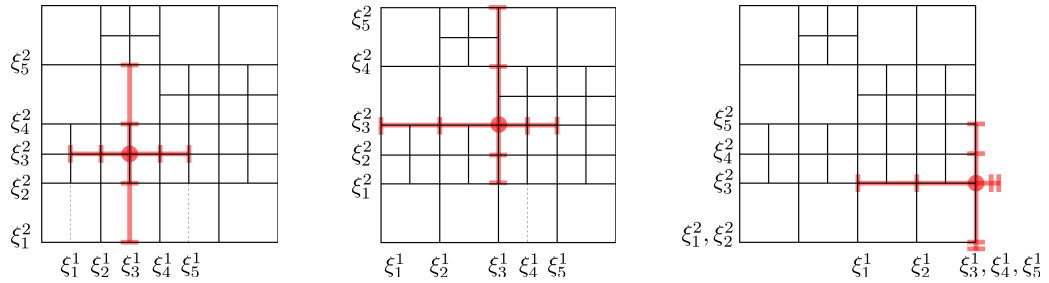
0-balanceadas. Una malla con estructura tipo tree está 0-balanceada si para cualquier $k$, ninguna celda de nivel $k$ comparte un vértice (0-cara) con una celda de nivel mayor a $k+1$. En otras palabras, un malla quadtree 0-balanceada implica que todas las celdas tienen contacto (vértice, arista o cara) sólo con celdas que difieren como mucho en un nivel de profunidad. Un ejemplo de quadtree 0-balanceado se muestra en Fig. 2.3. Para obtener un quadtree 0-balanceado se puede aplicar cualquier algoritmo estándar de balanceo. Es importante resaltar que los refinamientos llevados a cabo durante el 0-balanceo no se propagan, ver [56].

Es necesario destacar que el 0-balanceo de la T-mesh es un pre-requisito esencial para la construcción de espacios spline utilizando nuestra técnica. En general, si la T-mesh no está 0-balanceada, nuestras reglas para inferir los vectores de knots no conducen a espacios polinómicos. Además, es importante resaltar que que para nuestras 2D (3D) T-meshes, la subdivisión de cualquier celda se realiza mediante la subdivisión de la celda en 4 (8) subceldas iguales, de modo que todas las celdas del mismo nivel tienen en mismo tamaño y el tamaño de arista de una celda de nivel $k$ is dos veces más grande que el tamaño de arista de una celda de nivel $k+1$.

## 2.3. Inferencia de vectores de knots locales

Consideremos una T-mesh $T$ del espacio paramétrico cuadrado $\Omega = [0,1]^d$, $d = 2$ o 3. Llamamos *nodo regular* al nodo que no forma una unión en T. Vamos a asociar funciones de base sólo a los *nodos regulares* de la malla. El *esqueleto* de una malla $T$ de dimensión $d$ consiste en el lugar geométrico de puntos compuesto por la unión de todas las $(d-1)$ caras de la malla y se denota como skt$(T)$. En el caso 2D, el *esqueleto* de la malla sería la unión de todas las aristas de la malla, y en el caso 3D es la unión de todas las caras.

Para definir nuestras funciones spline de base sobre una T-mesh de dimensión $d$, un vector de knots local para cada dirección paramétrica $d$ debe ser asignado a cada función $N_\alpha$: $\Xi_\alpha^j = \left(\xi_1^j, \xi_2^j, \xi_3^j, \xi_4^j, \xi_5^j\right)$, $j = 1, ..., d$. De forma similar a [21], estos




(a) No T-junction is skipped.

(b) Se salta dos uniones en T en cada dirección.

(c) Se salta dos uniones en T y se repiten knots cuando se alcanza la frontera.

**Figura 2.4:** *Inferencia de vectores de knots locales para una función bivariada, atravesando las aristas de la T-mesh.*

vectores de knots se infieren atravesando el esqueleto de la T-mesh. Por simplicidad, vamos a describir este proceso para una malla bidimensional. Comenzando desde el knot central $(\xi_3^1, \xi_3^2)$, es decir, el ancla de la función, caminamos a través de la T-mesh hasta intersectar perpendicularmente una arista de la malla. Según nuestra estrategia, debemos saltar las uniones en T donde la arista ausente es perpendicular a la dirección de nuestra marcha, ver Fig. 2.4. Cuando se alcanza la frontera del dominio paramétrico, repetimos los knots creando así una estructura de vectores de knots abiertos a lo largo de la frontera, ver Fig. 2.4(c). Hay que tener en cuenta que todos los knots interiores tienen multiplicidad 1. De este modo obtenemos un conjunto de funciones de base $\{N_\alpha\}_{\alpha \in A_T}$, siendo $A_T$ el conjunto de índices. La Fig. 2.5 muestra un ejemplo de T-mesh en el espacio paramétrico y los anclas de todas la funciones de base definidas sobre esta malla. Todos los nodos regulares del interior tienen exáctamente una función asociada, pero a los nodos del contorno se les asocia más de una función debido a la estructura de vector de knots abierto.

El proceso de inferencia de los vectores de knots locales se puede resumir de la siguiente forma:

- *Las funciones de base se asocian sólo a los nodos regulares de la malla.*
- *Los vectores de knots locales se infieren caminando a través de la malla hasta intersectar perpendicularmente el esqueleto de la malla. Esta intesección no debe coincidir con una T-junction perpendicular a la dirección de marcha.*
- *Los knots se repiten en el contorno para crear una estructura de vectores de knots abiertos a lo largo de la frontera.*

Luego se modifica el soporte de algunas funciones para generar un espacio spline con buenas propiedades. Este proceso se explica en la siguiente sección.

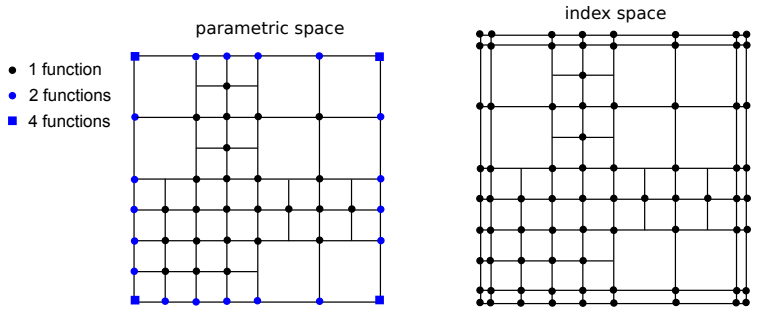



**Figura 2.5:** *Un ejemplo de T-mesh con sus anclas. Los círculos rojos representan los nodos del interior que tienen una función de base asociada, los círculos negros son los nodos del contorno que tienen dos funciones de base y los cuadrados negros son los nodos del contorno que tienen 4 funciones debido la estructura de vectores de knots abiertos a lo largo de la frontera.*

## 2.4. Modificación de los vectores de knots locales

La clave de nuestra estrategia recae en un conjunto de sencillas reglas para la modificación de los soportes de las funciones que conlleva a la construcción de espacios spline polinómicos sobre una T-mesh 0-balanceada dada. Para describir la idea, primero vamos a introducir alguna notación. Para los vectores de knots locales $\Xi_\alpha^j = \left(\xi_1^j, \xi_2^j, \xi_3^j, \xi_4^j, \xi_5^j\right)$ , $j = 1, ..., d$ denominamos la longitud de cada intervalo de knots como $\Delta_i^j = \xi_{i+1}^j - \xi_i^j$, $j = 1, ..., d$ y $i = 1, ..., 4$.

El soporte de un función de base $d$-variada $N_\alpha$ is un rectángulo $d$-dimensional *box*: $[\xi_1^1, \xi_5^1] \times \cdots \times [\xi_1^d, \xi_5^d]$. Llamaremos *frame* del soporte de una función a la unión de todas las $(d-2)$-caras de su *box* y lo denotaremos como $\mathrm{frm}(\mathrm{supp}\, N_\alpha)$. Es decir, para el soporte rectangular de una función bivariada, el *frame* es la unión de los cuatro vértices de su rectángulo. Para el soporte cúbico de una función trivariada, su *frame* está compuesto por la unión de las doce aristas de ese cubo.

Una vez que se ha inferido el soporte de la función, éste se modifica de modo que para cada función de base $N_\alpha$ sus nuevos vectores de knots $\Xi_\alpha^j$ $j = 1, ..., d$ verifican las siguientes condiciones:

**Condición 1:** *Vector de knots local de la función d-variada* $N_\alpha$ *verifica*[1]

$$\Delta_1^j \geqslant \Delta_2^j = \Delta_3^j \leqslant \Delta_4^j, \quad j = 1, ..., d, \tag{2.1}$$

**Condición 2:** *El frame del soporte de la función debe estar situado sobre el esqueleto de la malla*:

$$\mathrm{frm}(\mathrm{supp}\, N_\alpha) \in \mathrm{skt}(T). \tag{2.2}$$

[1]Exceptos los casos donde se repiten knots, los cuales se explican al final de la Sección 2.5.1.

Por lo tanto, los soportes de función que no cumplan las Condiciones 1 y 2 deben ser modificados. Para ello, el soporte original se extiende modificando algunos intervalos de knots hasta que el soporte resultante satisface ambas condiciones. Nos vamos a referir a estas modificaciones de soporte como *Reglas de extensión* 1 y 2 respectivamente.

In la siguiente sección se explica de forma detallada este proceso para los casos 2D y 3D.

## 2.5. Reglas de modificación de soportes

Aquí mostramos las reglas de extensión de soportes para obtener vectores de knots locales que cumplan las Condiciones 1 y 2 formuladas en la sección anterior. El proceso es el siguiente. Primero, si tras atravesar el *esqueleto* de la T-mesh, el vector de knots local de una función no cumple la Condición 1, algunos de sus knots se modifican para que la cumplan. Luego, se comprueba que cumpla la Condición 2, y si no es así, se realiza otra modificación de los knots. Como resultado de estas modificaciones, se obtiene un nuevo soporte extendido con vectores de knots locales que verifcan ambas condiciones. Estas modificaciones se pueden implementar fácilmente debido a la estructura de árbol balanceado de nuestra malla. Veamos este proceso en detalle.

Para aclarar la notación, en adelante denotaremos las coordenadas paramétricas como $(\xi, \eta, \zeta)$, las cuales corresponden a $(\xi^1, \xi^2, \xi^3) = (\xi, \eta, \zeta)$ en la notación previa. Consecuentemente, $(\Xi^1, \Xi^2, \Xi^3) = (\Xi, \mathcal{H}, \mathcal{Z})$ y $(\Delta_i^1, \Delta_i^2, \Delta_i^3)$ $= (\Delta_i^\xi, \Delta_i^\eta, \Delta_i^\zeta)$.

### 2.5.1. Reglas de extensión de soportes para mallas 2D

Para facilitar la descripción e ilustración de la estrategia, algunos conceptos y notación introducidos en la sección 2.4 se particularizan para el caso 2D. El *esqueleto* $\mathrm{skt}(T)$ de una malla $T$ de dimensión 2 es la unión de todas las aristas de la malla. Para una función bivariada llamaremos a los vértices de su soporte rectangular como $V_{1,1} = (\xi_1, \eta_1)$, $V_{5,1} = (\xi_5, \eta_1)$, $V_{5,5} = (\xi_5, \eta_5)$ y $V_{1,5} = (\xi_1, \eta_5)$. Luego, el *frame* del soporte de una función la unión de sus cuatro vértices, esto es, $\mathrm{frm}(\operatorname{supp} N_\alpha) = \{V_{n,m},\ n, m \in \{1, 5\}\}$. La Fig. 2.6 muestra esta notación para el soporte de una función bivariada.

La formulación de la Condición 1 para los vectores de knots locales $\Xi$ y $\mathcal{H}$ es sencillo y no require aclaración. En cuanto a la Condición 2 para el caso 2D se formula de la siguiente forma: *los cuatro vértices del soporte de la función deben estar siguados sobre las aristas de la malla.*

Regla de extensión 1. Si el vector de knots local $\Xi$ de una función no satisface la Condición 1, modificamos este vector saltando el número mínimo de knots hasta que se verifica $\Delta_1^\xi \geqslant \Delta_2^\xi = \Delta_3^\xi \leqslant \Delta_4^\xi$, y análogamente, para $\mathcal{H}$. Esta modificación se hace de forma independiente para cada dirección paramétrica aplicando

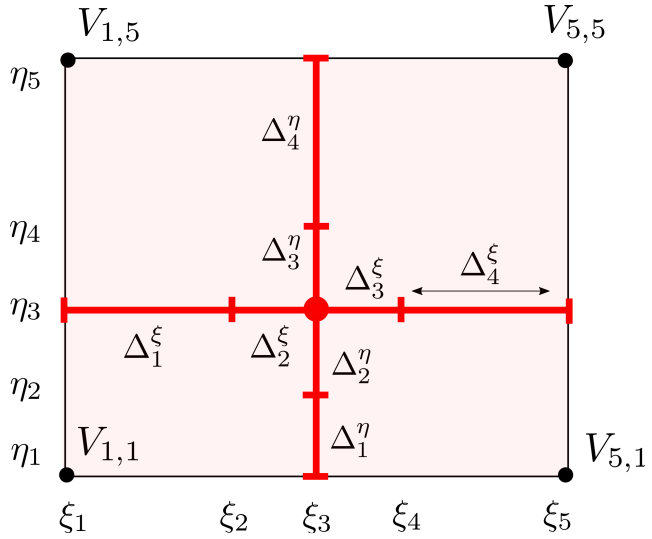
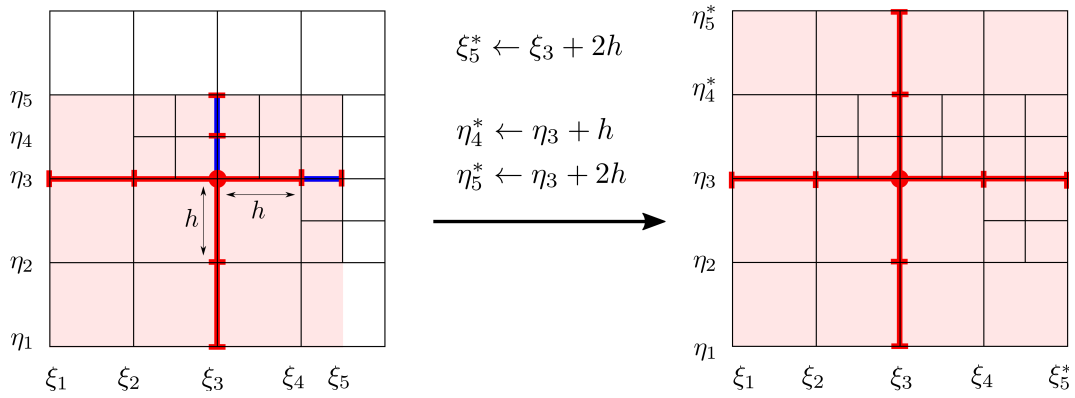



**Figura 2.6:** *Notación de los soportes para el caso 2D*

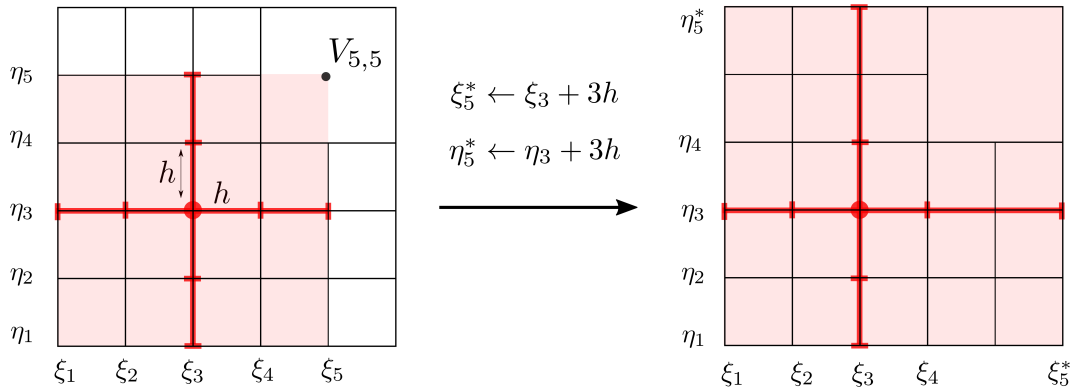
determinadas reglas de extensión. Veamos un ejemplo de extensión de soporte para una función bivariada. El soporte de función de más a la izquierda mostrado en la Fig. 2.7(a) no satisface la Condición 1. Para el vector de knots $\Xi$ tenemos $\Delta_3^\xi > \Delta_4^\xi$, por lo que el intervalo de knots $\Delta_4^\xi$ debe ser modificado. Llamemos $h = \text{máx}(\Delta_2^\xi, \Delta_3^\xi) = \text{máx}(\Delta_2^\eta, \Delta_3^\eta)$. Destacar que ambos máximos coinciden debido a la estructura quadtree y a que se saltan las T-junctions. Luego, el quinto knot $\xi_5$ se redefine como $\xi_5^* \leftarrow \xi_3 + 2h$. Para el vector de knots $\mathcal{H}$ tenemos $\Delta_2^\eta > \Delta_3^\eta$, por lo que los knots $\eta_4$ y $\eta_5$ se deben modificar como $\eta_4^* \leftarrow \eta_3 + h$, $\eta_5^* \leftarrow \eta_3 + 2h$.

Regla de extensión 2. Una vez se cumple la Condición 1, para la Condición 2 comprobamos si los vértices del soporte de la función están situados sobre las aristas de la malla. Si no es así, modificamos los vectores de knots saltando un knot en ambas direcciones paramétricas situando así el vértice sobre las aristas de la malla. La Fig. 2.7(b) muestra este proceso. La comprobación se realiza de forma independiente para cada uno de los cuatro cuadrantes del soporte de la función. Hay que destacar que para nuestro quadtree 0-balanceado sólo necesitamos hacer esta comprobación para algunas funciones. Por ejemplo, sin perder generalidad, el vértice del soporte $V_{5,5} = (\xi_5, \eta_5)$ se debe comprobar sólo si $\Delta_3^\xi = \Delta_4^\xi = \Delta_3^\eta = \Delta_4^\eta$. En la Fig. 2.7(b) se muestra un ejemplo de soporte de función que no cumple la condición 2. El vértice $V_{5,5} = (\xi_5, \eta_5)$ de este soporte no está situado sobre las aristas de la malla, por lo que el quinto knot en ambas direcciones paramétricas debe ser redefinido como $\xi_5^* \leftarrow \xi_3 + 3h$, $\eta_5^* \leftarrow \eta_3 + 3h$, y así, el nuevo vértice $V_{5,5}$ se situa sobre las aristas de la malla.

Para otros soportes de funciones, la extensión es completamente análoga a estos dos ejemplos. En todos los casos posibles, la extensión del soporte de una función implica cambiar uno o dos intervalos de knots duplicando su tamaño.




(a) No se cumple la Condición 1 porque $\Delta_4^\xi < \Delta_3^\xi$ y $\Delta_3^\eta < \Delta_2^\eta$.




(b) No se cumple la Condición 2 porque $V_{5,5} \notin skt(T)$.

**Figura 2.7:** *Reglas de extensión. (a) Ejemplo de modificación de soporte según la Regla de extensión 1. (b) Un ejemplo de modificación de soporte según la Regla de extensión 2.*

Los Algoritmos 1 y 2 muestran de forma detallada las Reglas de extensión para modificar el soporte de una función bivariada acorde a las Condiciones 1 y 2.

La Fig. 2.8 muestra algunos ejemplo de modificacion de soportes. Funciones que satisfacen ambas condiciones y por tanto no deben ser modificadas se muestran en la Fig. 2.8(a). En las Fig. 2.8(b), (c), (d) y (e) se muestra ejemplos de extensión de soportes de acuerdo con la Condición 1. En las Fig. s 2.8(f), (g) y (h) se ilustra la extensión de soportes de acuerdo con la Condición 2 o con ambas condiciones.

Hay que destacar que una excepción para la Condición 1 es un vector de knots que contiene un intervalo de knots de longitud 0 debido a la estructura de vector de knots abierto a lo largo de la fronterea. In este caso, un vector de knots debe cumplir la desigualdad (2.1) sin tener en cuenta los intervalos de knots de longitud 0. Consecuentemente, una excepción para la aplicación de las reglas de extensión is el caso cuando se alcanza la frontera del dominio paramétrico atravesando las aristas de la T-mesh, ver Fig. 2.8(c), (d) y (e).

**Algoritmo 1:** Regla de extensión 1.

**Input**: A knot vector $\Xi = (\xi_1, \xi_2, \xi_3, \xi_4, \xi_5)$, $\xi_i \in [0, 1]$.

**Function** ***Modify1***$(\Xi)$
- $\Xi^* \leftarrow \Xi$
- $h = \text{máx}(\Delta_2, \Delta_3)$
- **if** $\Delta_2 < \Delta_3$ *and* $\xi_2 > 0$ **then**
  - $\xi_2^* \leftarrow \xi_3 - h$
  - $\xi_1^* \leftarrow \xi_2^*$
  - **if** $\xi_1^* > 0$ **then** $\xi_1^* \leftarrow \xi_3 - 2h$
- **if** $\Delta_1 < \Delta_2$ *and* $\xi_1 > 0$ **then**
  - $\xi_1^* \leftarrow \xi_3 - 2h$
- **if** $\Delta_2 > \Delta_3$ *and* $\xi_4 < 1$ **then**
  - $\xi_4^* \leftarrow \xi_3 + h$
  - $\xi_5^* \leftarrow \xi_4^*$
  - **if** $\xi_5^* < 1$ **then** $\xi_5^* \leftarrow \xi_3 + 2h$
- **if** $\Delta_4 < \Delta_3$ *and* $\xi_5 < 1$ **then**
  - $\xi_5^* \leftarrow \xi_3 + 2h$
- **return** $\Xi^*$

**Output**: A corrected knot vector $\Xi^*$ that satisfies Condition 1.

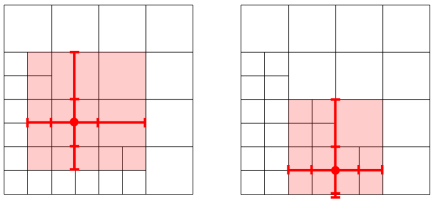
**Algoritmo 2:** Regla de extensión 2 en 2D.

**Input**: A *0-balanced* mesh $T$ and a pair of local knot vectors $S = \{\Xi, \mathcal{H}\}$.

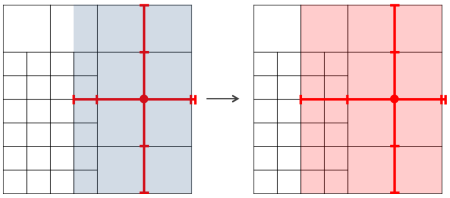
**Function** ***Modify2***$(T,\ S)$
- $S^* \leftarrow S$
- $h = \text{máx}(\Delta_2^\xi, \Delta_3^\xi)$
- **for** $n \in \{1, 5\}$ **do**
  - **for** $m \in \{1, 5\}$ **do**
    - **if** $(\xi_n, \eta_m) \notin skt(T)$ **then**
      - $\xi_n^* \leftarrow \xi_3 + 3h\,\text{sgn}(\xi_n - \xi_3)$
      - $\eta_m^* \leftarrow \eta_3 + 3h\,\text{sgn}(\eta_m - \eta_3)$
- **return** $S^*$

**Output**: A modified support $S^* = \{\Xi^*, \mathcal{H}^*\}$ that satisfies Condition 2.

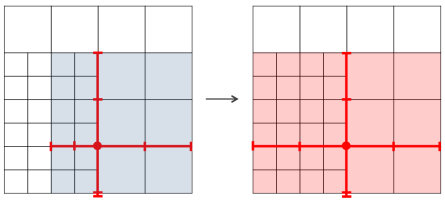
**Remark 1.** *Es importante señalar que la aplicación de las reglas de extensión siempre sitúa a los nuevos knots sobre las aristas de la malla, es decir, la reglas de extensión sólo salta algunos knots, pero no crea nuevos knots que no sean inducidos por la T-mesh.*



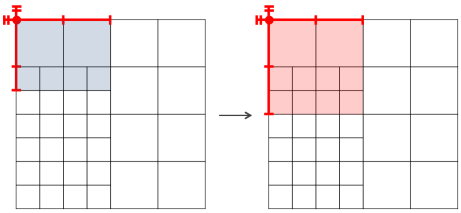
(a) Soportes iniciales que satisfacen ambas condiciones y no deben ser modificados



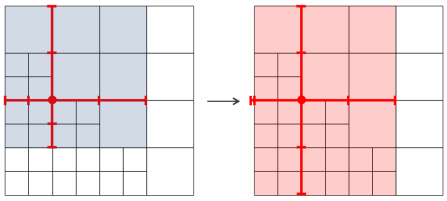
(b) La Condición 1 no se satisface porque $\Delta_1^\xi < \Delta_2^\xi$, así que $\xi_1^* \leftarrow \xi_3 - 2h$.



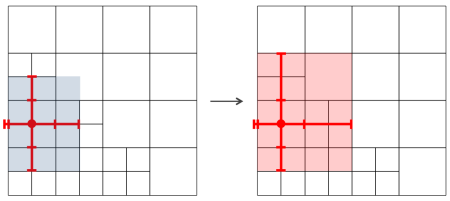
(c) La Condición 1 no se satisface porque $\Delta_2^\xi < \Delta_3^\xi$, así que $\xi_1^* \leftarrow \xi_3 - 2h$ y $\xi_2^* \leftarrow \xi_3 - h$.



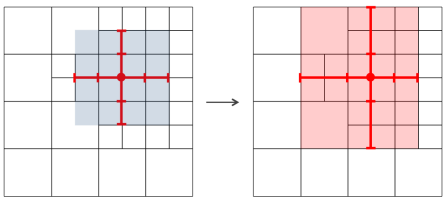
(d) La Condición 1 no se satisface porque $\Delta_1^\eta < \Delta_2^\eta$, así que $\eta_1^* \leftarrow \eta_3 - 2h$.



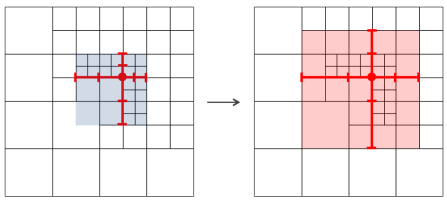
(e) La Condición 1 no se satisface porque $\Delta_2^\xi < \Delta_3^\xi$ y $\Delta_2^\eta < \Delta_3^\eta$, así que $\xi_2^* \leftarrow \xi_3 - h$, $\eta_2^* \leftarrow \eta_3 - h$ y $\eta_1^* \leftarrow \eta_3 - 2h$.



(f) La Condición 2 no se satisface porque $V_{5,5} \notin skt(T)$, así que $\xi_5^* \leftarrow \xi_3 + 3h$ y $\eta_5^* \leftarrow \eta_3 + 3h$.



(g) La Condición 2 no se satisface porque $V_{1,1}$ y $V_{1,5} \notin skt(T)$, así que $\xi_1^* \leftarrow \xi_3 - 3h$, $\eta_1^* \leftarrow \eta_3 - 3h$ y $\eta_5^* \leftarrow \eta_3 + 3h$.



(h) Las Condiciones 1 y 2 no se satisfacen $\Delta_2^\xi > \Delta_3^\xi$, $\Delta_2^\eta > \Delta_3^\eta$ y $V_{1,1} \notin skt(T)$, así que $\xi_4^* \leftarrow \xi_3 + h$, $\xi_5^* \leftarrow \xi_3 + 2h$, $\eta_4^* \leftarrow \eta_3 + h$, $\eta_5^* \leftarrow \eta_3 + 2h$, $\xi_1^* \leftarrow \xi_3 - 3h$, $\eta_1^* \leftarrow \eta_3 - 3h$.

**Figura 2.8:** *Ejemplos de modificación de soporte de funciones con las Reglas de extensión 1 y 2. El soporte inicial se marca en azul y el extendido en rojo.*

**Remark 2.** *Hay que destacar que en el caso cuando el soporte de una función viola la Condición 2, la extesión de su soporte para situar sus vértices sobre el esqueleto*

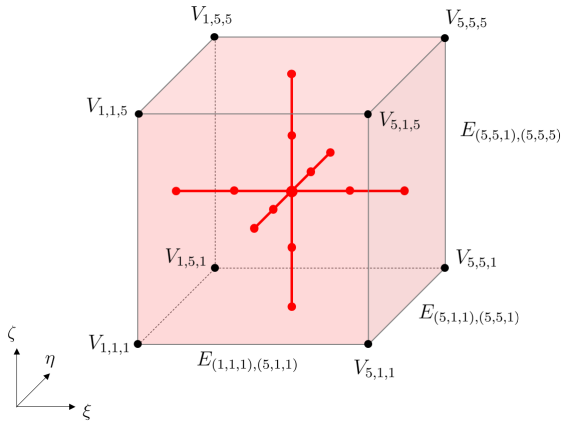



**Figura 2.9:** *Notación del soporte para el caso 3D.*

*de la malla, puede realizarse modificando sólo uno de los vectores de knots en lugar de modificar ambos. Esa opción llevaría a la pérdida de simetría en el espacio spline y a la no unicidad del soporte resultante. Así que por simplicidad, optamos por extender los vectores de knots en ambas direcciones paramétricas.*

**Remark 3.** *Es importante subrayar que la inferencia y modificación de cada soporte de función no depende del resto de funciones, por lo que el proceso se podría paralelizar.*

### 2.5.2. Extensión de soportes para mallas 3D

En esta sección se describe e ilustra la estrategia propuesa para definir funciones spline trivariadas sobre una T-mesh con estructura octree 0-balanceada.

El *esqueleto* $\mathrm{skt}(T)$ de una malla tridimensional $T$ es la unión de todas las caras de la malla. Para una función trivariada, llamamos a los vértices de su soporte como $V_{n,m,k} = (\xi_n, \eta_m, \zeta_k)$ donde $n, m, k \in \{1, 5\}$. Y la arista de un soporte formada por los vértices $V_{n,m,k}$ y $V_{p,q,r}$ se denota como $E_{(n,m,k),(p,q,r)}$. Luego, el *frame* $\mathrm{frm}(\mathrm{supp}\, N_\alpha)$ del soporte de una función trivariada is la unión de sus 12 aristas. La Fig. 2.9 muestra la notación introducida para el soporte de una función de base trivariada.

La formulación de la Condición 1 para los vectores de knots locales de un función trivariada es análogo al caso 2D. La Condición 2 adaptada a mallas 3D es la siguiente: *las aristas del soporte cuboidal de la función deben estar situadas sobre las caras de la malla.*

La implementación de la estrategia para 3D es similar al caso 2D. Para satisfacer la Condición 1, se aplica la Regla de extensión 1 a cada uno de los tres vectores de knots de una función, análogamente al caso 2D usando el Algoritmo 1.

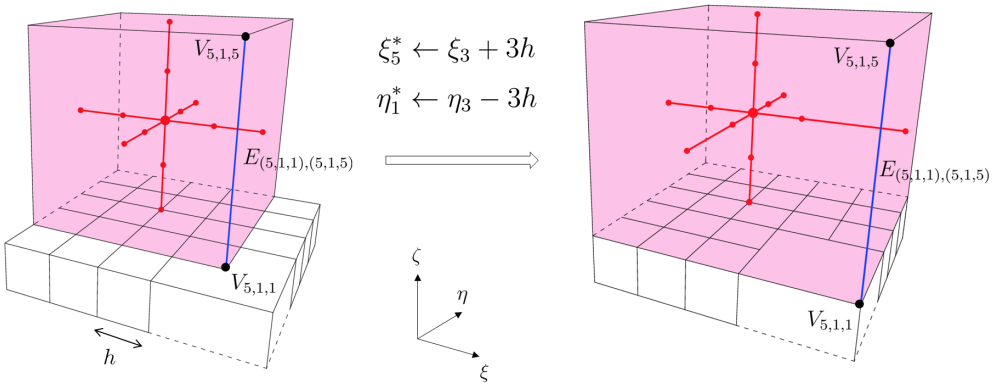
Regla de extensión 2. Para satisfacer la Condición 2, se comprueba si las aristas del soporte de la función están situadas sobre las caras de la malla. Si no es

así, se debe modificar los dos vectores de knots perpendiculares a esta arista, saltando un knot en ambas direcciones paramétricas y situando así esta arista sobre las casras de la malla. Esta comprobación se realiza de forma independiente para cada uno de los 8 cuadrantes del soporte de la función y, en cada cuadrante, se debe comprobar tres aristas. La Fig. 2.10 ilustra el procedimiento de extensión de soporte para el cuadrante del vértice $V_{5,1,1}$. Debido a la estructura octree, sólo puden ocurrir dos casos: (i) el cuadrante contiene una arista que no está situada sobre las caras de la malla o (ii) el cuadrante contiene tres aristas y un vértice que no están situados sobre el *esqueleto* de la malla. A continuación estudiamos cada caso.

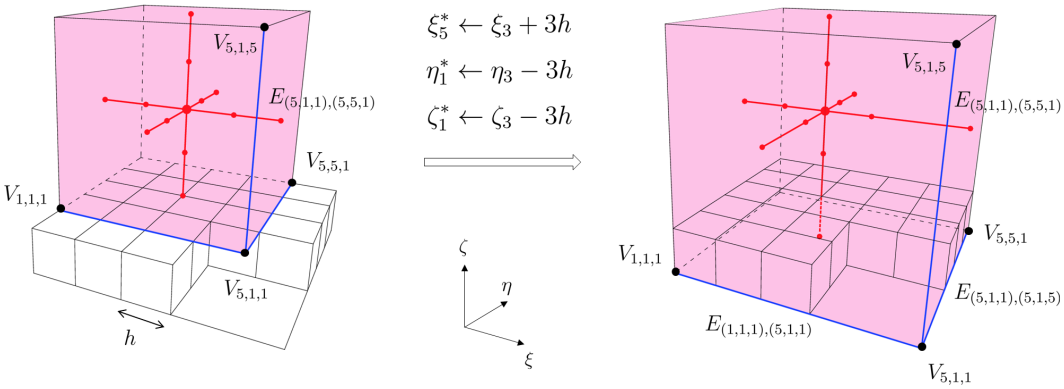
(i) Si un cuadrante contiene una arista que con cumple la Condición 2, entonces se modifican los dos vectores de knots perpendiculares a esta arista, ver Fig. 2.10(a). Para el soporte de función mostrado en la Fig. 2.10(a) izquierda, la arista $E_{(5,1,1),(5,1,5)}$ no está situada sobre las caras de la malla. Por tanto, se modica sus dos vectores de knots $\Xi$ y $\mathcal{H}$ perpendiculares a esta arista de modo que la arista se sitúe sobre las caras de la malla, es dcir, los knots $\xi_5$ y $\eta_1$ se redefinen como $\xi_5^* \leftarrow \xi_3 + 3h$ y $\eta_1^* \leftarrow \eta_3 - 3h$, donde $h = \text{máx}(\Delta_2^\xi, \Delta_3^\xi) = \text{máx}(\Delta_2^\eta, \Delta_3^\eta) = \text{máx}(\Delta_2^\zeta, \Delta_3^\zeta)$.

(ii) Si un cuadrante contiene tres aristas que no están situadas sobre las caras de la malla, entonces se modifican los tres vectores de knots saltando un knot en cada dirección paramétrica, ver Fig. 2.10(b). El vértice $V_{5,1,1}$ y las tres aristas conectadas a él no están situadas sobre el *esqueleto* de la malla, por lo que se modifican los tres vectores de knots para situar las tres aristas sobre las caras de la malla: $\xi_5^* \leftarrow \xi_3 + 3h, \quad \eta_1^* \leftarrow \eta_3 - 3h, \ \zeta_1^* \leftarrow \zeta_3 - 3h$.

El Algoritmo 3 explica la Regla de extensión 2 utilizada para modificar el soporte de una función trivariada acorde a la Condición 2.




(a) Sólo una arista violando la Condición 2. El nodo $V_{5,1,1}$ se sitúa en el centro de la cara de tamaño $2h$. El soporte se extiende enn dos direcciones.




(b) Tres aristas violando la Condición 2. El nodo $V_{5,1,1}$ se sitúa en el centro de la celda de tamaño $2h$. El soporte se extiende en tres direcciones.

**Figura 2.10:** *Regla de extensión 2 para la modificación del soporte de una función trivariada.*

**Algoritmo 3:** Regla de extensión 2 en 3D.

**Input**: A *0-balanced* T-mesh $T$ and three local knot vectors $S = \{\Xi, \mathcal{H}, \mathcal{Z}\}$.

**Function** ***Modify2*** $(T,\ S)$
- $S^* \leftarrow S$
- $h = \text{máx}(\Delta_2^\xi, \Delta_3^\xi)$
- $mov(i) := i + 4\,\text{sgn}(3 - i)$
- **for** $n \in \{1, 5\}$ **do**
  - **for** $m \in \{1, 5\}$ **do**
    - **for** $k \in \{1, 5\}$ **do**
      - **if** $E_{(n,m,k),(mov(n),m,k)} \notin skt(T)$ **then**
        - $\eta_m^* \leftarrow\ \eta_3 + 3h\,\text{sgn}(\eta_m - \eta_3)$
        - $\zeta_k^* \leftarrow\ \zeta_3 + 3h\,\text{sgn}(\zeta_k - \zeta_3)$
      - **if** $E_{(n,m,k),(n,mov(m),k)} \notin skt(T)$ **then**
        - $\xi_n^* \leftarrow\ \xi_3 + 3h\,\text{sgn}(\xi_n - \xi_3)$
        - $\zeta_k^* \leftarrow\ \zeta_3 + 3h\,\text{sgn}(\zeta_k - \zeta_3)$
      - **if** $E_{(n,m,k),(n,m,mov(k))} \notin skt(T)$ **then**
        - $\xi_n^* \leftarrow\ \xi_3 + 3h\,\text{sgn}(\xi_n - \xi_3)$
        - $\eta_m^* \leftarrow\ \eta_3 + 3h\,\text{sgn}(\eta_m - \eta_3)$
- **return** $S^*$

**Output**: A modified support $S^* = \{\Xi^*, \mathcal{H}^*, \mathcal{Z}^*\}$ that satisfies Condition 2.

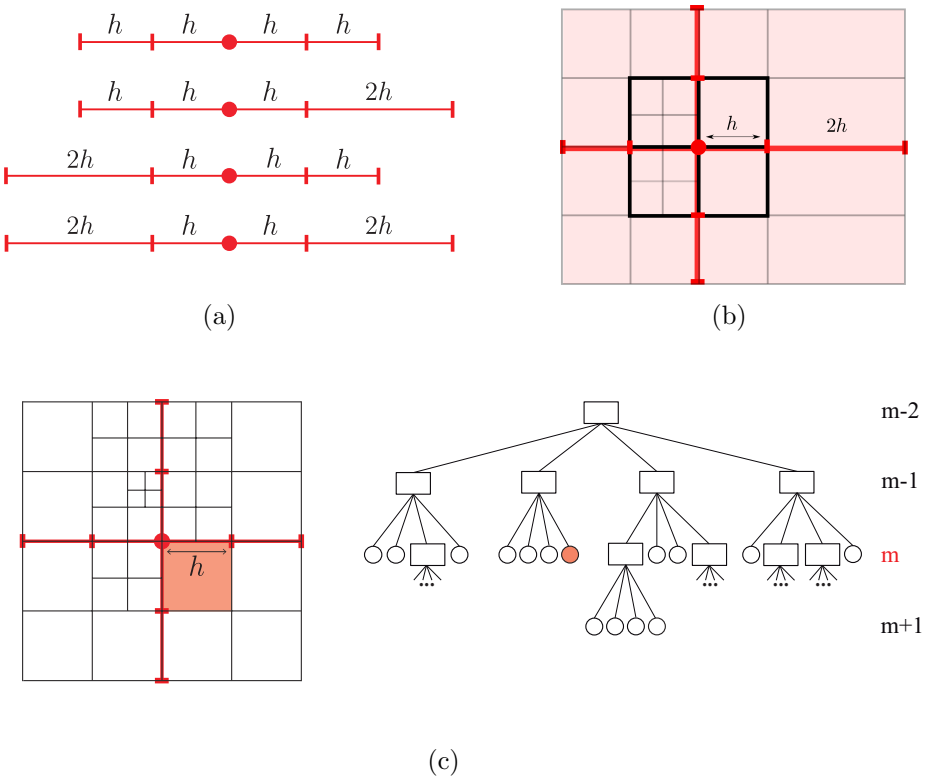



**Figura 2.11:** *(a) Todas las configuraciones posibles para un knot vector. (b) Para cualquier función se cumple* $h = \Delta_2^\xi = \Delta_3^\xi = \Delta_2^\eta = \Delta_3^\eta$. *(c) Un soporte en función del nivel* $m$*, que se determina por la celda marcada en rojo.*

## 2.6. Clasificación de los soportes bivariados

Cabe mencionar que las reglas propuestas y la restricción de 0-balanceo de la malla conduce a un número muy limitado de todas las funciones posibles que se pueden definir sobre una T-mesh.

Consideremos cualquier knot vector $\Xi = (\xi_1, \xi_2, \xi_3, \xi_4, \xi_5)$ y sus intervalos $\Delta_i^\xi = \xi_{i+1} - \xi_i$, $i = 1, ..., 4$. El conjunto $\boldsymbol{\Delta}^\xi = (\Delta_1^\xi, \Delta_2^\xi, \Delta_3^\xi, \Delta_4^\xi)$ determina el knot vector $\Xi$. Debido a Regla de extensión 1 y la condición de malla balanceada, tenemos las siguientes configuraciones posibles para el conjunto $\boldsymbol{\Delta}^\xi$: $(h, h, h, h)$, $(h, h, h, 2h)$, $(2h, h, h, h)$ y $(2h, h, h, 2h)$, ver Fig. 2.11(a). Todos los posibles soportes de función bivariada se obtienen mediante la combinación de todas las configuraciones de $\Xi$ y $\mathcal{H}$, es decir, la combinación de dos conjuntos $\boldsymbol{\Delta}^\xi$ y $\boldsymbol{\Delta}^\eta$. Hay que tener en cuenta que, puesto que nos saltamos las T-junctions mientras inferimos los knot vectors, el valor de $h$ es el mismo para $\boldsymbol{\Delta}^\xi$ y $\boldsymbol{\Delta}^\eta$, es decir, $h = \Delta_2^\xi = \Delta_3^\xi = \Delta_2^\eta = \Delta_3^\eta$, ver Fig. 2.11(b), y el valor de $h$ determina el nivel de una función. Decimos que una función es de nivel $m$ si la celda más grande que contacta con su ancla es de nivel $m$, ver Fig. 2.11(b) (c). Por lo tanto, las funciones definidas según nuestra

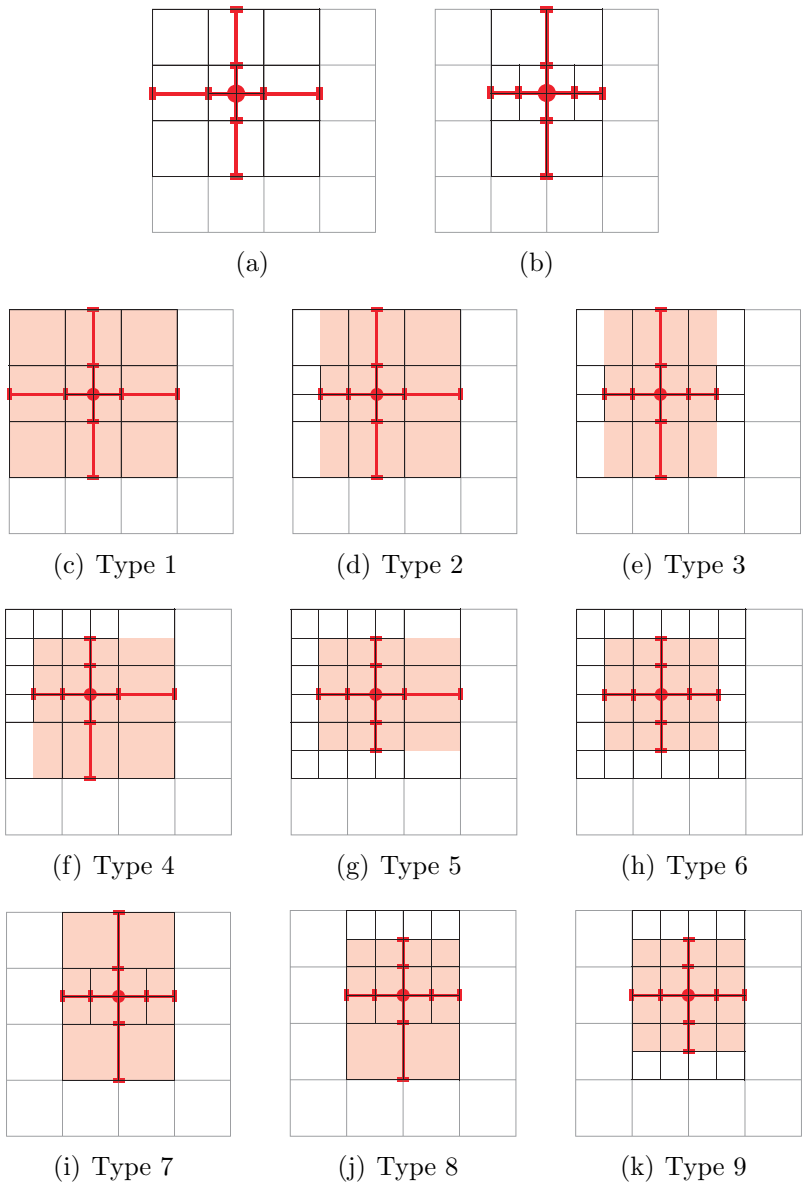



**Figura 2.12:** *Dos escenarios posibles para un soporte de acuerdo con la malla local subyacente. (a) El ancla de la función de nivel m coincide con el centro de una celda de nivel (m-1). (b) El resto de las funciones. (c)-(k) Clasificación de funciones de acuerdo con la malla local subyacente y conFig. ciones de $\Delta^{\xi}$ y $\Delta^{\eta}$.*

estrategia tienen una jerarquía.

Podemos clasificar todas las funciones de la siguiente manera. En primer lugar, podemos distinguir una función de acuerdo con la malla local subyacente que conduce a este soporte. Básicamente depende de la posición del ancla de la función, véase Fig. 2.12. En el escenario (a) el ancla de la función de nivel $m$ coincide con el centro de una celda de nivel *(m-1)* del quadtree, el escenario (b) es el resto de

las funciones, es decir, el ancla de la función de nivel $m$ no corresponde al centro de ninguna celda de nivel *(m-1)*. Segundo, teniendo en cuenta todas las consideraciones acerca de los valores de $\Delta^{\xi}$ y $\Delta^{\eta}$, podemos clasificar todos los soportes en nueve tipos que se ilustran en la Fig. 2.12. Los seis primeros tipos corresponden al escenario (a) y los otros tres tipos corresponden al escenario (b).

## 2.7. Propiedades de los espacios spline propuestos

A continuación se resumen las propiedades y algunas características de los espacios spline construidos con nuestro método.

Nuestra conjetura: para cualquier T-mesh 0-balanceada $T$ sobre el dominio $\Omega$, el conjunto de funciones spline definido de acuerdo con nuestra estrategia genera el espacio $S_T(\Omega) = \text{span}\{N_\alpha : \alpha \in A_T\}$ con las siguientes propiedades:

1. *Las funciones* $\{N_\alpha\}_{\alpha \in A_T}$ *son* $C^2$*-continuas.*
2. *No-negatividad:* $N_\alpha \geq 0$.
3. *Las funciones* $\{N_\alpha\}_{\alpha \in A_T}$ *son linealmente independientes.*
4. *Partición de unidad no-negativa:* $\sum_{\alpha \in A_T} c_\alpha N_\alpha = 1$, *donde* $c_\alpha \geq 0$.
5. *Encaje de espacios:* $T_1 \subset T_2 \Rightarrow S_{T_1} \subset S_{T_2}$.
6. *Capacidad de reproducir polinomios:* $\mathbb{P}_3(\Omega) \subset S_T(\Omega)$.

Las dos primeras propiedades son evidentes. La tercera, la cuarta y la quinta tienen que ser demostradas. La sexta propiedad se deduce de la quinta, ya que el espacio inicial B-spline posee esta propiedad.

Por ahora solo podemos proporcionar una demostración del comportamiento de anidación de los espacios y la partición de unidad no-negativa para el caso 2D.

# CAPÍTULO 3

## Parametrización del dominio computacional

En este capítulo exponemos un método para obtener una parametrización spline de geometrías complejas 2D y 3D. Los modelos CAD sólo proporcionan información sobre la superficie de un sólido, pero el análisis isogeométrico requiere la representación volumétrica de la geometría. El problema de la parametrización de un volumen 3D es todavía un problema abierto. Aquí presentaremos algunos resultados preliminares sobre nuestro método de parametrización.

## 3.1. El método del Meccano. Parametrización volumétrica de los sólidos

Actualmente disponemos de un método que permite obtener la parametrización spline de un sólido. El método ha sido desarrollado dentro del grupo de investigación [58, 59], y está basado en la idea del método del Meccano [60], ver Fig. 3.1. Primero, el método del Meccano particiona la superficie del dominio en seis parches que son parametrizados, usando el método de Floater, sobre cada cara de un cubo en el espacio paramétrico. El cubo se subdivide en 6 tetraedros que siguen la estructura de refinamiento de Kossaczký. Esta malla de tetraedros en el espacio paramétrico se refina adaptativamente con respecto a la superficie en el espacio físico. Una vez obtenida una malla de volumen en el espacio paramétrico, cuya superficie se adapta a la superficie del sólido en el espacio físico, se procede a la deformación de los tetraedros en el espacio físico para que la calidad de la malla resultante en el espacio físico sea óptima. Este proceso de deformación se realiza mediante un proceso de suavizado y desenredo, que sólo modifica la posición de los nodos sin afectar a su conectividad. Por este motivo podemos decir que las mallas en el espacio paramétrico y en el espacio físico son isomórficas. Dado que cualquier punto del volumen es determinado por las coordenadas baricéntricas del tetraedro en el que se encuentra, podemos definir un mapeo uno-a-uno entre la malla paramétrica y la malla física, lo que proporciona una parametrización volumetrica del sólido. En la Figura 3.2 podemos ver los distintos pasos del método del Meccano.

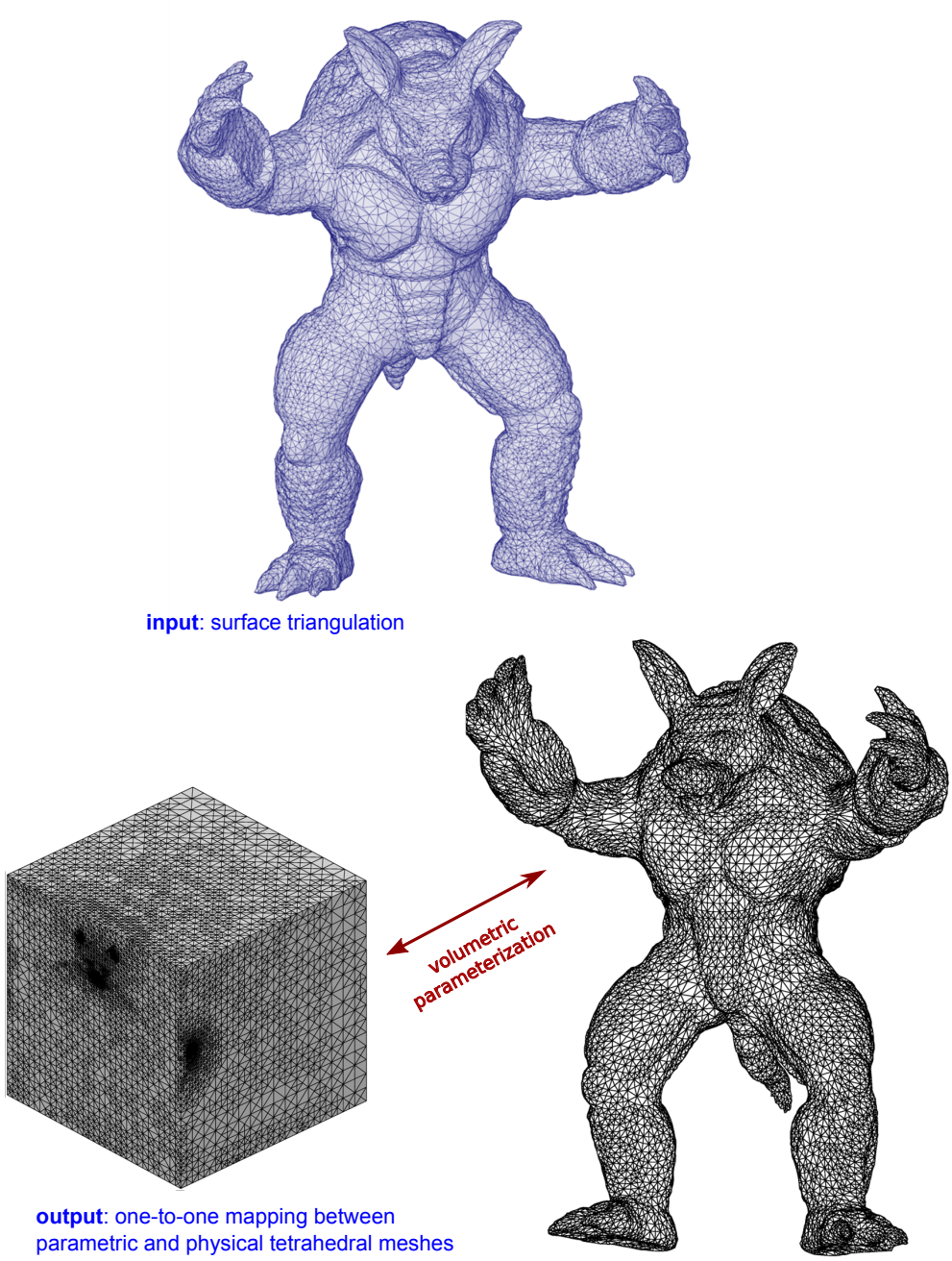
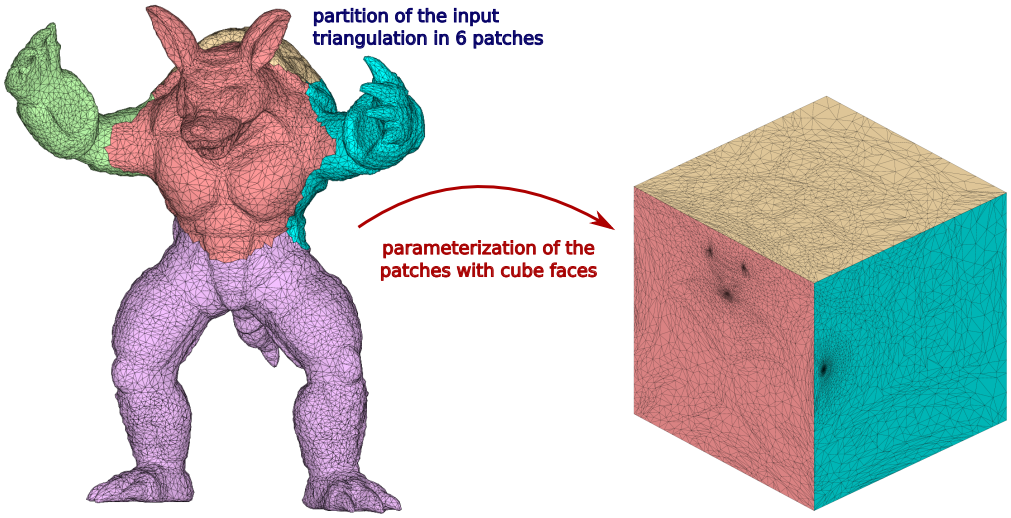



**Figura 3.1:** *Método del Meccano para la generación automática de mallas de tetraedros.*

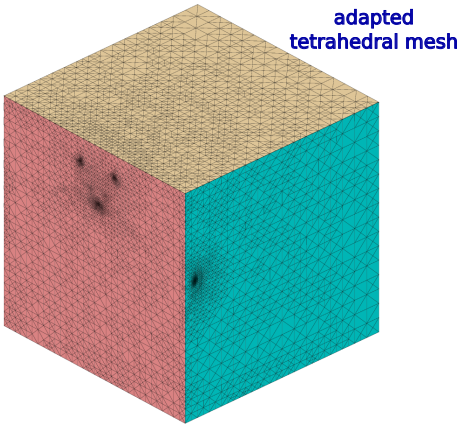
El método del Meccano fue diseñado para la generación automática de mallas de tetraedros, pero su capacidad de generar una parametrización volumétrica entre el espacio paramétrico y el espacio físico puede ser explotada en el análisis isogeométrico para la parametrización volumétrica mediante splines. Para ello, se genera una T-mesh con la misma resolución que la malla de tetraedros. Se cons-

truye un mapeo trivariado de splines $\mathbf{S} : \hat{\Omega} \to \Omega$ imponiendo las condiciones de interpolación sobre los puntos del espacio físico dados por la parametrización del método del Meccano. La Figura 3.3 muestra el resultado final del algoritmo.

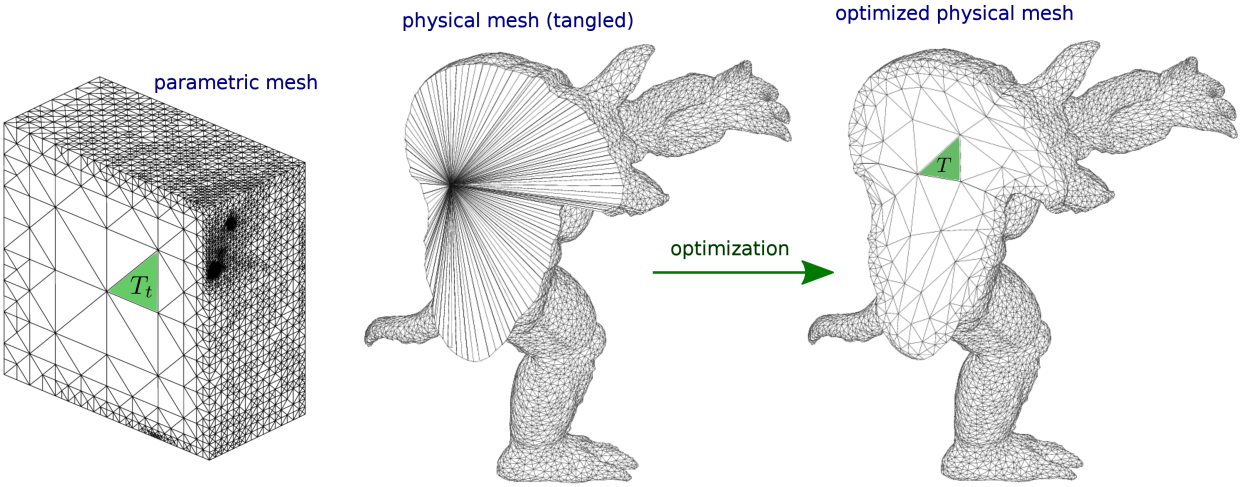
Los resultados obtenidos con este método sólo son buenos para geometrías poco distorsionadas. Para conseguir mejores parametrizaciones spline es conveniente optimizar la T-mesh directamente. Esto es objeto de investigación por nuestro grupo en actualidad. Un resumen de ello será expuesto a continuación.




(a) Parametrización de la superficie del sólido.




(b) Malla de tetraedros adaptada.




(c) Optimización de la malla de tetraedros.

**Figura 3.2:** *Etapas del método del Meccano.*

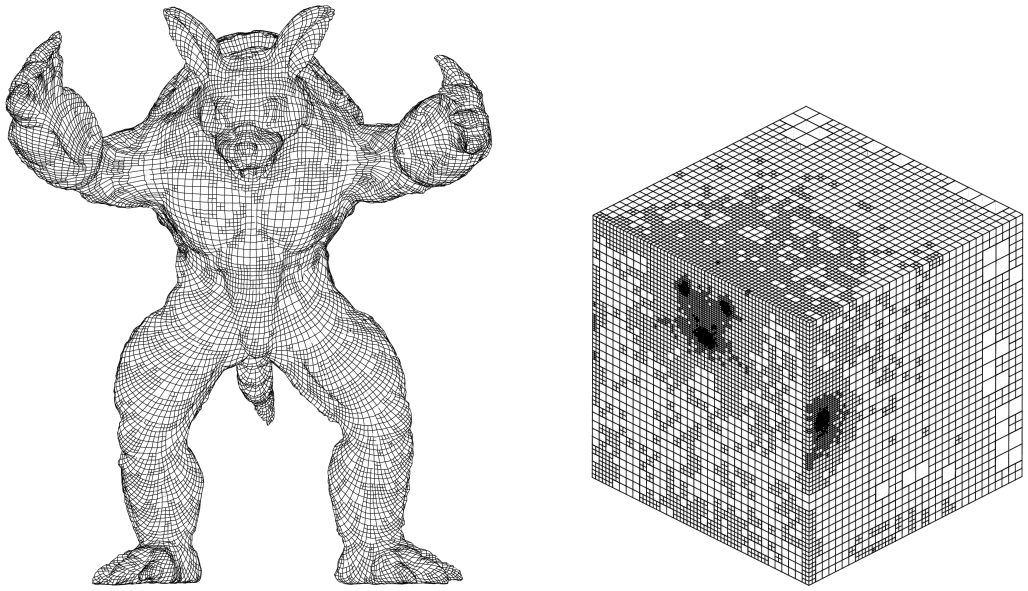


**Figura 3.3:** *T-mesh en el espacio paramétrico y representación T-spline de la geometría.*

## 3.2. Parametrización para geometrías 2D

La parametrización con splines de geometrías 2D se basa en el método del Meccano combinado con un procedimiento novedoso de suavizado y desenredo de T-mesh. Dada la geometría de la frontera de un objeto 2D, este método genera una parametrización de alta calidad entre una geometría 2D y el dominio paramétrico, en general un cuadrado de tamaño unidad. Una descripción detallada del método se puede consultar en [62].

El método incluye las siguientes etapas:

1. *Parametrización de la frontera y construcción de una T-mesh adaptada:* Se define una correspondencia biyectiva entre la frontera del objeto y la frontera del espacio paramétrico. A continuación se genera una T-mesh refinando en la malla inicial los elementos, para los que la distancia entre la proyección de las arista de la frontera física distan más que una tolerancia. En este proceso se crea una proyección de los nodos frontera de la malla del espacio paramétrico al espacio físico.

2. *Optimización de la T-Mesh:* Se calcula la posición óptima de los nodos interiores de la T-mesh usando un proceso de suavizado y desenredo.

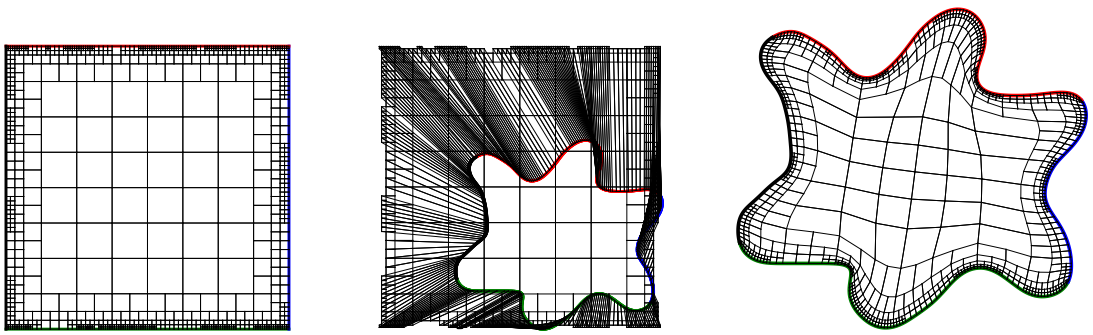
3. *Construcción de la representación de la geometría mediante T-splines:* Se obtiene un mapeo spline imponiendo condiciones de interpolación. Se toman como puntos de interpolación los vértices del espacio físico de la T-mesh.

4. *Refinamiento adaptativo para mejorar la calidad de la malla:* Cuando la calidad de la malla resultante no es satisfactoria, adaptamos la malla para incrementar los grados de libertad en las zonas donde la distorsión entre el espacio paramétrico y el físico es muy grande. Una vez refinada la malla se vuelve al paso 2 iterativamente hasta que no haga falta refinar.

A continuación describiremos brevemente cada una de las etapas.

### 3.2.1. Parametrización de la frontera y construcción de una T-mesh adaptada

Para definir una parametrización entre la frontera de la geometría y el espacio paramétrico necesitamos seleccionar cuatro puntos de la frontera correspondientes a los cuatro vértices del cuadrado. Estos cuatro puntos dividen la frontera en cuatro partes cada una de las cuales es proyectada a las aristas del espacio paramétrico mediante una parametrización que conserva longitudes. A continuación generamos una T-mesh adaptada a la frontera. El proceso de adaptación se realiza proyectando todos los puntos de la frontera del espacio paramétrico al espacio físico y comprobando si la distancia entre la arista que une los dos puntos y la frontera de la geometría es mayor que una tolerancia $\epsilon$, caso en el que se debe refinar el elemento. Los nodos interiores se recolocan usando el método propuesto en el próximo capítulo. En las figuras 3.4(a) y 3.4(b) se representa el resultado de la T-mesh en el espacio paramétrico y físico respectivamente.



(a) (b) (c)

**Figura 3.4:** *Distintas etapas de la construcción de la T-mesh para la geometría* Spot*: (a) T-mesh adaptada a la geometría en el espacio paramétrico; (b) Proyección de la frontera en el espacio físico con la T-mesh enredada (las aristas coloreadas representan los cuatro lados de la frontera); (d) T-mesh final optimizada*

### 3.2.2. Optimización de la T-mesh

El proceso de optimización de la T-mesh es la pieza clave que permite obtener parametrizaciones de la T-mesh de buena calidad.

La optimización de la malla es un proceso iterativo que se realiza mediante la recolocación de cada nodo interior de la malla atendiendo a un criterio de calidad hasta que todos los nodos están en su posición óptima. Para determinar la calidad se usan las métricas para elementos cuadriláteros definidas por Knupp en [63, 64]. El valor de la calidad está comprendido entre 0 (la peor calidad) y 1 (calidad óptima). La calidad para un cuadrilátero se mide en función de calidades de ciertos triángulos que definen el cuadrilátero.

### 3.2.3. Construcción de la representación spline de la geometría

El objetivo es encontrar una parametrización global que proyecte el espacio paramétrico en el espacio físico $\mathbf{S} : \hat{\Omega} = [0,1]^2 \rightarrow \Omega$.

La representación de la superficie en el espacio físico se calcula como una combinación lineal de la funciones spline.

$$\mathbf{S}\left(\boldsymbol{\xi}\right) = \sum_{\alpha \in A} \mathbf{P}_\alpha \, N_\alpha \left(\boldsymbol{\xi}\right) \tag{3.1}$$

donde $\mathbf{P}_\alpha \in \mathbb{R}^2$ es el punto de control correspondiente a la función $\alpha$-th.

Para encontrar los puntos de control $\mathbf{P}_\alpha$ necesitamos imponer condiciones de interpolación. Para ello resolvemos el siguiente sistema lineal de ecuaciones

$$\mathbf{x}_\beta = \mathbf{S}\left(\boldsymbol{\xi}_\beta\right) = \sum_{\alpha \in A} \mathbf{P}_\alpha \, N_\alpha \left(\boldsymbol{\xi}_\beta\right), \quad \forall \boldsymbol{\xi}_\beta, \ \beta \in A \tag{3.2}$$

donde $\boldsymbol{\xi}_\beta$ son los puntos de interpolación en el espacio paramétrico y $\mathbf{x}_\beta$ son sus imágenes en el espacio físico determinados en el proceso de optimización.

En la Fig. 3.5 podemos ver la parametrización resultante para la geometría *Spot*.

### 3.2.4. Evaluación de la calidad y su mejora

En este punto tenemos que evaluar la parametrización que hemos generado, ya que pueden existir variaciones grandes del Jacobiano que pueden aumentar los errores numéricos.

Usaremos el *mean ratio Jacobian* para evaluar la calidad de la parametrización construida. El mapa de colores del *mean ratio Jacobian* para la geometría *Spot* se muestra in Fig. 3.6

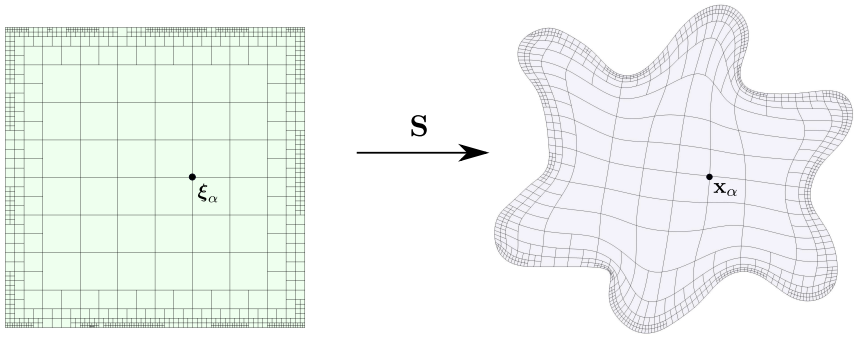
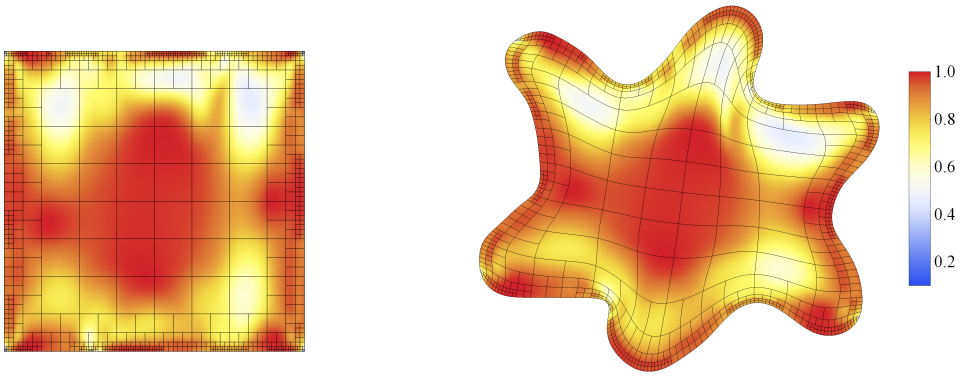



**Figura 3.5:** *Geometría* Spot *con* 844 *elementos y* 1456 *puntos de control. Dominio paramétrico y la representación spline del dominio físico.*




(a) Dominio paramétrico. (b) Dominio físico.

**Figura 3.6:** *Geometría-test* Spot. *Mapa de colores del mean ratio Jacobian.*

## Mean ratio Jacobian

El *mean ratio Jacobian* se define como

$$J_r(\boldsymbol{\xi}) = \frac{2 \det(J)}{\|J\|^2}, \tag{3.3}$$

donde $J$ es la matriz Jacobiana de la proyección $\mathbf{S}$ en el punto $\boldsymbol{\xi} = (\xi, \eta)$ y $\|J\|$ su norma de Frobenius.

El *mean ratio Jacobian* en un punto es minimo, 1, cuando el mapeo conserva la ortogonalidad y produce la misma distorsión longitudinal en ambas direcciones paramétricas, en otras palabras, cuando el mapeo es conforme.

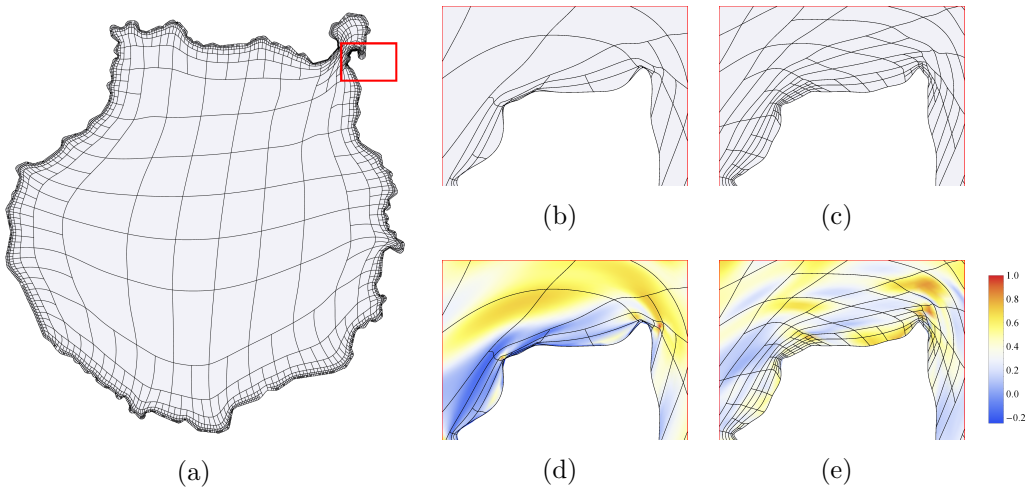



**Figura 3.7:** *Refinamiento adaptativo para mejorar calidad de la parametrización de la geometría de Gran Canaria. (a) Representación spline del dominio; (b) parametrización inicial con Jacobiano negativo; (c) parametrización spline sin Jacobiano negativo despúes de aplicar la estrategia adaptativa; (d) mean ratio Jacobian del mapeo inicial; (e) mean ratio Jacobian del mapeo final.*

### Refinamiento adaptativo

La parametrización de geometrías complejas implica una distorsión alta, e incluso en algunos casos, pueden aparecer celdas con Jacobiano negativo. Esto puede ser debido la falta de grados de libertad asociados a los nodos internos. A fin de mejorar la calidad de la malla en estos casos añadimos grados de libertad refinando las celdas con valores bajos del mean ratio Jacobian. Calculamos el *mean ratio Jacobian* de cada elemento usando $16 = 4 \times 4$ puntos de cuadratura. Un elemento se refina si el valor del *mean ratio Jacobian* en un punto de cuadratura es menor a un cierto umbral $\delta$. Si se ha refinado algún elemento, se vuelve a repetir el proceso de optimización y construcción de la geometría hasta que no haya elementos a refinar. La Fig. 3.7 ilustra la estrategia propuesta. La malla inicial con 3439 celdas produce un mapeo paramétrico spline con baja calidad en algunas áreas y auto-intersecciones en la parte noreste de la isla. Se aplicaron refinamientos adicionales con $\delta = 0.2$. Después del refinamiento adaptativo llegamos a una malla con 3577 celdas y Jacobiano positivo en todo el dominio. El valor mínimo del *mean ratio Jacobian* en los puntos de cuadratura es 0.21. En la Fig. 3.8 se puede ver el resultado completo de la parametrización de la geometría de Gran Canaria.

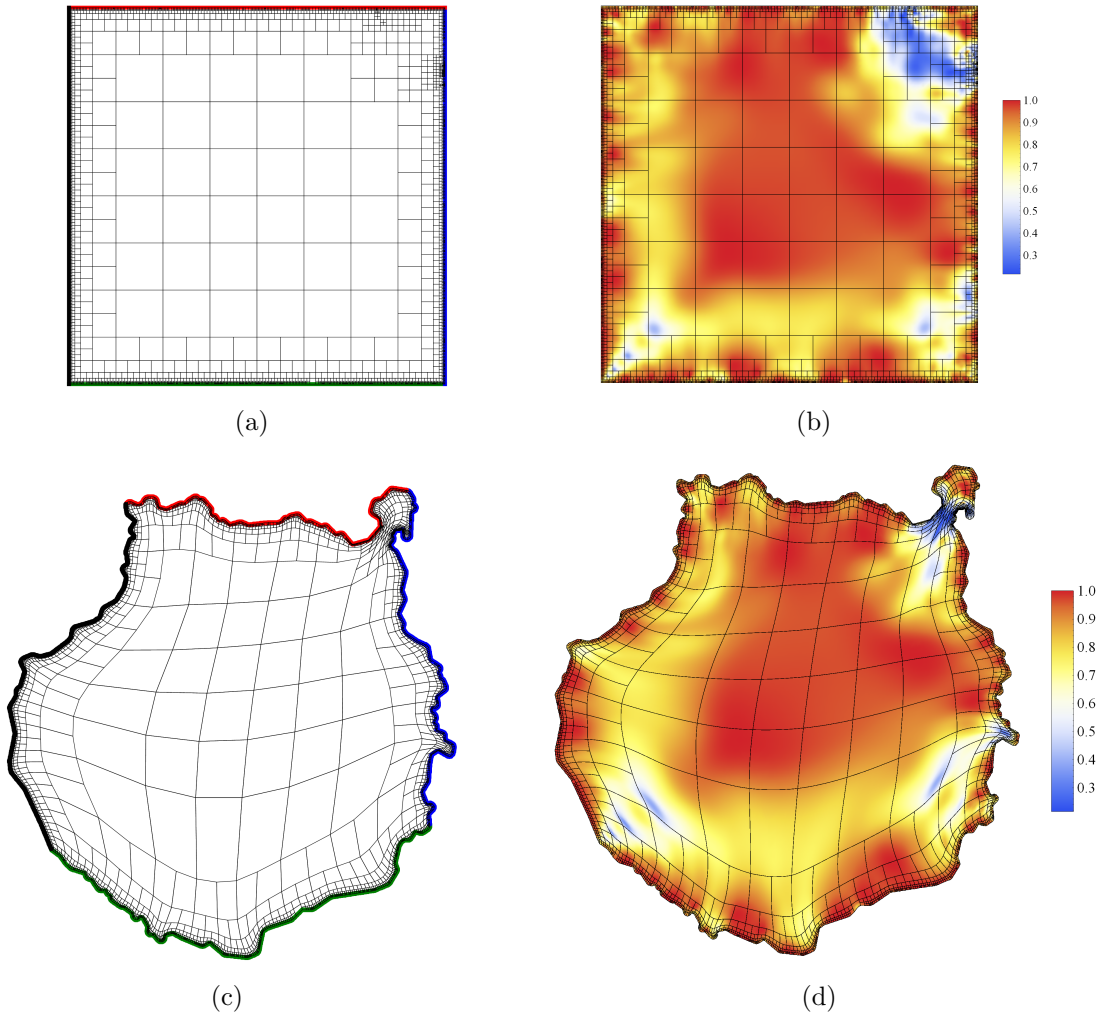



**Figura 3.8:** *Parametrización de la isla de Gran Canaria con* 3577 *elementos y* 6054 *puntos de control. (a) T-mesh en el espacio paramétrico; (b) Valor del* mean ratio Jacobian *en el espacio paramétrico; (c) T-mesh en el espacio físico (las aristas coloreadas muestran la correspondencia de la frontera entre el espacio físico y el paramétrico); (d) Valor del* mean ratio Jacobian *en el espacio físico.*

## 3.3. Generalización a 3D

Para generalizar el método a 3D se necesita una técnica de optimización de T-mesh. Esta tarea no es trivial y se está estudiando en el marco de otra tesis doctoral en el grupo de investigación. De momento, solo podemos realizar una parametrización basada en la optimización de mallas uniformes de hexaedros (sin T-junctions). Mediante este método somos capaces de generar parametrizaciones de volumen para geometrías ligeramente distorsionadas.

El mapeo spline trivariado se obtiene imponiendo condiciones de interpola-

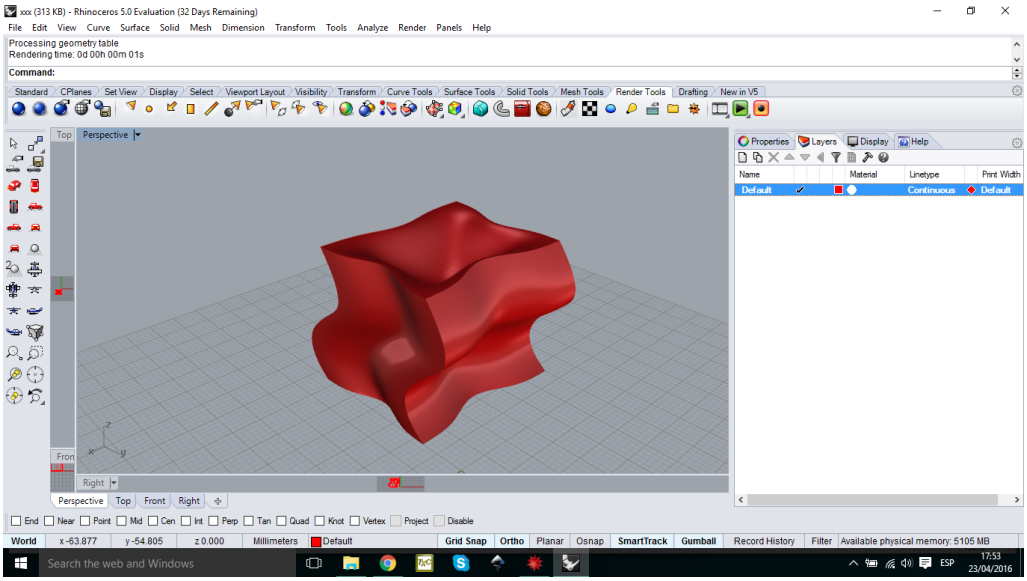
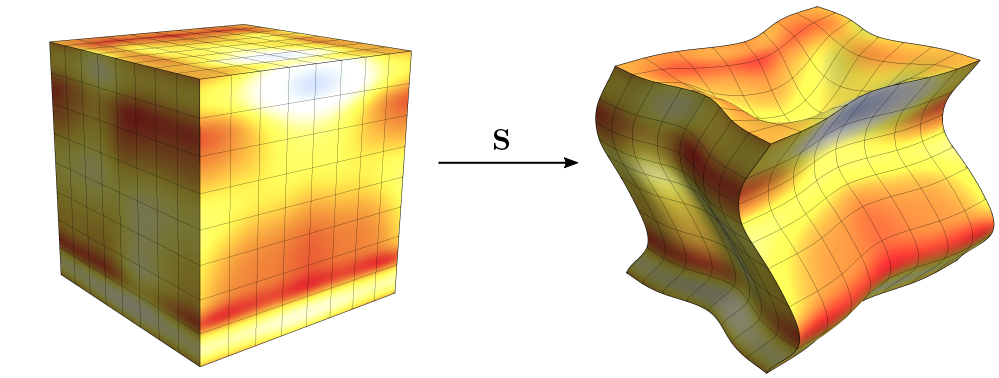


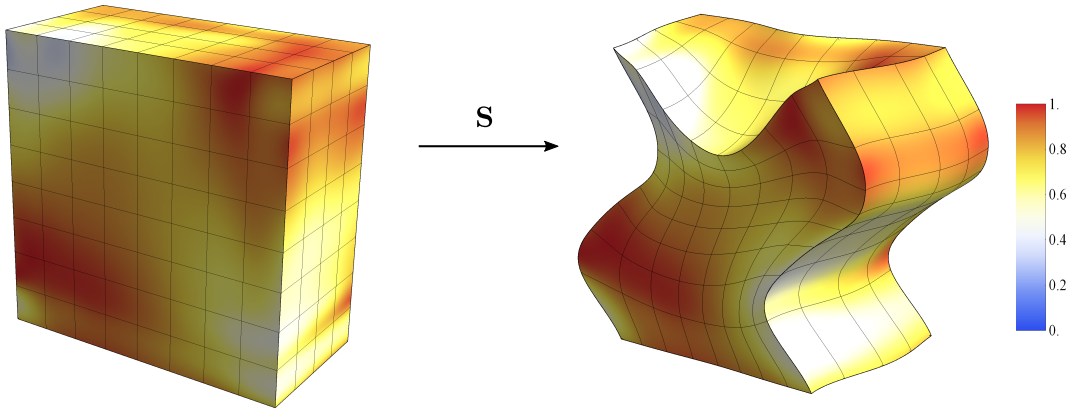

**Figura 3.9:** *Geometría 3D ligeramente distorsionada diseñada en Rhinoceros.*

ción, tomando como puntos de interpolación los vértices de la malla optimizada en el espacio físico. La calidad la medimos con el *mean ratio Jacobian*, definido como $J_r = \frac{3 \det J}{\|J\|^3}$, donde $J$ es la matriz Jacobiana del mapeo paramétrico $\mathbf{S}$

Este método se ha aplicado a la geometría diseñada en *Rhinoceros*, ver Fig. 3.9. La parametrización resultante junto con los valores del *mean ratio Jacobian* se muestran en la Fig. 3.10.




(a)




(b)

**Figura 3.10:** *Parametrización resultante de la geometría diseñada en Rhinoceros y su calidad. El mapa de colores muestra los valores del mean ratio Jacobian.*

# CAPÍTULO 4

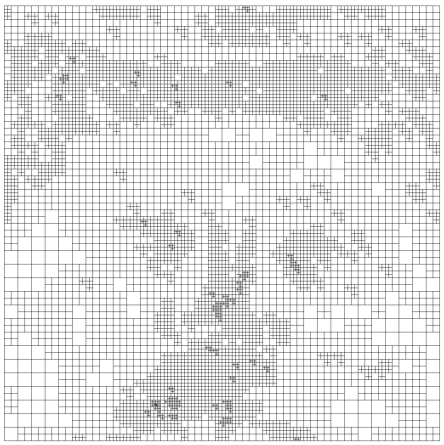
## Pruebas de la estrategia. Ejemplos computacionales

En este capítulo ponemos a prueba los espacios spline propuestos para su uso en diseño geométrico y el Análisis Isogeométrico. Se analiza la capacidad de aproximación de las spline propuestas en diversos tipos de problemas con singularidades, donde es necesario un refinamiento adaptativo para lograr una buena precisión de la solución numérica. Para los ejemplos de análisis hemos considerado problemas elípticos de segundo orden, donde se conoce la solución exacta.

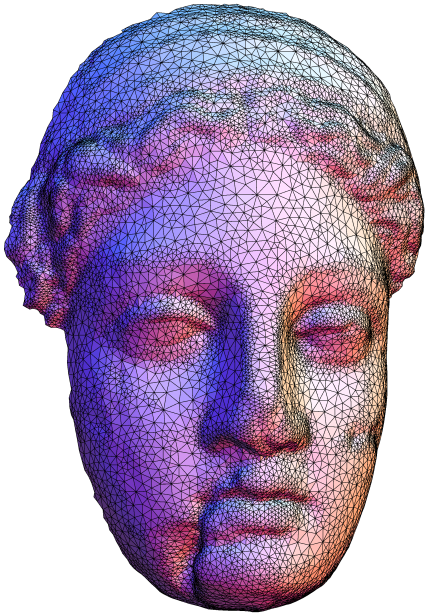
## 4.1. Modelado geométrico. Representación spline de una superficie a partir de su triangulación

Comenzamos con el ejemplo de modelado geométrico. Aquí construimos una representación spline de una superficie dada por su triangulación. Primero, se obtiene una parametrización global de la triangulación de la superficie utilizando el método propuesto por M. Floater en [69]. Como resultado, obtenemos un mapeo uno-a-uno entre la triangulación plana del dominio paramétrico y la triangulación de la superficie. Luego, con el fin de reproducir bien todas las características de la superficie, construimos una T-mesh adaptada a la triangulación plana. Con este fin hemos elegido el siguiente criterio. Se parte de una malla uniforme y se refina hasta que cada celda contiene a lo sumo un cierto número de nodos de la triangulación de entrada (3 en nuestro caso). Una vez hecho esto, se define un espacio spline sobre la malla adaptada $T$. La aproximación spline de la superficie se construye como una combinación lineal de funciones de base bivariadas

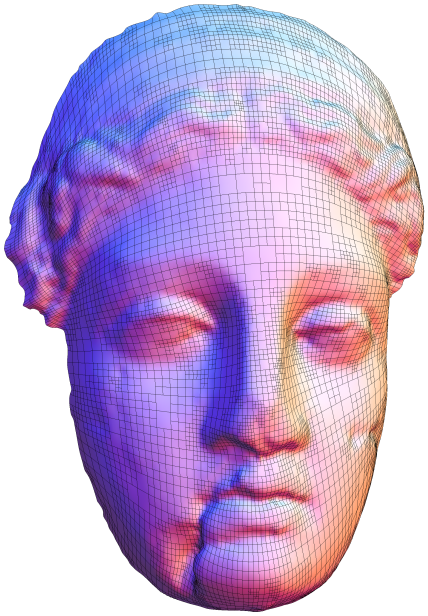
$$S(\boldsymbol{\xi}) = \sum_{\alpha \in A_T} \mathbf{P}_\alpha \, N_\alpha \left(\boldsymbol{\xi}\right).$$



(a) T-mesh paramétrica.



(b) Triangulación de la superficie-dato.



(c) Representación spline de la superficie.

**Figura 4.1:** *Igea. Representación spline de la superficie a partir de su triangulación.*

Los puntos de control $\mathbf{P}_\alpha \in \mathbb{R}^3$ se encuentran mediante la imposición de las condiciones de interpolación

$$\boldsymbol{x}_\beta = \sum_{\alpha \in A_T} \mathbf{P}_\alpha \, N_\alpha \left( \boldsymbol{\xi}_\beta \right), \qquad \forall \boldsymbol{\xi}_\beta, \ \beta \in A_T,$$

(a) Triangulación de la superficie-dato. (b) Representación spline de la superficie.

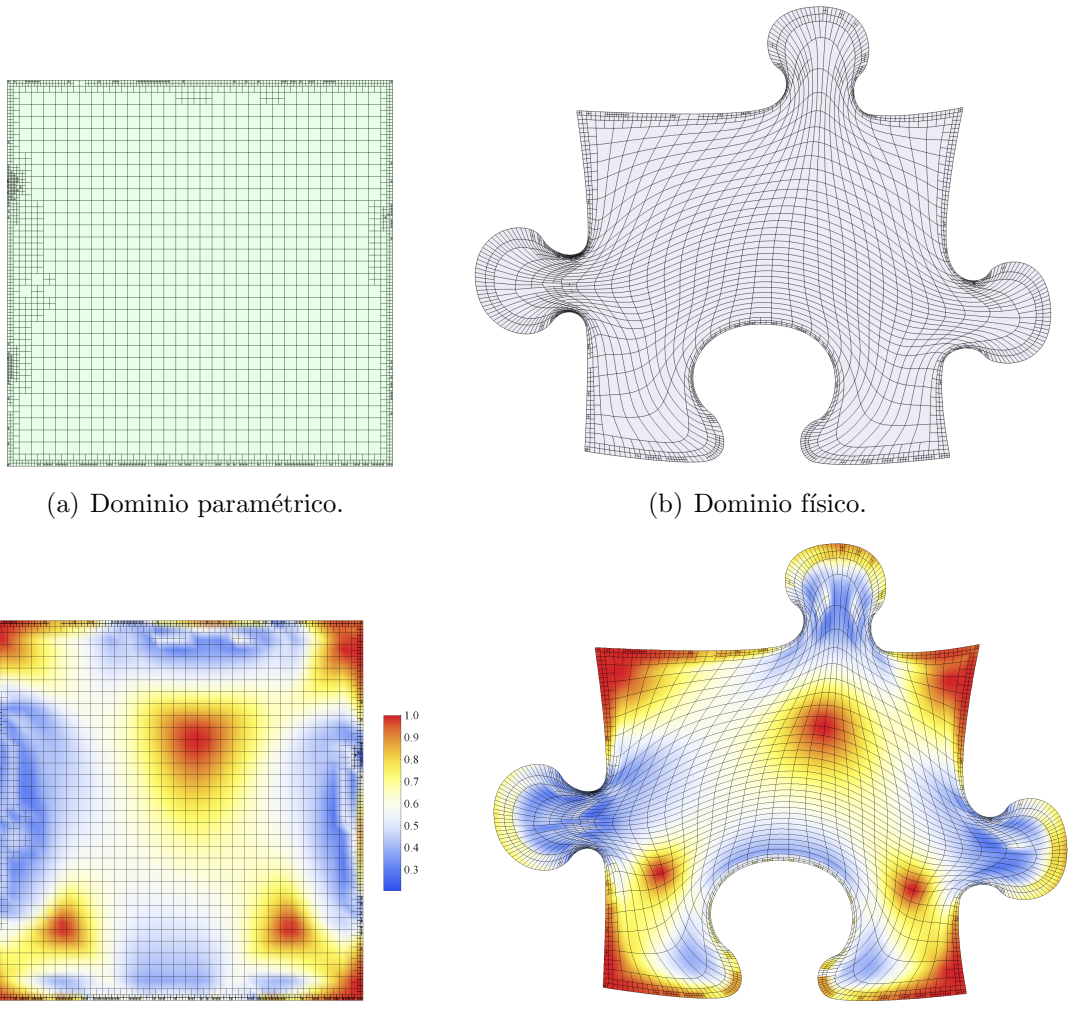
**Figura 4.2:** *Igea. La comparación entre la triangulación original y la aproximación spline.*

donde $\boldsymbol{\xi}_\beta \in \mathbb{R}^2$ son puntos de interpolación en espacio paramétrico y $\boldsymbol{x}_\beta \in \mathbb{R}^3$ sus imagines en el espacio físico determinadas por la parametrizacion triangular.

En la Fig. 4.1 se muestran la superficie spline resultante, la T-mesh paramétrica y la triangulación de la superficie. Podemos observar que la representación spline logra reproducir bien la superficie de entrada. Ademas, se puede apreciar en la Fig. 4.2 que, incluso con menos grados de libertad, la representación spline mejora considerablemente el aspecto del objeto de entrada, ya que obtiene una superficie visiblemente más suave.

## 4.2. Problema de Poisson sobre un dominio complejo

En el siguiente ejemplo resolvemos un problema de Poisson sobre un dominio complejo $\Omega$, ver Fig. 4.3. La parametrización del dominio computacional se realizó mediante el algoritmo descrito en el capítulo 3.2. Esta técnica, basada en un procedimiento de optimización para T-mesh, permite obtener un mapeo paramétrico




(a) Dominio paramétrico.

(b) Dominio físico.

(c) Mean ratio Jacobian.

(d) Mean ratio Jacobian.

**Figura 4.3:** *Puzzle piece. La parametrización del dominio computacional para el problema de Poisson y su calidad (mean ratio Jacobian).*

de buena calidad apto para la aplicación del Análisis Isogeométrico. El mean ratio Jacobian $J_r(\boldsymbol{\xi}) = \frac{2\,\det(J)}{\|J\|^2}$ se utiliza para evaluar la calidad de la parametrización en el sentido de su ortogonalidad y uniformidad. La Figura 4.3 muestra la parametrización resultante y el mapa de colores de su mean ratio Jacobian.

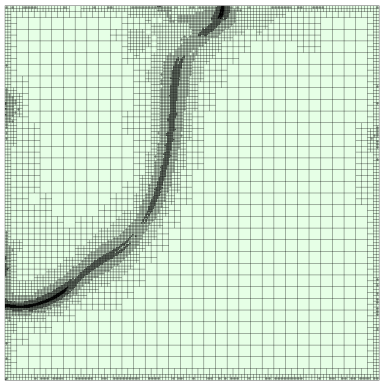
La solución exacta para el problema de Poisson con condición de contorno tipo Dirichlet es

$$u(r) = \arctan(\alpha(r - r_0)),$$

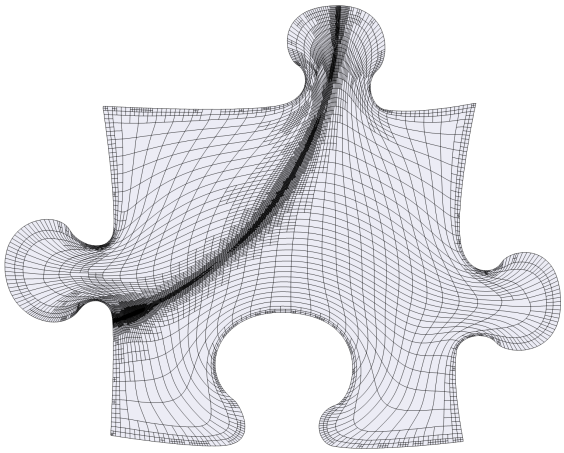
donde $r = \sqrt{(x - x_c)^2 + (y - y_c)^2}$. Es una función con un frente de onda abrupto, donde el parámetro $\alpha$ determina la pendiente del frente de onda y $r_0$ su ubicación. En este ejemplo, $\alpha = 200$ y $r_0 = 0.6$. El centro del frente $(x_c, y_c) = (0, 0)$ está

situado fuera del dominio computacional, por tanto la función es suave en $\Omega$.

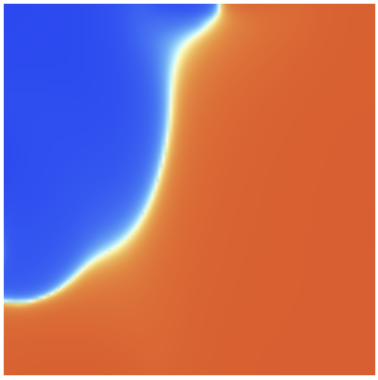
La solución numérica del problema y la malla correspondiente a la iteración final de refinamiento adaptativo se muestran en la Fig. 4.4(a)-(d). Como era de esperar, el estimador de error ha marcado para refinar la region cercana al frente de onda. La evolución del error exacto en la norma $L^2$ y la seminorma $H^1$ se muestran en la Fig. 4.4(e). La comparación con el refinamiento uniforme se muetra en la Fig. 4.4(f). Debido a la suavidad de la solución exacta se han conseguido óptimos ordenes de convergencia tanto para el refinamiento adaptativo como para el uniforme.



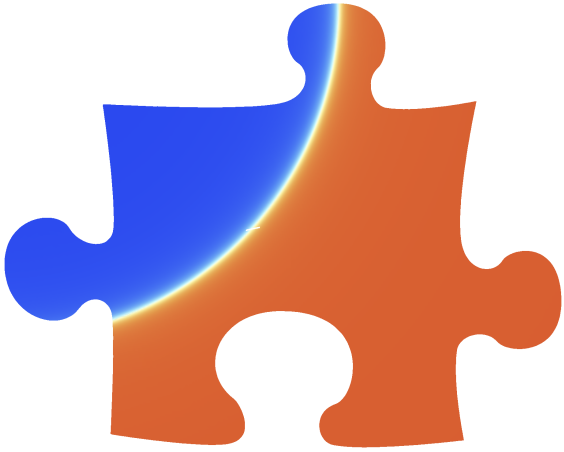
(a) Refinamiento final, dominio paramétrico.



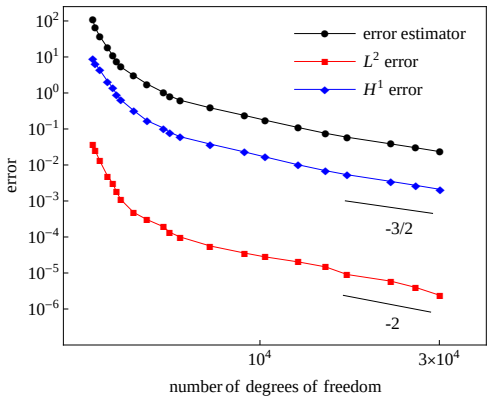
(b) Refinamiento final, dominio físico.



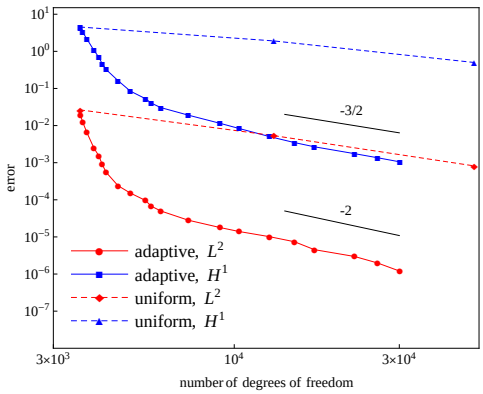
(c) Solución numérica.



(d) Solución numérica.




(e) Convergencia del error para refinamiento adaptativo.




(f) Convergencia del error para refinamiento adaptativo y uniforme.

**Figura 4.4:** *Los resultados del refinamiento adaptativo para el problema de Poisson sobre la pieza de puzzle.*

# CAPÍTULO 5

## Conclusiones y trabajos futuros

## 5.1. Resumen y conclusiones

En esta tesis hemos propuesto una estrategia para construir espacios spline para su uso en el Análisis Isogeométrico y CAD. Nuestro método permite definir fácilmente una base de splines cúbicas $C^2$ sobre una T-mesh con estructura de tipo quadtree/octree 0-balanceada.

Nuestro trabajo estuvo motivado por la incapacidad de los espacios B-spline con estructura tensorial de realizar refinamientos locales, lo cual dificulta su uso en el análisis. Actualmente existen diferentes estrategias para la construcción de espacios splines con capacidad de refinamiento local, pero todas ellas presentan ciertos inconvenientes. Nuestro objetivo era desarrollar un procedimiento alternativo para la construcción de espacios splines con todas las propiedades necesarias para el análisis. Buscábamos, además, que fuera fácil de implementar y que tuviera un coste computacional bajo en 2D y 3D.

En suma, los espacios spline construidos con la estrategia propuesta en esta tesis tienen las siguientes propiedades:

- las blending functions son $C^2$-continuas y linealmente independientes;
- los espacions generados por mallas encajadas son también encajados;
- su implementación es sencilla, tanto en 2D como 3D, y solo requiere una T-mesh de tipo quadtree/octree;
- las bases se definen de forma independiente unas de otras, de manera que el proceso puede ser fácilmente paralelizado;

Esta tesis incluye las siguientes partes principales.

– descripción de las reglas para inferir los vectores de knots y los correspondientes algoritmos para 2D y 3D

– discusión y análisis de las propiedades de los espacios de splines construidos

– demostración del anidamiento de los espacios y de la partición de unidad no-negativa para el caso 2D

– ejemplos de Análisis Isogeométrico y de CAD que imponen un refinamiento adaptativo.

Las propiedades de aproximación de los espacios propuestos han sido testadas en numerosos ejemplos computacionales. Las singularidades que presentan algunos de los problemas test propuestos necesitan un refinamiento adaptativo para alcanzar una solución numérica con la precisión requerida. En todos los ejemplos computacionales se consiguieron tasas de convergencia óptimas. Para alguno de estos ejemplos se compararon las soluciones numéricas proporcionadas por el IGA y por FEM, observándose que la precisión de IGA es algo superior a la de FEM. Creemos que la simplicidad y buen comportamiento de las bases de splines presentadas en esta tesis hacen que estas sean una herramienta atractiva para su aplicación en IGA y en CAD.

También se han presentado brevemente algunos resultados preliminares sobre el método de parametrización desarrollados en nuestra investigación. Recordemos que una parametrización adecuada (de buena calidad) del dominio es clave para la aplicación del Análisis Isogeométrico. En el caso de 3D, el método se basa en la optimización de una malla de tetraedros del sólido a parametrizar. La T-mesh se construye a partir de esta parametrización. En 2D se ha dado un paso más y se prescinde de la triangulación del dominio. El proceso de optimización en que está basado el procedimiento se aplica directamente a la T-mesh. Esta técnica se ha utilizado para la parametrización de algunos dominios complejos utilizados en los ejemplos computacionales.

## 5.2. Trabajos futuros

En cuanto a los espacios de splines propuestos cabe mencionar las siguientes líneas de investigación:

– En primer lugar, se necesita una demostración rigurosa de las propiedades de anidamiento (caso 3D) e independencia lineal (2D y 3D) de los espacios spline propuestos.

– Los espacios de splines propuestos presentan propiedades esenciales para su aplicación en el análisis, sin embargo, podrían mejorarse en algunos aspectos. A saber, puede aparecer un excesivo solapamiento de las bases en algunos problemas, lo que conlleva un incremento en la densidad de la matriz global y un empeoramiento en su condicionamiento. Aunque esta situación solo se presenta en raras ocasiones, sería deseable reducir el soporte de funciones base a fin de limitar el solapamiento. Creemos que esto sería posible modificando en cierta medida las reglas de inferencia de knots propuestas en esta tesis.

También sería interesante que las funciones base que surgieran de estas reglas satisficieran la propiedad de la partición de la unidad de forma estricta. Por último, también sería importante extender la estrategia actual a splines de orden diferente a tres.

– En este trabajo los espacions spline están definidos sobre un dominio paramétrico cuadrado o cúbico. Sin embardo, la parametrización de dominios más complejos con una alta distorsión requiere de espacios paramétricos más generales, con formas más parecidas al dominio físico. Por tanto, es necesario elaborar un procedimiento para construir splines sobre este tipo de espacios paramétricos, intentando conservar la regularidad de las bases. Otra alternativa sería construir espacios splines sobre triangulaciones del dominio computacional. Las splines sobre triangulaciones pueden ofrecer mayor flexibilidad que las splines definidas a través de productos tensoriales. Sin embargo esta línea de investigación todavía está poco desarrollada. Las opciones actuales presentan una gran complejidad y tienen bastantes limitaciones. Se necesita una estrategia sencilla para definir $C^2$ splines sobre triangulaciones (2D y 3D) capaces de competir con las splines producto tensorial.

# Índice de figuras